Analysis and Design of Cryptographic Hash Functions

Bart PRENEEL

February 2003

Acknowledgements

I like prefaces. I read them. Sometimes I do not read any further. *Malcolm Lowry*

At the end of this Ph.D. project it is a pleasure to thank everybody who has helped me along the way.

In the first place, I would like to express my thanks to my supervisors, Prof. R. Govaerts and Prof. J. Vandewalle who have introduced me to the field of cryptography. I appreciate their guidance and support, and I value the interesting discussions on various subjects we had. They also have taught me many practical aspects of research. I also would like to thank them for the many opportunities I have received to broaden my knowledge and to discuss my work with other researchers.

I would also like to thank Prof. A. Oosterlinck for giving me the opportunity to work at ESAT, and Prof. A. Haegemans and Prof. J.-J. Quisquater for reviewing this manuscript. Prof. A. Barbé, Prof. J. Berlamont, and Prof. P. Landrock are gratefully acknowledged for serving on the jury.

I would like to mention my COSIC colleagues, Joan Daemen, Rita De Wolf, Mark Vandenwauver, Luc Van Linden, and Jan Verschuren, who created a nice working atmosphere. Special thanks go to Antoon Bosselaers for the many years of collaboration, for the assistance with the software, and for the valuable comments on all my drafts. I would like to thank Ria Vanden Eynde for the active proof reading and for the moral support.

I also would like to acknowledge the collaboration with the students who have been working under my supervision. I have learned many things from my colleagues of the RIPE project and from the experts of ISO/IEC JTC1/SC27/WG2.

Many people in the cryptographic community have contributed to this research, through interesting discussions or by providing useful information. I would like to mention especially Prof. T. Beth, dr. C. Carlet, Prof. D. Chaum, dr. D. Coppersmith, Prof. I. Damgård, dr. B. den Boer, dr. A. Jung, L. Knudsen, dr. X. Lai, dr. K. Martin, dr. W. Meier, dr. K. Nyberg, dr. L. O'Connor, dr. R. Rueppel, dr. B. Sadeghiyan, dr. O. Staffelbach, dr. H. Tiersma, and dr. Y. Zheng.

Finally I would like to acknowledge the support of the Belgian National Science Foundation (N.F.W.O.).

Abstract

The subject of this thesis is the study of cryptographic hash functions. The importance of hash functions for protecting the authenticity of information is demonstrated. Applications include integrity protection, conventional message authentication and digital signatures. Theoretical results on cryptographic hash functions are reviewed. The information theoretic approach to authentication is described, and the practicality of schemes based on universal hash functions is studied. An overview is given of the complexity theoretic definitions and constructions. The main contribution of this thesis lies in the study of practical constructions for hash functions. A general model for hash functions is proposed and a taxonomy for attacks is presented. Then all schemes in the literature are divided into three classes: hash functions based on block ciphers, hash functions based on modular arithmetic and dedicated hash functions. An overview is given of existing attacks, new attacks are demonstrated, and new schemes are proposed. The study of basic building blocks of cryptographic hash functions leads to the study of the cryptographic properties of Boolean functions. New criteria are defined and functions satisfying new and existing criteria are studied.

Contents

List of notations

Such is the advantage of a well-constructed language that its simplified notation often becomes the source of profound theories. *P.S. Laplace*

List of abbreviations

$AGL(n)$	:	affine linear group of $GF(2^n)$
ANSI	:	American National Standards Institute
CBC	:	Cipher Block Chaining
CCITT	:	International Telegraph and Telephone Consultative Committee
CFB	:	Cipher Feedback
$CI(m)$	:	Correlation Immune of order m
$CIB(m)$	:	Balanced and Correlation Immune of order m
$CIN(m)$	:	Non-balanced and Correlation Immune of order m
CPU	:	Central Processing Unit
CRC	:	Cyclic Redundancy Check
CRF	:	Collision Resistant Function
CRHF	:	Collision Resistant Hash Function
ECB	:	Electronic Codebook
EFT	:	Electronic Funds Transfer
$EPC(k)$	:	Extended Propagation Criterion of degree k
ETEBAC	:	Echanges Télématiques Entre les Banques et leurs Clients
FIPS	:	Federal Information Processing Standard
$GF(p^n)$	:	Galois Field with p^n elements
$GL(n)$	:	general linear group of $GF(2^n)$
IEC	:	International Electrotechnical Committee
ISO	:	International Organization for Standardization
IV	:	Initial Value
MAC	:	Message Authentication Code
MDC	:	Manipulation Detection Code
NIST	:	National Institute for Standards and Technology (US)
OFB	:	Output Feedback
OFBNLF	:	Output Feedback with a Non-Linear Function
OWF	:	One-Way Function
OWHF	:	One-Way Hash Function
$PC(k)$	:	Propagation Criterion of degree k
PCBC	:	Plaintext-Ciphertext Block Chaining
PSG	:	Pseudo-random String Generator
RCC	:	Random Code Chaining
RIPE	:	Race Integrity Primitives Evaluation
SAC	:	Strict Avalanche Criterion
SRF	:	Sibling Resistant Function
SV	:	Starting Variable
UOWHF	:	Universal One-Way Hash Function

List of cryptographic algorithms

BCA	:	Binary Condensing Algorithm
BMAC	:	Bidirectional MAC
DEA	:	Data Encryption Algorithm
DES	:	Data Encryption Standard
DSA	:	Decimal Shift and Add algorithm
	:	Digital Signature Algorithm
DSAA	:	Dect Standard Authentication Algorithm
FEAL	:	Fast data Encipherment Algorithm
FFT-hash	:	Fast Fourier Transform hash function
IDEA	:	International Data Encryption Algorithm
IPES	:	Improved Proposed Encryption Standard
MAA	:	Message Authenticator Algorithm
MD-x	:	Message Digest x
PES	:	Proposed Encryption Standard
QCMDC	:	Quadratic Congruential MDC
QCMDCV4	:	Quadratic Congruential MDC Version 4
RSA	:	Rivest Shamir Adleman
SHA	:	Secure Hash Algorithm
TRASEC	:	TRAnsmission Security

List of mathematical symbols

$x \| y$	:	the concatenation of the binary strings x and y
$\lfloor z \rfloor$	:	the greatest integer less than or equal to z
$\lceil z \rceil$	:	the least integer greater than or equal to z

Chapter 1

Authentication and Privacy

The beginning is easy; what happens next is much harder.

1.1 Introduction

The title of this chapter will sound familiar and yet a little odd to anyone who is interested in cryptography. The explanation is that the frequently cited 1979 overview paper of W. Diffie and M. Hellman in the Proceedings of the IEEE [96] is entitled "Privacy and Authentication: an introduction to cryptography". In spite of the title, this overview paper is devoted almost completely to the protection of privacy. This is not surprising, since at that time cryptology was mainly concentrating on the privacy problem, and it was widely believed that the authentication problem was only a subproblem, in the sense that protection of authenticity would follow automatically from privacy protection. W. Diffie and M. Hellman conclude *"The problems of privacy and authentication are closely related and techniques for solving one can frequently be applied to the other"*.

However, their seminal 1976 paper [95] has given cryptography a new orientation, through the introduction of new concepts and definitions. These concepts gave birth to new ideas and approaches, resulting in a clear separation of the privacy and authentication problem. About the protection of authenticity, they state that *"Not only must a meddler be prevented from injecting totally new, authentic messages into a channel, but he must be prevented from creating apparently authentic messages by combining, or merely repeating, old messages which he has copied in the past. A cryptographic system intended to guarantee privacy will not, in general, prevent this latter form of mischief."* The development of both theoretical and practical cryptographic systems to guarantee authenticity has been an important research topic in the cryptographic community during the last fifteen years.

In this chapter basic concepts of privacy and authentication will be briefly ex-

plained. Subsequently, it will be shown that privacy and authentication are two different concepts. This will require the description of a model for symmetric and asymmetric cipher systems and an explanation of how cryptographically secure hash functions can be used to provide authentication and to optimize digital signature schemes. A taxonomy will be given for authentication systems, comparing the information theoretic approach, the complexity theoretic approach, and the system based or practical approach. Finally an outline of this thesis will be given and the main contributions will be described.

1.2 Background and definitions

It is well known that the concealment of information or protection of *privacy* is as old as writing itself. Human ingenuity found many ways to conceal information: steganography, i.e., the hiding of the mere existence of a message, codes, where words or combinations of words are replaced by fixed symbols, and cryptology or ciphers, where information is transformed to render it useless for the opponent. The distinction between the latter two is rather subtle, and can be made on the fact that codes split up information according to semantic borders, while ciphers operate on chunks of information independently of the linguistic interpretation. The technological evolution from handwritten messages on paper sent by courier to the communication of information through both local and worldwide communication networks and the storage and processing in a variety of computer systems certainly has increased the vulnerability of information to eavesdropping. Cryptology was the only solution that was able to make the leap from the closed world of generals and diplomats to worldwide commercial applications.

Apart from concealment or privacy protection, it is equally important that both the contents and the originator of the information are not modified. Both requirements are captured in the term *authentication*. An attacker who tries to modify contents or origin of information is called an active attacker. The fact that the relative importance of this threat has increased can be illustrated by the emergence of malicious software programs. The best known examples of this group are certainly the computer viruses [51]. Others include worms [306], Trojan horses, and logical bombs. Every effective solution will have to be based on a verification of the authenticity of the software when it is loaded on the hard disk and when it is loaded by the CPU. The latter application will require very high throughput of 100 Mbytes per second and even more. A second illustration is situated in the banking world. The authenticity of financial transactions is generally considered more important than the secrecy, as one successful fraud can result in a considerable benefit for the attacker. The problem here is not only the economical value of a single attack, but the fact that the trust in the system can be lost [117]. A third application that will become more and more important is the protection of the authenticity of pictures and moving images (e.g. videoconferencing). As one can expect that it will become feasible to "edit" moving pictures and make a person say and do things he or she never said or did, it is required that one can guarantee

the authenticity of moving images. This will impose even higher requirements on the throughput. Other applications where authentication is important are alarm systems, satellite control systems, distributed control systems, and systems for access control [88].

Authentication is the protection of the communicating parties against attacks of a third party. However, a different threat emerges when the two communicating parties are mutually distrustful and try to perform a *repudiation*. This means that sender or receiver will try to modify a message and/or deny to have sent or received a particular message. In paper documents, protection against this type of attack is offered by a handwritten signature. It is clear that in case of electronic messages, a simple name at the end of the message offers no protection at all. This is analogous to the fact that a photocopy of a document with a manual signature has no value, as one can always produce a bogus document with cut and paste operations. A typical example of this fraud is the electronic communication of orders to a stockbroker. The customer can place an order to buy shares of company X. If some days later the transaction turns out badly, he will try to deny his order. If on the other hand, the transaction is successful, the stockbroker might claim that he never received the order with the intention to keep the profit. In case of a dispute, a third party (a judge), has to take a decision. An elegant technical solution to this problem was offered by the concept of digital signature schemes based on trapdoor one-way functions [95]. It will be discussed in more detail in the next section.

1.3 The separation of privacy and authentication

1.3.1 Models for symmetric and asymmetric cipher systems

In this section it will be explained how the concepts of authentication and privacy, that were at first closely related, grew more and more apart. First a model has to be given for a cipher system. Hereto the model for a symmetric or conventional cipher system introduced by C. Shannon in 1949 [303] will be extended to include asymmetric or public-key cipher systems (figure 1.1). The sender Alice wants to transmit the plaintext P to the receiver. She will transform the plaintext P into the ciphertext C with the encryption algorithm E. The encryption algorithm E is actually a family of transformations parameterized by an encryption key K^E. The receiver Bob will recover the plaintext P by applying the decryption algorithm D. This algorithm is in the same way parameterized by a decryption key K^D. The model also contains a key generation algorithm KG, that produces corresponding pairs K^E and K^D. For simplicity it will be assumed that the generation of keys is controlled by Bob. The key K^E has to be sent to Alice through a secure channel. The eavesdropper Eve, who also will be called cryptanalyst or opponent, knows the description of E and D, and has access to C. She will try to derive information on the plaintext P.

In case of a symmetric cipher, K^E and K^D are equal and the channel to distribute K^E has to protect both its privacy and its authenticity. A possible but expensive solution is to have the key communicated by a courier. The security of the cipher

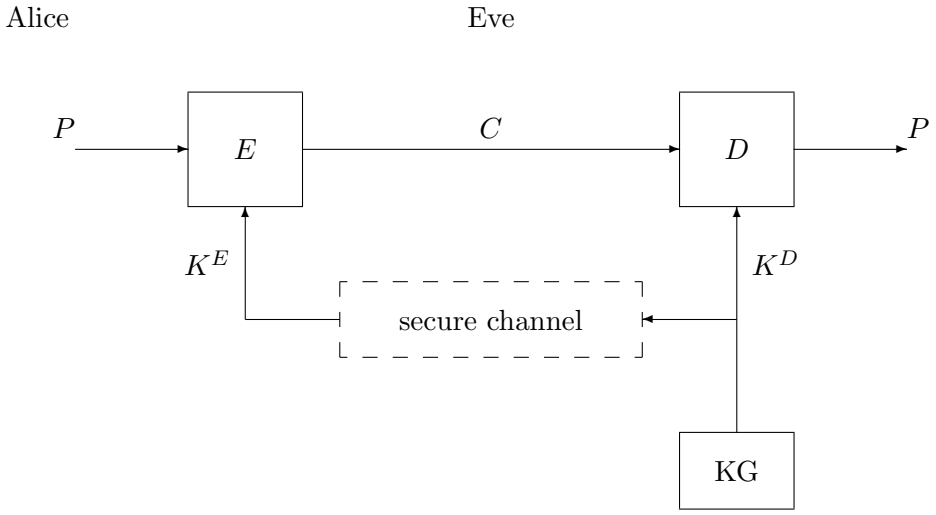

Figure 1.1: Model of cipher system.

relies on the fact that knowledge of E, D, and C does not allow to derive P.

In case of an asymmetric cipher, K^E is made public, and therefore this type of cipher is also called public-key algorithm. The channel to distribute K^E only has to protect the authenticity, while Bob has to protect both the authenticity and the secrecy of K^D. The assumptions underlying this cipher are that knowledge of E, D, and C does not allow to derive P and that knowledge of K^E does not allow to derive K^D. The concept invented to achieve these properties is the trapdoor one-way permutation (in fact a trapdoor one-way function suffices). This is a permutation that is hard to invert unless one knows some secret trapdoor information.

A beautiful property of the public-key algorithms is that if the encryption function is a trapdoor one-way permutation, they can be turned easily into a digital signature scheme. If Bob transforms a plaintext P with his secret key K^D, he obtains the ciphertext C'. It is possible for Alice — in fact for everyone who knows the corresponding public key K^E — to encipher C' with K^E and to verify that P is obtained. Note that here the implicit assumption is made that P contains some verifiable redundancy, as will be discussed in the next section. Because Bob is the only person who knows K^D, it is clear that he is the only person who can possibly have generated C', and hence he can not deny to have sent P. If both Alice and Bob generate their own key pair and exchange their respective public keys, a superposition of both operations will guarantee both privacy and authentication (figure 1.2). Alice will decipher P with her secret key K_A^D, subsequently encipher the result with the public key K_B^E of Bob, and send the resulting ciphertext C'' to Bob. Bob can obtain the corresponding plaintext and verify its authenticity by deciphering C'' with his secret key K_B^D and subsequently encrypting the result with the public key K_A^E of Alice.

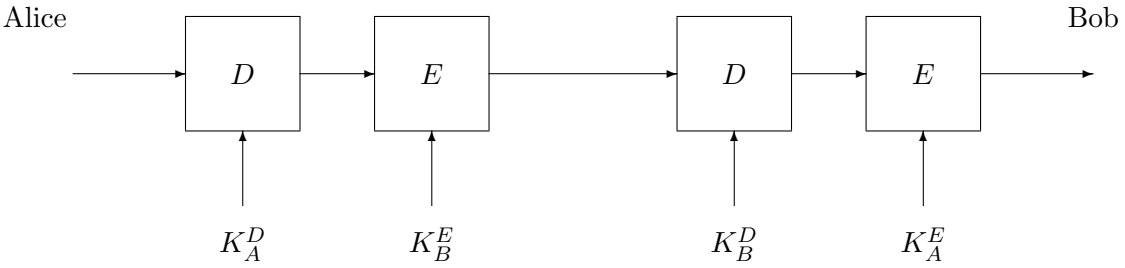

Figure 1.2: Protection of both authenticity and privacy with a public key system.

It is clear that the extension of this model to a model with central key generation and distribution is straightforward. In this case a hierarchical approach is possible based on master keys.

1.3.2 Information authentication and digital signatures

This section aims to illustrate briefly how cryptographic hash functions can be used to protect the authenticity of information and to improve digital signature schemes. A more detailed treatment will be given in chapter 2.

The protection of the authenticity of information includes two aspects:

- the protection of the originator of the information, or in ISO terminology [151] data origin authentication,
- the fact that the information has not been modified, or in ISO terminology [151] the integrity of the information.

There are two basic methods for protecting the authenticity of information.

- The first approach is analogous to the approach of a symmetric or asymmetric cipher, where the secrecy of large data quantities is based on the secrecy and authenticity of a short key. In this case the authentication of the information will also rely on the secrecy and authenticity of a key. To achieve this goal, the information is compressed to a quantity of fixed length, which is called a *hashcode*. Subsequently the hashcode is appended to the information. The function that performs this compression operation is called a *hash function*. The basic idea of the protection of the integrity is to *add redundancy* to the information. The presence of this redundancy allows the receiver to make the distinction between authentic information and bogus information.

  In order to guarantee the origin of the data, a secret key that can be associated to the origin has to intervene in the process. The secret key can be involved in the compression process or can be used to protect the hashcode and/or the

information. In the first case the hashcode is called a Message Authentication Code or MAC, while in the latter case the hashcode is called a Manipulation Detection Code or MDC.

- The second approach consists of basing the authenticity (both integrity and origin authentication) of the information on the authenticity of a Manipulation Detection Code or MDC. A typical example for this approach is a computer user who will calculate an MDC for all its important files. He can store this collection of MDC's on a floppy, that is locked in his safe, or he can write them down on a piece of paper. If he has to transfer the files to a remote friend, he can simply send the files and communicate the MDC's via telephone. The authenticity of the telephone channel is offered here by voice identification.

The second application of cryptographically secure hash functions is the optimization of digital signature schemes and the construction of digital signature schemes that are not based on a trapdoor one-way permutation (cf. section 2.3.3). The optimization is obtained through signing the MDC of a message instead of every bit or block. The description of the hash function can be public and it does not depend on any secret parameter. The advantages of this approach are that the signature has a fixed short length and that the computational requirements are minimized. In some cases the security level of the signature scheme can be increased. For some signature schemes the hash function is even an integral part of the scheme. Digital signature schemes based on one-way functions are in general less practical, but can be an alternative if one is not allowed or willing to use a scheme based on a trapdoor one-way permutation.

1.3.3 Privacy and authentication: two different concepts

Until recently, it was generally believed that encryption of information suffices to protect its authenticity. The argument was that if a ciphertext resulted after decryption in meaningful information, it should be originated with someone who knows the secret key, guaranteeing authenticity of message and sender. As a consequence, if an opponent wants to modify an enciphered message or to send a fake message, he has to know the secret key and hence to break the encryption algorithm. The opposite observation is clearly true: someone who has broken the encryption algorithm can easily modify messages or send bogus messages. One of the famous persons who experienced this was Mary, Queen of Scotland, who organized with A. Babington in 1586 a plot to assassinate Elisabeth. Her encrypted communications were deciphered by Phelippes. This enabled Phelippes to add a correctly enciphered postscript to one of Mary's letters asking for *"the names and the qualities of the six gentlemen which are to accomplish the designment"*, [174], pp. 122-123. Although the conspirators were caught before the answer could be intercepted, the forgery clearly illustrates the point.

Several counterexamples will show that it is not necessary to break the cipher system in order to be able to falsify messages. At first, the protection of integrity is strongly dependent on the *encryption algorithm* and on the *mode* in which the algorithm is used. The reader who is not familiar with the four standardized modes of a

cipher (Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB)) and with the difference between stream ciphers and block ciphers will find a detailed treatment of the modes and their properties in appendix A.

A famous cipher that offers unconditional secrecy is the Vernam cipher or modulo 2 one-time pad [322]. It was invented in 1917 by G. Vernam to encipher printed telegraph communications. The ciphertext is obtained through an addition modulo 2 or exor of the key. Its security relies on the fact that ciphertext and plaintext are statistically independent. The disadvantage of the system is that the size of the key is as large as the size of the plaintext. However, C. Shannon showed in 1949 [303] that this is optimal to achieve unconditional or perfect secrecy. Nevertheless an active attacker can change any bit of the plaintext by simply flipping the corresponding bit of the ciphertext, as was remarked by Feistel [104, 105]. This observation is also valid for any additive stream cipher and for the OFB mode of any block cipher. It holds partially if a block cipher is used in CFB or CBC mode [91].

If a plaintext longer than one block is enciphered with a block cipher in ECB mode, an active attacker can easily reorder the blocks. Another example is the vulnerability to active attacks of a plaintext encrypted in Cipher Feedback Mode (CFB). Due to the self-synchronization properties, any modification in the ciphertext will cause a corresponding modification to the plaintext and will subsequently garble the next part of the plaintext. When the error has shifted out of the feedback register, the ciphertext will be deciphered correctly again. If the last part of the ciphertext is modified, it is completely impossible to detect this. If the garbling occurs in the middle of the plaintext, it can only be detected based on redundancy, as will be discussed in the next paragraph.

In other modes (e.g. CBC) every ciphertext bit is a complex function of the previous plaintext bits and an initial value. If the modification of a single ciphertext bit results in t bits of the plaintext being garbled, the probability that the new plaintext will be accepted as meaningful equals 2^{-tD}, where D is the redundancy in the information. In case of natural language $D \approx 3.7$, and this probability is negligible for $t > 8$. However, if $D = 0$ all messages are meaningful, and encryption offers no authentication, independently of the encryption algorithm or of the mode. This means that an attacker can modify messages or forge messages of his choice. The limitation is that he does not know on beforehand what the corresponding plaintext will be, but many applications can be considered where such an attack would cause serious problems. Note that even if redundancy is present, a human checker or a designated computer program is required to check its presence. A detailed discussion of the use of different modes for authentication purposes will be given in chapter 5.

The second illustration of the independence of privacy and authentication is given by the use of *public-key algorithms*. From section 1.3.1 it is clear that two independent operations and two independent key pairs are necessary to protect both privacy and authenticity.

A third example is the use of a Message Authentication Code or MAC to protect

the authenticity, as discussed in section 1.3.2. A widely accepted and standardized way to compute a MAC is to encrypt the plaintext in CBC mode. The ciphertext corresponding to the last block depends on the secret key, on the initial value, and on all bits of the plaintext, hence it can be used as a MAC. In case the plaintext has to be encrypted, it is very appealing to use the same key for the encryption and for the calculation of the MAC, but a different initial value. However, it can be shown that this approach is insecure [167], and that a different key should be used for authentication and encryption purposes. This is discussed in more detail in chapter 5. A similar observation is made in [91] for an authentication scheme based on a stream cipher.

The last argument is an interesting series of toy examples with a two bit key and a one bit plaintext [202] illustrating that a cipher can offer either perfect secrecy, or perfect authenticity or both. The conclusion is that *"... secrecy and authenticity are independent attributes of a cryptographic system ..."*

1.4 Three approaches to the authentication problem

In present day cryptography, three approaches can be identified to solve most problems comprising information secrecy and information authenticity. These approaches differ in the assumptions about the capabilities of an opponent, in the definition of a cryptanalytic success, and in the notion of security. This taxonomy is based on the taxonomy that was developed for stream ciphers by R. Rueppel [287], and deviates from the taxonomy for authentication developed by G. Simmons [310].

A first method is based on information theory, and it offers unconditional security, i.e., security independent of the computing power of an adversary. The complexity theoretic approach starts from an abstract model for computation, and assumes that the opponent has limited computing power. The system based approach tries to produce practical solutions, and the security estimates are based on the best algorithm known to break the system and on realistic estimates of the necessary computing power or dedicated hardware to carry out the algorithm. In [310] the second and third approach are lumped together as computationally secure, and in [287] a fourth approach is considered, in which the opponent has to solve a problem with a large size (namely examining a huge publicly accessible random string); it can be considered as both computationally secure and information theoretically secure.

The properties of the three approaches are compared in table 1.1. It should be noted that the information theoretic approach is impractical for most applications because of the size of the secret key. Sometimes the complexity theoretic approach allows for efficient constructions, but in a practical scheme the dimensions of the scheme are fixed and the proof of security has only a limited value.

1.4.1 Information theoretic approach

This approach results in a characterization of unconditionally secure solutions, which implies that the security of the system is independent of the computing power of the

	computing power of opponent	security	practicality
information theoretic	unlimited	provable (unconditional)	impractical
complexity theoretic	polynomial	asymptotic (assumptions)	impractical
system based	fixed	no proof	efficient

Table 1.1: Comparison of the three approaches in cryptography.

opponent. E.g., in case of privacy protection, it has been shown by C. Shannon that unconditional privacy protection requires that the entropy of the key is lower bounded by the entropy of the plaintext. It should be remarked that both unconditional privacy and unconditional authenticity are only probabilistic: even if the system is optimal with respect to some definition, the opponent has always a non-zero probability to cheat. However, this probability can be made exponentially small. The advantage of this approach lies in the unconditional security. Like in the case of the Vernam scheme, the price paid for this is that these schemes are rather impractical.

The cryptographer considers a game-theoretic model, in which the opponent observes l messages and subsequently tries to impersonate or substitute messages. The cryptographer will encipher his messages under control of a secret key. Because the goal of the cryptographer is now the protection of the authenticity (or the combination of secrecy and authenticity), the transformation will be called an *authentication code*. The information theoretic study of authentication has now been reduced to the design of authentication codes, that are in some sense dual to error correcting codes [310]. In both cases redundant information is introduced: in case of error correcting codes the purpose of this redundancy is to allow the receiver to reconstruct the actual message from the received codeword, and to facilitate this the most likely alterations are in some metric close to the original codeword; in case of authentication codes, the goal of the redundancy is to allow the receiver to detect substitutions or impersonations by an active eavesdropper, and this is obtained by spreading altered or substituted messages as uniformly as possible.

The advantage of this approach lies in the unconditional security. Like in case of the Vernam scheme, the price paid for this is that these schemes are rather impractical.

1.4.2 Complexity theoretic approach

The approach taken here is to define at first a model of computation, like a Turing machine [5] or a Boolean circuit [98]. All computations in this model are parameterized by a security parameter, and only algorithms or circuits that require asymptotically polynomial time and space in terms of the size of the input are considered feasible. The next step is then to design cryptographic systems that are provably secure with respect to this model. This research program has been initiated in 1982 by A. Yao [331, 332] and tries to base cryptographic primitives on general assumptions. Exam-

ples of *cryptographic primitives* are: secure message sending, cryptographically secure pseudo-random generation, general zero-knowledge interactive proofs, Universal One-Way Hash Functions (UOWHF), Collision Resistant Hash Functions (CRHF), and digital signatures. It will be shown that the latter three are relevant for information authentication. Examples of *general assumptions* to which these primitives can be reduced are the existence of one-way functions, injections, or permutations, and the existence of trapdoor one-way permutations. A third aspect is the *efficiency of the reduction*, i.e., the number of executions of the basic function to achieve a cryptographic primitive, and the number of interactions between the players in the protocol.

Several lines of research have been followed. A first goal is to reduce cryptographic primitives to weaker assumptions, with as final goal to prove that the reduction is best possible. A different approach is to produce statements about the impossibility of basing a cryptographic primitive on a certain assumption [150]. One can also try to improve the efficiency of a reduction, possibly at the cost of a stronger assumption. If someone wants to build a concrete implementation, he will have to choose a particular one-way function, permutation, etc. The properties of a particular problem that is believed to be hard can be used to increase the efficiency of the solutions. Examples of problems that have been intensively used are the factoring of a product of two large primes, the discrete logarithm problem modulo a prime and modulo a composite that is the product of two large primes, and the quadratic residuosity problem.

The complexity theoretic approach has several advantages:

1. It results in *provable secure* systems, based on a number of assumptions.
2. The constructions of such proofs requires *formal definitions* of the cryptographic primitives and of the security of a cryptographic primitive.
3. The *assumptions* on which the security of the systems is based are also defined formally.

The disadvantage is that the complexity theoretic approach has only a limited impact on practical implementations, due to limitations that are inherently present in the models.

1. In complexity theory, a number of operations that is *polynomial* in the size of the input is considered to be feasible, while a superpolynomial or exponential number of operations in the size of the input is infeasible. In an asymptotic setting, abstraction is made from both constant factors and the degrees of the polynomials. This implies that this approach gives no information on the security of concrete instances (a practical problem has a finite size). Secondly, the scheme might be impractical because the number of operations to be carried out is polynomial in the size of the input but impractically large.
2. The complexity theoretic approach yields only results on the *worst case or average case* problems in a general class of problems. However, cryptographers studying the security of a scheme are more interested in the subset of problems that is easy.

3. Complexity usually deals with *single isolated instances* of a problem. A cryptanalyst often has a large collection of statistically related problems to solve.

It is interesting to remark that the starting point of this approach was the *informal* definition of a one-way function and a trapdoor one-way permutation in the seminal paper of W. Diffie and M. Hellman [95]. The first practical public-key cryptosystems were based on the hardness of factoring the product of two large primes (the RSA system proposed by R. Rivest, A. Shamir and L. Adleman [278]) and on the subset sum or knapsack problem (proposed by R. Merkle and M. Hellman [210]). However, it is not possible to show that the security of these systems is equivalent to solving the underlying hard problem. The best illustration of this fact is the fate of the knapsack public-key cryptosystems, that are almost completely broken [32, 92]. Although no one has been able to show that the security of RSA public-key cryptosystem is equivalent to factoring, no attack on the RSA scheme has been proposed that is more efficient than factoring the modulus. Historically, the attempts to prove the security of RSA resulted in the construction of new schemes for which it was possible to prove that breaking the scheme is equivalent to factoring. The next step was to design systems based on other assumptions and finally to generalize these assumptions.

1.4.3 System based or practical approach

In this approach schemes with fixed dimensions are designed and studied, paying special attention to the efficiency of software and hardware implementations. The objective of this approach is to make sure that breaking a cryptosystem is a difficult problem for the cryptanalyst.

By trial and error procedures, several *cryptanalytic principles* have emerged, and it is the goal of the designer to avoid attacks based on these principles. Typical examples are statistical attacks and meet in the middle attacks. An overview of these principles for cryptographic hash functions will be given in chapter 2.

The second aspect is to design *building blocks with provable properties*. These building blocks are not only useful for cryptographic hash functions, but also for the design of block ciphers and stream ciphers. Typical examples are statistical criteria, diffusion and confusion, correlation, and non-linearity criteria. The study of these criteria leads to the study of fundamental properties of Boolean functions in chapter 8.

Thirdly, *the assembly of basic building blocks* to design a cryptographic hash functions can be based on theorems. Results of this type are often formulated and proven in a complexity theoretic setting, but can easily be adopted for a more practical definition of "hardness" that is useful in a system based approach. A typical example is the theorem discovered independently in [66] and [213], stating that a collision-resistant hash function can always be constructed if a collision-resistant function exists, where the first reference uses a complexity theoretic approach and the second a more practical definition. A similar observation holds for the theorem that hash functions can be parallelized efficiently under certain assumptions, where a complexity theoretic approach was proposed in [66] and a practical approach independently by the author in

[252]. But even if the results are formulated directly in a practical approach, interesting results can be produced. A nice example is the design of a collision resistant hash function [213] based on the assumption that the underlying block cipher is random.

1.5 Outline of the thesis

In this thesis cryptographic hash functions will be studied according to the three approaches that have been presented in the previous section.

Chapter 2 contains an overview of cryptographic hash functions. This comprises practical definitions of the basic concepts, a discussion of applications, a general model for constructions of hash functions, and an extensive taxonomy of attacks on hash functions.

Chapter 3 contains a brief overview of the information theoretic approach to authentication, with the emphasis on the practicality of the constructions for authentication without secrecy.

In chapter 4 the complexity theoretic approach is presented. The uniform notation and definitions to describe the different schemes will make the complexity theoretic results more accessible to non-experts in this domain.

Our research concentrated on the practical approach. First an overview of three types of practical schemes will be given, and subsequently some design criteria for basic building blocks are discussed. In chapter 5, which is the most extensive chapter, cryptographic hash functions based on block ciphers are discussed. This comprises a brief introduction explaining the origin of the concept of an MDC, and subsequently the discussion of a large variety of schemes. Also three new schemes are proposed in this chapter.

In chapter 6 a survey of cryptographic hash functions based on modular arithmetic is given.

In chapter 7 proposals for dedicated hash functions are reviewed, and design criteria for these hash functions are discussed.

In chapter 8 the study of basic building blocks of cryptographic hash functions leads to the study of Boolean functions. Here new criteria are defined, and functions satisfying these criteria are studied. Extensions of these criteria to S-boxes will also be mentioned.

Chapter 9 contains the conclusions from this thesis and discusses some open problems.

1.6 Main contributions

One of the main contributions of this thesis is that it is the first text that gives an extensive treatment of cryptographic hash functions according to the information theoretic, complexity theoretic, and system based approach. The emphasis in our research lies on the more practical schemes, and in the interaction between theoretical and

practical constructions. In chapter 5 it is explained how the concept of cryptographic hash functions has developed from the redundancy that is added before encryption.

The contributions of chapter 2 are the establishment and discussion of a general model for hash functions. Also a taxonomy for attacks on hash functions is presented that can be used as a checklist for the evaluation of a hash function.

In chapter 3 that deals with the information theoretic approach, the efficiency of schemes that provide authentication without secrecy is compared for practical parameters.

In chapter 5, 6, and 7 all practical proposals published in the literature are described and evaluated. For hash functions based on block ciphers a synthetic approach was developed for the case where the size of the hashcode equals the block length of the underlying block cipher. This approach is applicable to the study of both MDC's and MAC's and can be extended to hash functions based on modular exponentiation. Three new hash functions based on a block cipher are proposed. In these chapters twelve new attacks on published hash functions are discussed, and for many other schemes an improved evaluation is presented. For some schemes this evaluation is based on an expression for the number of operations to obtain a multiple collision, that is derived in appendix B.

In chapter 5 and appendix C the technique of differential cryptanalysis is extended to hash functions based on block ciphers. In the same chapter the treatment of the addition schemes is generalized, as well as the treatment of the interaction between MAC calculation and encryption. In chapter 7 design criteria for dedicated hash functions are discussed.

The main contribution of chapter 8 is the definition of a new criterion for Boolean functions that generalizes existing criteria. Also, properties of quadratic functions are characterized through connections with matrix and graph theory. It proved useful to determine the number of graphs with a given minimum degree (appendix D). Functions that satisfy new and existing criteria are studied. Moreover this chapter proposes a new construction for bent functions and an efficient way to count the number of bent functions of 6 variables (which was previously unknown). Finally a new class of functions is presented that has some interesting properties.

Chapter 2

Cryptographic Hash Functions

To know that we know what we know, and that we do not know what we do not know, that is true knowledge. *Henry David Thoreau*

2.1 Introduction

Hash functions are functions that compress an input of arbitrary length to a result with a fixed length. If hash functions satisfy additional requirements, they are a very powerful tool in the design of techniques to protect the authenticity of information. In a first section cryptographic hash functions will be informally defined, and subsequently it will be discussed how hash functions can be used in a variety of applications. Next a general model will be presented together with a brief discussion of some specific constructions. Finally an extensive taxonomy is given of methods of attack on hash functions. These attacks are mainly relevant for the system based approach, that will be followed in chapters 5 to 8. However, the models and attacks presented in this chapter make it easier to understand the unconditionally secure and complexity theoretic constructions that are presented in chapter 3 and 4 respectively.

Many articles have discussed general aspects and applications of hash functions and the definitions have grown organically. Although a specific reference has been made wherever possible, the following valuable references on hash functions could not be indicated at a specific location: [6, 39, 64, 70, 71, 74, 86, 128, 129, 136, 166, 167, 168, 169, 170, 183, 211, 216, 220, 223, 230, 233, 250, 264, 280, 295, 335]. The most important contributions of this chapter are the establishment and discussion of a general model and the taxonomy for attacks. Part of the results in this chapter have been published in [253, 254, 256, 261].

2.2 Practical definitions

In the first chapter, it was explained how the authenticity of information can be verified through the protection of the secrecy and/or the authenticity of a short imprint or hashcode. In this section informal definitions will be given for a hash function that uses a secret key (Message Authentication Code or **MAC**) and for a hash function that does not make use of a secret key (Manipulation Detection Code or **MDC**). This last category can be split in two classes, depending on the requirements: one-way hash functions (OWHF) or weak one-way hash functions and collision resistant hash functions (CRHF), collision free hash functions, or strong one-way hash functions.

A brief discussion of the existing terminology can avoid confusion that is found in the literature. The term *hash functions* originates historically from computer science, where it denotes a function that compresses a string of arbitrary input to a string of fixed length. Hash functions are used to allocate as uniformly as possible storage for the records of a file. The name hash functions has also been widely adopted for *cryptographic hash functions* or cryptographically strong compression functions, but the result of the hash function has been given a wide variety of names in the cryptographic literature: hashcode, hash total, hash result, imprint, (cryptographic) checksum, compression, compressed encoding, seal, authenticator, authentication tag, fingerprint, test key, condensation, Message Integrity Code (MIC), message digest, etc. The terms MAC and MDC originated from US standards and are certainly not perfect (a MAC or an MDC are actually no codes, and both can serve for message authentication), but the adoption of these terms offers a practical solution to the momentary "Babel of tongues". One example of the confusion is that "checksum" is associated with the well known Cyclic Redundancy Checks (CRC) that are of no use for cryptographic applications. In this thesis the names MAC and MDC will also be used for the hashcode obtained with a MAC and an MDC respectively. Sometimes a MAC is called a keyed hash function, but then one has to use for an MDC the artificial term unkeyed or keyless hash function. According to their properties, the class of MDC's will be further divided into one-way hash functions (OWHF) and collision resistant hash functions (CRHF). The term collision resistant hash function (CRHF) is preferable over strong one-way hash function, as it explains more clearly the actual property that is satisfied. The term collision free hash function proposed by I. Damgård is also more explicit, but can be slightly misleading: in fact collisions do exist, but it should be hard to find them. An alternative that was proposed in [339, 341] is collision intractible hash functions. The term weak one-way hash function was proposed by R. Merkle in [213], in order to stress the difference with a strong or collision resistant hash function. Finally note that in a complexity theoretic context the term universal one-way hash function (UOWHF) was proposed by M. Naor and M. Yung in [233]. The main characteristic of this one-way hash function is that it is randomly selected from a large set and independently of the data to be hashed. This implies trivially that producing collisions for a single hash function is useless. To avoid confusion between this very specific definition and the more general one-way hash function, this term will only be used in the complexity theoretic approach. The relation between the different

hash functions can be summarized in figure 2.1.

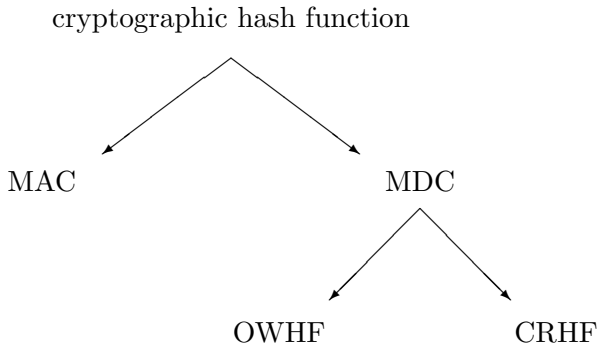

Figure 2.1: A taxonomy for cryptographic hash functions.

In the following the hash function will be denoted with h, and its argument, i.e., the information to be protected with X. The image of X under the hash function h will be denoted with $h(X)$ and the secret key with K.

2.2.1 One-way hash function (OWHF)

The first informal definition of a OWHF was apparently given by R. Merkle [211, 213] and M. Rabin [274].

Definition 2.1 *A* **one-way hash function** *is a function h satisfying the following conditions:*

1. *The description of h must be* publicly known *and should not require any secret information for its operation (extension of Kerckhoffs's principle*[1]*).*
2. *The argument X can be of* arbitrary *length and the result $h(X)$ has a* fixed *length of n bits (with $n \geq 64$, cf. section 2.5.1).*
3. *Given h and X, the computation of $h(X)$ must be* "easy"*.*
4. *The hash function must be* one-way *in the sense that given a Y in the image of h, it is "hard" to find a message X such that $h(X) = Y$ and given X and $h(X)$ it is "hard" to find a message $X' \neq X$ such that $h(X') = h(X)$.*

The first part of the last condition corresponds to the intuitive concept of one-wayness, namely that it is "hard" to find a preimage of a given value in the range. In the case of permutations or injective functions only this concept is relevant. The second part of this condition, namely that finding a second preimage should be hard, is a stronger condition, that is relevant for most applications. Formal definitions of a OWHF can be obtained through insertion of a formal definition of "hard" and "easy" in combination

[1] The Dutchman A. Kerckhoffs (1853–1903) was the first to enunciate the principle that the security of a cipher must reside entirely in the secret key.

with the introduction of a security parameter. In the complexity theoretic approach (cf. chapter 4) this means that the number of operations is superpolynomial in the size of the input. For a practical definition, one still has several options. In the case of "ideal security", introduced by X. Lai and J. Massey [183], producing a (second) preimage requires 2^n operations operations. However, it may be that an attack requires a number of operations that is smaller than $O(2^n)$, but is still computationally infeasible.

2.2.2 Collision resistant hash function (CRHF)

The first formal definition of a CRHF was apparently given by I. Damgård [64, 65] and will be discussed in chapter 4. An informal definition was given by R. Merkle in [213].

Definition 2.2 *A* **collision resistant hash function** *is a function h satisfying the following conditions:*

1. *The description of h must be* publicly known *and should not require any secret information for its operation (extension of Kerckhoffs's principle).*
2. *The argument X can be of* arbitrary *length and the result $h(X)$ has a* fixed *length of n bits (with $n \geq 128$, cf. section 2.5.1).*
3. *Given h and X, the computation of $h(X)$ must be* "easy"*.*
4. *The hash function must be* one-way *in the sense that given a Y in the image of h, it is "hard" to find a message X such that $h(X) = Y$ and given X and $h(X)$ it is "hard" to find a message $X' \neq X$ such that $h(X') = h(X)$.*
5. *The hash function must be* collision resistant: *this means that it is "hard" to find two distinct messages that hash to the same result.*

In section 4.3.3 a result will be given that suggests that in certain cases the first part of the one-way property follows from the collision resistant property. Similarly formal definitions of a CRHF can be obtained through insertion of a formal definition of "hard" and "easy" in combination with the introduction of a security parameter. For a practical definition, several options are available. In the case of "ideal security" [183], producing a (second) preimage requires 2^n operations and producing a collision requires $O(2^{n/2})$ operations. This can explain why both conditions have been stated separately. One can however also consider the case where producing a (second) preimage and a collision requires at least $O(2^{n/2})$ operations, and finally the case where one or both attacks require less than $O(2^{n/2})$ operations, but the number of operations is still computationally infeasible (e.g., if a larger value of n is selected).

2.2.3 Message Authentication Code (MAC)

Message Authentication Codes have been used for a long time in the banking community and are thus older than the open research in cryptology that started in the mid seventies. However, MAC's with good cryptographic properties were only introduced after the start of open cryptologic research.

Definition 2.3 *A* **MAC** *is a function satisfying the following conditions:*

1. *The description of h must be* publicly known *and the only secret information lies in the key (extension of Kerckhoffs's principle).*
2. *The argument X can be of* arbitrary *length and the result $h(K, X)$ has a* fixed *length of n bits (with $n \geq 32 \ldots 64$, cf. section 2.5.1).*
3. *Given h, X and K, the computation of $h(K, X)$ must be* "easy"*.*
4. *Given h and X, it is "hard" to determine $h(K, X)$ with a probability of success "significantly higher" than $1/2^n$. Even when a large set of pairs $\{X_i, h(K, X_i)\}$ is known, where the X_i have been selected by the opponent, it is "hard" to determine the key K or to compute $h(K, X')$ for any $X' \neq X_i$. This last attack is called an adaptive chosen text attack* [2]*.*

Note that this last property implies that the MAC should be both one-way and collision resistant for someone who does not know the secret key K. This definition leaves open the problem whether or not a MAC should be one-way or collision resistant for someone who knows K. In the next section, some applications will be discussed where this property could be useful, and in chapter 5 some new MAC's based on a block cipher will be proposed that satisfy one or both properties.

2.3 Applications of hash functions

In chapter 1 it has been explained that cryptographic hash functions can be used to protect information authenticity and to protect against the threat of repudiation. In this section both applications will be discussed in detail, together with the more specific problem of authentication for messages with more than one recipient. It will also be discussed how one can use hash functions for identification with passwords, and how one can derive an encryption algorithm from a hash function. The example of protection of a computer program against viruses will be used to illustrate the different concepts.

2.3.1 Information authentication

The basic idea of the use of cryptographic hash functions is to reduce the protection of the authenticity of information of arbitrary length to the protection of the secrecy and/or authenticity of quantities of fixed length. First, a distinction will be made between protection of authentication with and without secrecy. The second option is whether the protection of authenticity will depend on the secrecy and authenticity of a key or on the authenticity of an information dependent hashcode.

2.3.1.1 Authentication without secrecy

In this case there is only a plaintext available, which significantly reduces the number of options.

[2] For a more extensive discussion of attacks on a MAC the reader is referred to section 2.5.

MAC The simplest approach is certainly the use of a Message Authentication Code or MAC. In order to protect the authenticity of the information, one computes the MAC and appends this to the information. The authenticity of the information now depends on the secrecy and authenticity of the secret key and can be protected and verified by anyone who is privy to this key. Essentially the protection of authenticity has been reduced to the problem of secure key management. This scheme can only protect against outsider attacks, as all parties involved have the same capabilities and hence should trust each other.

The scheme can be made even more secure but less practical if also the authenticity and/or secrecy of the MAC of every plaintext is protected. A possible implementation could consist of an exchange of messages via a high speed communication link, while the corresponding MAC's are sent via a slower channel, that protects authenticity and/or secrecy. A simple authentic channel can be a telephone line (with voice recognition) or the conventional mail system (with manual signatures). The advantage is that it becomes impossible for any of the parties that know the secret key to modify an existing message and the corresponding MAC.

An issue of discussion is whether it should be "hard" for someone who knows the secret key to construct two arguments with the same MAC for that key. This will strongly depend on the application: in general an internal procedure has to be established to resolve disputes. A third party can make no distinction between the parties involved, but it is possible that, although both parties have the same capabilities, a certain asymmetry exists in their relation, e.g., the bank versus a customer. This procedure should specify what happens if someone denies a messages or subsequently claims he has sent a different message. If some additional protection against insider attacks is obtained from protection of the MAC, this property should be satisfied. However, also physical solutions based on tamper resistant devices can offer such protection. Nevertheless, a better way to solve disputes is to provide for non-repudiation through digital signatures.

MDC The alternative for a MAC is the use of an MDC. In this case the authenticity of the information is transferred to the authenticity of a string of fixed length. The advantage over a MAC is that there is no need for key management. In exchange for this, an authentic channel has to be provided for every MDC. This means that the capacity of the channel will increase with the number of messages. Although the life time of a key is also related to the number of times it has been used, it is clear that the authentic channel for the MDC will need a significantly greater capacity than the channel that protects both authenticity and privacy of the secret key for a MAC.

Just as in case of a MAC, the parties that use this approach are supposed to trust each other, but it is important to consider what will happen if a dispute arises, or what will happen if an insider will attack the system. An insider will try to find a collision, i.e., two plaintexts X and X' such that $h(X) = h(X')$. Subsequently he will protect the authenticity of X through $h(X)$, but at any time later he will be able to substitute X' for X. In order to avoid this attack, h should be a CRHF.

However, one can certainly imagine applications where this attack is not relevant. In that case one only has to be protected against outsiders, hence it suffices that h is a OWHF: an outsider can not select X, but will only be able to observe X and $h(X)$ and subsequently try to come up with an X' such that $h(X) = h(X')$.

1. The parties involved completely trust each other, which is trivially the case if there is only one party. One could think of someone who protects the integrity of his computer files through the calculation of an MDC that he stores in printed form in this vault. Every day he can repeat the calculation and verify the result.
2. The computation of the $h(X)$ involves a random component, that can not be controlled by the insider [213]: X can be randomized before applying h through encryption of X with a good block cipher using a truly random key, that is added to the resulting ciphertext [211], or through the selection of a short random prefix to X [64]; h itself can be randomized through randomly choosing h from a family of functions indexed by a certain parameter.

The advantage of a OWHF is that its design is easier and that storage for the hash-code can be halved (64 bits instead of 128 bits). The price paid for this is a degrading of the security level proportional to the number of applications of h: an outsider who knows a set $\{h(X) \mid X \in \text{Domain}(h)\}$ of size s has increased his probability to find an X' with a factor s. This limitation can be overcome through the use of a parameterized OWHF.

2.3.1.2 Authentication with secrecy

If both authentication and secrecy are protected, this can be used in certain cases to simplify the overall system. For insider attacks, the additional encryption makes no difference, as an insider knows the secret key for the encryption. This means that for certain applications it should be hard to find collisions. For an outsider, an attack on the scheme becomes in general harder, as his knowledge decreases.

Although it is tempting to use this fact to lower the requirements imposed on the hash functions, this is certainly not a good practice. The additional protection offered by the encryption is dependent on the encryption algorithm and on the mode of the encryption algorithm (cf. appendix A). Examples of this can be found in chapter 5.

MAC Several options can be considered, but all share the problem of a double key management: one for the authentication and one for the encryption. It is tempting to use the same key twice, but this has to be discouraged strongly: not only are there dangerous interactions possible between the encryption scheme and the MAC (these are extensively discussed in section 5.4.1), but the management of both keys should be different (e.g. lifetime, storage after use). The advantage of this approach is a high security level, owing to the complete separation of protection of privacy and authentication.

The most straightforward option is to calculate the MAC, append it to the information and subsequently encrypt the new message. An alternative is to omit the

encryption of the MAC. The third solution is to calculate the MAC on the enciphered message. The advantage is that the authenticity can be verified without knowing the plaintext or the secret key of the encryption algorithm, but in general it is preferable to protect the authenticity of the plaintext instead of the authenticity of the ciphertext.

MDC The advantages for using an MDC are a simplified key management and the fact that the authentication is derived directly from the privacy protection. The key management will be simplified because only one key will be necessary to protect both privacy and authenticity. The fact that the authentication is based on the privacy protection implies that it requires no additional secret key or authentic channel. In the context of the ISO Open System Interconnect Reference Model [151] integrity and confidentiality can be protected at different layers. No secret information would be necessary at the layer that calculates the MDC. The disadvantage is that the protection of authenticity depends on the privacy protection: if the encryption algorithm is weak, the protection of authenticity will also be jeopardized.

The most straightforward option is to calculate the MDC, append it to the information, and subsequently encrypt the new message. An alternative is to omit the encryption of the MDC. This approach seems to be more vulnerable to outsider attacks, but it should cause no problem if the MDC satisfies the imposed conditions. The third solution is to calculate the MDC on the enciphered message. However, this approach can not be recommended: the result has to be protected now with an authentic channel, and an important advantage of this approach is lost.

A special warning should be added in case the encryption algorithm is an additive stream cipher where the opponent knows the plaintext [167, 169]: in that case he can easily compute the key-stream sequence. Subsequently he can modify the plaintext, calculate the MDC, and encrypt both the new plaintext and the new MDC. This attack depends only on the mode of operation of the encryption and is independent of the choice of the MDC. A solution suggested by R. Jueneman is to let the MDC depend on a random initial value IV (cf. section 2.4.1) that is added to the plaintext, which means that it can not be known by an adversary. This is equivalent to using a MAC and adding the key for the MAC to the plaintext. In a communication environment this could be realized in practice by making the MDC in a message dependent on the previous message and on the previous MDC. The MDC in the first message should be based on a secret random component that was sent by the *receiver* when the session was established. The last message should contain and end-of-session symbol, the MDC of the previous message, and the MDC of the current message. This approach is limited to systems where sessions are established in real-time.

2.3.2 Authentication of multi-destination messages

In some applications a user Alice wants to send an authenticated message X to more than one receiver. To reduce communication bandwidth, only a single message is sent (e.g., in electronic mail applications this would allow that the message is only replicated where it needs to be). To simplify the treatment, it will be assumed that

there are only two intended receivers, Bob and Cecile. The results can be generalized easily to more receivers. It is assumed that Alice and Bob share a secret key K_{AB}, and Alice and Cecile share a secret key K_{AC}. Bob and Cecile both trust Alice, but Cecile will try to cheat Alice and Bob by sending a message X' to Bob and claiming that it was sent by Alice. This problem together with some solutions were considered in [218]. In this context it seems natural to use digital signatures based on public-key techniques, as will be discussed in the next section, but in some cases this is excluded for performance reasons or because of legal restrictions. The use of a secret key for every group of intended receivers is not secure (group members can cheat each other) and it is not practical (the management of group keys is complex). Three solutions will be discussed based on a MAC, and it will be shown that the requirements for the MAC can be different.

- The most straightforward solution is that Alice calculates a MAC of the message X using the secret key K_{AB} which will be denoted with $\mathrm{MAC}(K_{AB}, X)$. Similarly she computes $\mathrm{MAC}(K_{AC}, X)$ and appends both MAC values to the message. Subsequently she sends the complete message to Bob and Cecile. Both can easily verify the authenticity of X, and Cecile can not cheat because she does not know K_{AB}. The disadvantage of this scheme is that Alice has to compute twice a MAC.

- To reduce the computational load, a scheme was proposed where the MAC is computed only once with a single session key K_S [192]: $\mathrm{MAC}(K_S, X)$. This key is sent together with the message under encrypted form such that K_S can be retrieved by both Bob and Cecile. The MAC is encrypted under K_{AB} and under K_{AC} and appended to the message. The complete message will have the following form:

  $$E(K_{AB}, K_S), E(K_{AC}, K_S), E(K_{AB}, \mathrm{MAC}(K_S, X)), E(K_{AC}, \mathrm{MAC}(K_S, X)), X\,.$$

  In this case, Cecile will be able to obtain both K_S and $\mathrm{MAC}(K_S, X)$. If Cecile can find an X' such that $\mathrm{MAC}(K_S, X') = \mathrm{MAC}(K_S, X)$, she will be able to replace X by X' and convince Bob that Alice has sent to him X'. Note that if the MAC intended for Bob was not encrypted under K_{AB}, it would be trivial for Cecile to replace X by X' and to update the MAC accordingly. If Cecile can select a message that has to be sent by Alice, it is even sufficient that she can produce a pair X, X' such that $\mathrm{MAC}(K_S, X') = \mathrm{MAC}(K_S, X)$, for the given key K_S. If K_S is generated randomly before the MAC calculation, Cecile will not know this key, and the equality will have to hold for a reasonably large subset of the key space in order to make the attack work. If on the other hand K_S is predictable, the MAC should be collision resistant for someone who knows the key.

  The first way of cheating can be thwarted if the MAC is one-way in both senses as defined in definition 2.1. This was also remarked in [218], but it was only stated there that finding a preimage should be hard. The fact that the MAC

proposed in [192] was not one-way allowed C. Mitchell to break the scheme [221]. To thwart the second attack, Alice should randomize every message that she wants to authenticate, or the MAC should be collision resistant as stated above. The solution for this case that is proposed in [218] is to use a CRHF and to encrypt the result with K_S.

- In order to extend the lifetime of the shared secret keys, one can also generate a different session key for every intended receiver of the message: in this way the requirements to be imposed on the MAC are less stringent (an attacker does not know the value of the MAC), but the computational overhead for MAC calculation will again be higher [218].

The results in this section can easily be generalized to the case where both secrecy and authenticity have to be protected.

2.3.3 Non-repudiation

The technical term *non-repudiation of origin* [151] denotes a service whereby the recipient is given guarantee of the message's authenticity, in the sense that the recipient can subsequently prove to a third party that the message is authentic even if its originator subsequently revokes it. The need for a *"purely digital, unforgeable, message dependent signature"* has been identified by W. Diffie and M. Hellman in their 1976 paper [95]. In the same paper the authors propose an elegant solution based on trapdoor one-way permutations. The first practical proposal of a public-key system with digital signature capability as suggested by the title of the original paper was the RSA cryptosystem [278]. Its security is based on the fact that it is "easy" to find two large primes, but "hard" to factor their product. Subsequently new schemes appeared, based on the other number theoretic problems like the discrete log problem [100]. The complexity theoretic approach has resulted in provably secure digital signature schemes based on claw-free pairs of permutations [128, 129], one-way permutations [233], and finally on one-way functions [284], which can be shown to be optimal. A remarkable evolution here is the intertwining in some schemes between the signature and the hashing operation. The 1988 CCITT X.400 and X.500 recommendations [44, 45] (cf. also [219]) and in particular CCITT X.509 [46] are an example of recent standards that offer security facilities that are based on digital signatures. Digital signature schemes based on the practical approach were further optimized but have not received too much attention. Some of these schemes will be discussed briefly.

However, it is not within the scope of this thesis to give an extensive treatment of digital signature schemes. For such an overview the reader can consult [223]. We will limit ourselves to the interaction between these schemes and hash functions: first it will be discussed how digital signature schemes can be optimized using hash functions, and subsequently it will be shown how practical signature schemes can be constructed based on one-way functions.

2.3.3.1 Optimization of digital signature schemes

The basic idea to speed up all digital signature schemes is to compress the information to be signed with an MDC to a string of fixed length. Instead of signing the information one signs the hashcode, that is also called the "imprint" in standards terminology. In order to verify a signature, the outcome of the verification process of the signature is compared to the hashcode that can be calculated from the information. The advantages of this approach are the following:

1. The size of the signature can be reduced from the size of the information to one block length, independent of the length of the signed information. If no use is made of an MDC, it is of course possible to store only the signed information, and to compute the information from the signature whenever it is needed, but this solution involves a significant computational overhead.

2. The sign and verify function of most known signature schemes are several orders of magnitude slower in hardware and software than symmetric encryption functions, MAC's or MDC's. An example of highly optimized assembly code for a 16 MHz IBM PS/2 Model 80: the verification operation for 512-bit RSA takes about 330 milliseconds (1,550 bit/sec), the signing operation with a small public exponent a few milliseconds (100 Kbit/sec). This should be compared to an encryption speed for the block cipher DES [8, 108] of 97 μseconds (660 Kbit/sec) and for the hash function MD4 (cf. chapter 7) of 6.3 Mbit/sec.

3. If no MDC is used, and the information to be signed is longer than one block, it is easy to manipulate these individual blocks. The simplest example is a reordering of blocks.

4. The algebraic structure of the message space can be destroyed. In the case of RSA the message space has a multiplicative structure, i.e., the signature of the product of two messages equals the product of their signatures. A generalization is described in [102]. A similar property holds for the ElGamal signature scheme [223]. Examples of how this algebraic structure can be exploited in a protocol are described in [133, 232].

5. The reblocking problem can be avoided. This problem occurs when both privacy and authentication are protected with the RSA public-key cryptosystem. If the sender first encrypts the message with the public key of the receiver (to protect privacy), the result might be larger than his modulus. Before he can apply his secret key (to protect authenticity) he has to reblock the message. It is not too hard to overcome the reblocking problem: a simple solution is to let the order of the operations depend on the size of the two moduli. However, it is preferable to have no reblocking problem at all.

6. The signature protocol will not be useful for an opponent trying to obtain the plaintext corresponding to encrypted messages. This can only happen if one uses

the same key and digital signature scheme for privacy protection or authentication.

One has to be careful with the selection of the hash function, as an unfortunate interaction between the digital signature scheme and the hash function can result in specific vulnerabilities (cf. section 2.5.3).

Again the problem arises to determine the properties that have to be satisfied by the MDC. To protect against an attack of an outsider, if suffices that the MDC is a OWHF. He is not able to select the messages that he will attack, and hence he has to come up with a new message with the same hashcode as one of these messages. It is of no help for him to have a set of collisions that are randomly distributed over the message space. For an insider the situation is completely different: if he is able to find two messages say X and X' with the same hashcode, he can sign X and at a later stage claim that he has signed X'. Note that the insider is not necessarily the owner of the secret key: it is sufficient that he can construct messages that this person is willing to sign. In order to thwart this type of attack it is clear that the MDC should be a CRHF. One can argue that a OWHF would be sufficient if a randomization occurs in the signature process (X is randomized or h is randomly selected). The problem with this randomization is that it should not be under control of the insider.

Because this type of attack can only be performed by an insider, namely the person who produces the messages to be signed, it is not possible to repeat it too often: other users would not trust him any more. Nevertheless, digital signature schemes are used in cases where a repudiation could cause problems, and hence it is not acceptable that even one repudiation is possible. The only exception is an environment where digital signatures are used for message authentication solely for the purpose of information authentication: they could be preferred over a MAC because of the flexibility of the verification, i.e., they can be verified by anyone that knows the correct public key.

The hashcode can also be generated with a MAC, which implies that only persons privy to this secret key can verify the signature. The disadvantage of this approach is the management of an additional secret key: for digital signatures it is sufficient to guarantee the authenticity of the public key of the signer. One can however indicate some advantages with respect to outsider and insider attacks. For an outsider his knowledge of the system decreases. Breaking the scheme will require a chosen text attack on the system with the secret key for the MAC in place, and hence no parallel attacks are possible. A similar observation holds for an insider who can prepare messages to be signed, but who does not know the secret key. For an insider who knows the secret key it should be hard to construct collisions, which means that the requirements on the MAC are very strong.

For the signature schemes based on zero knowledge techniques, e.g., Fiat-Shamir [107], Guillou-Quisquater [266] and Schnorr [296], the hash function forms an integral part of the signature process. First a random number is generated, that is transformed with a one-way function to obtain y. The hash function is then applied to the concatenation of y and the message. The hashcode is subsequently transformed with the secret signature key. The signature then consists of the outcome of this transformation

together with y. From the discussion it follows that it is sufficient for the hash function to be a OWHF: an attacker can not completely control its input.

Note that if short messages are to be signed, the only argument to use a hash function is the destruction of the algebraic structure of the message space. An alternative to a hash function is the addition of well chosen redundancy to the information. An elegant proposal to avoid multiplicative attacks against the RSA digital signature scheme can be found in [137, 155]. Other solutions and their analysis can be found in [79, 80].

2.3.3.2 Practical digital signatures based on a one-way function

One-way functions are closely related to one-way hash functions; the only difference is that the input now has to be of fixed length. Although several definitions exist for one-way functions, it is sufficient here to modify the second condition in the definition of a OWHF and a CRHF to obtain the definition of a one-way function (OWF) and of a collision resistant function (CRF). Again formal definitions can be obtained through insertion of a formal definition of "hard" and "easy" in combination with the introduction of a security parameter.

Definition 2.4 *A* **one-way function (OWF)** *is a function g satisfying the following conditions:*

1. *The description of g must be* publicly known *and should not require any secret information for its operation (extension of Kerckhoffs's principle).*
2. *The argument X has a* fixed *length of m bits and the result $g(X)$ has a* fixed *length of n bits (with $n \geq 64$, cf. section 2.5.1).*
3. *Given g and X, the computation of $g(X)$ must be* "easy"*.*
4. *The function must be* one-way *in the sense that given a Y in the image of g, it is "hard" to find a message X such that $g(X) = Y$ and given X and $g(X)$ it is "hard" to find a message $X' \neq X$ such that $g(X') = g(X)$.*

Definition 2.5 *A* **collision resistant function (CRF)** *is a function g satisfying the following conditions:*

1. *The description of g must be* publicly known *and should not require any secret information for its operation (extension of Kerckhoffs's principle).*
2. *The argument X has a* fixed *length of m bits and the result $g(X)$ has a* fixed *length of n bits (with $n \geq 128$ cf. section 2.5.1).*
3. *Given g and X, the computation of $g(X)$ must be* "easy"*.*
4. *The function must be* one-way *in the sense that given a Y in the image of g, it is "hard" to find a message X such that $g(X) = Y$ and given X and $g(X)$ it is "hard" to find a message $X' \neq X$ such that $g(X') = g(X)$.*
5. *The function must be* collision resistant*: this means that it is "hard" to find two distinct messages that yield the same result.*

Several digital signatures have been proposed based on the practical definition of a one-way function. The choice between a CRF and a OWF is based on the same arguments: if the attacker is able to select X, the function should be a CRF.

The first scheme is the Diffie-Lamport one time signature [95]. In order to sign a single bit message, Alice randomly selects a pair $x_1, x_2 \in \mathrm{Dom}(g)$ and computes $y_1 = g(x_1)$ and $y_2 = g(x_2)$. Next she puts y_1, y_2 in an authenticated public file. If Alice wants to sign the message, she reveals x_1 if the message bit equals 0 and x_2 if the message bit equals 1. Bob can subsequently easily verify that $y_i = g(x_i)$. A signature scheme for M possible k-bit messages requires for every user pair a public storage of $2Mkn$ bits. If $M = 2^{20}$ and $k = n = 128$ then this amounts to 4 Gigabytes. The signer needs $2k$ applications of g for every message, while the verifier needs only k applications of g. Note that if k is large, one can always apply a CRHF to reduce its size to about 128 bits and sign subsequently this hashcode.

A first improvement can be obtained through coding of the message. The basic observation is that, if one forgets about x_1 and y_1, it becomes easy for Bob to change a 1 into a 0, by simply denying that he ever received x_2. If a k-bit message has to be signed, this can be solved by appending the count of the 0 bits to the message, resulting in an improvement with a factor

$$\frac{2}{1 + \frac{\log_2 k}{k}},$$

as only $k^* = k + \log_2 k$ $(x_i, g(x_i))$ pairs are necessary. An additional improvement was proposed by R. Winternitz and is described by R. Merkle in [212]. It reduces the size of the signature at the expense of an increased computation. If y_i is obtained from 2^α repeated applications of g to x_i, every y_i an be used to sign an α-bit message, hence the size of the signature can be reduced with a factor of α. If both optimizations are implemented, signer and verifier need about $k2^{\alpha-1}$ applications of g, while the storage requirements are reduced to Mkn/α bits. For the same example and $\alpha = 6$ the public storage capacity will now be 360 Megabytes. The main limitations of these schemes is that they are one time signatures, i.e., the authenticated public quantities can be used only once. This is reflected by the fact that the public storage grows linearly with M.

The tree authentication scheme proposed by R. Merkle [212] results in a significant reduction of the storage requirements. Without going into the details, one can mention the following conclusions. The public storage will be only k bits. The size of the signature grows logarithmically with M and is equal to $2k^*n \log_2 M$. The signing operation will require on average about $3k^* \log_2 M$ applications of g (this comprises the use of g to generate pseudo-random variables instead of using truly random quantities) and a memory of size $(\log_2 M)^2 n/2$. The verification operation requires about $k^* \log_2 M$ applications of g. For the example this means only 3.1 Kbytes for the computations and 8100 applications of g. The length of a signature is 84.4 Kbytes. The author indicates that further optimizations are possible, that reduce the number of applications of g with a factor 4. Note that the logarithmic dependence on the number of messages to be signed M implies that this number can be increased significantly without requiring excessive storage.

The advantage of these schemes is that they can be based on any one-way function, and that one has an informal proof of their security. The disadvantage is that the public storage is rather large compared to the 64 bytes for a public key for RSA (in the case of a small public exponent) and that it depends on the number of messages that will be signed. The number of operations is comparable: the speed of the fastest one-way functions (e.g., functions based on DES or derived from existing hash functions) is about three orders of magnitude faster than the speed of a modular exponentiation (cf. section 2.3.3.1). An overview of the size of the parameters for 4 signature schemes (Rabin, Diffie-Lamport, Winternitz and Merkle) is given in table 2.1. The Rabin scheme [274] is a scheme with probabilistic verification depending on a security parameter t: the probability that an invalid signature is accepted by the verifier is approximately $\sqrt{\pi t}/2^{2t}$, which means that t should be $\approx n/2$ to obtain a similar security level.

	# operations		size	
	signing	verification	public storage	signature
Rabin ($k = n$)	$2t$	t	$4Mtn$	$2tn$
Diffie-Lamport	k	k	Mk^*n	k^*n
Winternitz	$k2^{\alpha-1}$	$k2^{\alpha-1}$	Mk^*n/α	k^*n/α
Merkle	$3k^* \log_2 M$	$k^* \log_2 M$	$(\log_2 M)^2 n/2$	$2 \log_2 Mk^*n$

Table 2.1: Parameters for 4 signature schemes based on a one-way function.

2.3.4 Identification with passwords

An MDC can be used to generate from a password or a passphrase a publicly accessible (readable) password file: one stores the MDC corresponding to the password or passphrase, instead of the password itself. Subsequently one has to protect only the integrity of the password file. In most applications it should be infeasible to derive a valid password from an entry in the file, which implies that a OWHF is sufficient. This is in fact one of the few cases were only finding a first preimage should be hard. Historically this is probably the first application of one-way functions. If a passphrase of arbitrary size has to be compressed, one will need a one-way hash function. A related application is commitment to a string without revealing it.

2.3.5 Encryption algorithms based on hash functions

Without going into details, one can remark that every hash function can be used in several ways to produce an encryption algorithm. A first possibility is to use the hash function as the F-function in a Feistel cipher [104, 105]. The text input of the round can be used as the chaining variable of the hash function (cf. section 2.4.1), and the key can go to the data input. An alternative is to have a fixed chaining variable and to concatenate the data to the key. Interesting theoretical results can be shown if the

hash function is pseudo-random and if the round keys are independent (e.g. [340]). A second possibility is to construct a key stream generator with a mode where the output of the hash function is fed back to the input, or where the input is derived from a counter. In case of a MAC the design can even be simpler, as the use of a secret key is already part of the algorithm. It is certainly possible to design more efficient encryption algorithms from scratch, but this type of solutions could be acceptable for applications where encryption is required occasionally.

2.3.6 Application to software protection

To illustrate the use of a MAC, MDC, and a digital signature scheme, it will be explained how these three techniques can be applied to protect the integrity of software [216]. The two parties involved in the application are the software vendor (who is also supposed to be the author of the software) and the user. The attacker will try to modify the software: this can be a computer virus, a competitor or even one of the parties involved. For this application there is clearly no need for secrecy. The three approaches will be discussed together with their advantages and disadvantages.

MAC: the software vendor will use his secret key to compute the MAC for the program and append the MAC to the program. The main problem here is the distribution of the secret key to the user through a channel that protects both its secrecy and its authenticity, which induces a significant overhead. This secret key has to be protected carefully by both software vendor and user: if a compromise at one place occurs, the protection is lost. Both software vendor and user can modify the program and the corresponding MAC, and thus in case of a dispute, a third party can not make any distinction between them. The vulnerability of the secret key implies that it is mandatory that every user shares a different key with the software vendor. The advantage of this approach is that the secret storage is independent of the number of programs to be protected, but depends on the number of users (for the software vendor) and on the number of different software vendors (for the user).

MDC: the software vendor will compute the MDC for the program. The main problem here is the distribution of the MDC to the user through a channel that protects the authenticity of the MDC. This is easier than the distribution of a secret key, but for every update of the program or for every new program a new transmission of an MDC is necessary. If the authenticity of the MDC is compromised, the protection is lost: the software vendor, the user, and any third party can modify the program and the corresponding MDC. If a dispute occurs, one has to show to a judge that the value of an MDC is authentic: it is generally not possible to prove to the judge who actually modified the authentic channel and the program. The main advantage is that this approach requires no secret storage. Every program needs an authentic storage both at the user's site and at the vendor's site.

Digital signature: the software vendor will append to the program a digital signature that is computed with his secret key. The main problem here is the distribution of the corresponding public key to the user through a channel that protects its authenticity. The secrecy and authenticity of the secret key have to be protected carefully by the software vendor: if it is compromised, anyone can modify programs and update the corresponding signature. If the authenticity of the public key is compromised, the protection is also lost: anyone can replace it with the public key corresponding to his secret key. The difference with the previous approaches is the asymmetry: only the software vendor can generate a signature, but anyone can verify it. This implies that the vendor can be held liable to a third party if the program contains a virus. The only way he can escape is by claiming that his secret key was stolen. He can however not repeat this type of fraud, as he will loose quickly the trust of his customers. Every vendor has to store one secret key, while every user needs an authentic storage for the public key of every vendor.

The selection of a particular solution will depend on the one hand on the number of users, vendors and programs, and on the other hand on the availability of authentic and/or secret storage and communication. The digital signature is clearly the only solution that can protect the users against a malicious software vendor.

A similar verification process can be executed when the program is loaded from disk to the Central Processing Unit. If the disk is not shared, non-repudiation is not required, but it is still attractive to use a digital signature scheme: the CPU has to know only the public key corresponding to the disk. An alternative is to provide for authentic storage of the MDC of a file that contains the MDC's of all programs. In [69] a scheme is described that combines a digital signature for checking new software with a MAC for verification at run-time.

2.4 General constructions

In this section a general model will be established for a hash function. Based on this model the relation will be discussed between the properties of the hash function and the properties of its building block, the "round function". Subsequently a tree construction for hash functions is described and cascading of hash functions is discussed.

2.4.1 General model

The general model for describing a hash function will be sketched. All known hash functions are based on a compression function with fixed size input; they process every message block in a similar way. This has been called an "iterated" hash function in [183].

The information is divided into t b-bit blocks X_1 through X_t. If the total number of bits is no multiple of the block length b, a padding procedure has to be specified.

A number of examples of padding rules with increasing strength are given, where the roles of 0's and 1's can always be interchanged.

- The simplest padding rule is to complete the information with 0's. This padding rule is ambiguous as it not clear how many trailing 0's are part of the information. However this can be acceptable if the length of the information is known in advance or if it is included in the information.
- Padding of the information on the right with a 1 followed, if necessary, by 0's until it becomes complete. If the last block is complete, a supplementary block is added to the information, equal to a 1 followed by 0's.
- Padding of the information with z 0's except for the last r bits; these contain the r-bit binary representation of z. If no r bits are left in the incomplete last block, one or more blocks have to be added.
- Padding of the information with z 0's except for the last r bits; these contain the length of the information in bits. If no r bits are left in the incomplete last block, one or more blocks have to be added. If this padding rule is applied, no message can be obtained from another message by deleting the first blocks.

The choice between these different rules depends on the application, but it will become clear that the last one offers a larger security level, and is therefore strongly recommended.

The hash function h with compression function or round function f can then be defined as follows:

$$\begin{aligned} H_0 &= IV \\ H_i &= f(X_i, H_{i-1}) \qquad i = 1, 2, \ldots t \\ h(X) &= H_t \,. \end{aligned}$$

Here H_i are the intermediate variables or chaining variables that have a length of n bits, and the X_i are b-bit blocks. The result of the hash function is denoted with $h(X)$ and IV is the abbreviation for Initial Value. This corresponds to the description of a finite state machine with initial state IV, input X_i and next state function f. In some cases the input is subjected to preprocessing. This preprocessing stage introduces additional redundancy to increase the strength of the hash function. This redundancy consists of constant bits, a repetition of bits, or even a more complex procedure. The price paid for this redundancy is a decreased speed of the hash function.

The specification of a hash function requires the description of f, IV, a padding procedure, and optionally a preprocessing stage. If a hash function h is collision resistant, this implies that it should be hard to find an input pair X, X' such that $h(X) = h(X')$, for a given IV, padding procedure and preprocessing stage. The same holds for the one-way property. Several attacks on hash functions have been described that attack the hash function for a different IV. The following cases can be distinguished:

- If a collision pair can be found for an $IV' \neq IV$, this also creates suspicion on the security of the hash function, depending on the size and nature of the sets of IV's: if IV' is the outcome of a pseudo-random process, this is not so dangerous, as the probability of hitting the right IV is negligible.

- The situation is different for a CRHF if a pair X, X' is constructed that results in the same hashcode if IV and respectively IV' are used as initializing variables of the hash function. It is clear that this deviates significantly from the strict definition of a collision, and therefore we suggest the term "pseudo-collision". Most hash functions are not designed to meet this criterium. Of course it is a nice property if even finding such a pseudo-collision is hard, but it is only relevant for applications where IV can be modified by an attacker, or where the hash function is constructed in a different way (cf. section 2.4.4).

- A similar remark applies if a preimage can be found for an $IV' \neq IV$: for a large class of constructions it is always possible to find such a "pseudo-preimage". If the attack yields an IV' that is the outcome of a pseudo-random process, the probability of hitting the right IV is as small as finding a random preimage. However, finding a "pseudo-preimage" should not be too easy, as one can transform a pseudo-preimage into a preimage with a so called meet in the middle attack (cf. section 2.5.2.1).

One can conclude that every specification of a hash function should fix an initial value IV (or a small set of initial values), together with a motivation for the choice. If IV is generated pseudo-randomly, the algorithm should be described: a proposal is to take the all zero block or the hash value (with $IV = 0$) of a block consisting of all zeroes. If IV is not specified, it should be hard to produce collisions for any IV. In that case it is clearly necessary to add the length of the message at the end, in order to avoid trivial attacks, like omitting one or more blocks at the beginning of the message. If the first message block is derived from IV, say $X_1 = g(IV)$ (note that in some cases it is insecure to take the identity function), one can show that finding a pseudo-preimage is not easier than finding a preimage: an attacker can go backwards till the first stage, but he will not be able to come up with an IV such that $f(IV, g(IV)) = H_1$. Note that in the case of a digital signature it is easy to protect against pseudo-collisions: it is sufficient to sign IV together with the hashcode.

It has been indicated that adding the length of the message in the padding procedure can thwart trivial attacks in case IV is not specified. Moreover it can also protect against fixed point attacks (cf. infra) and against attacks that produce a second preimage for a long message in less than 2^n operations. The long message attack is basically due to R. Winternitz [328], and it was improved in [183]. Here a further generalization is given, that takes into account the required storage.

Proposition 2.1 *Given a t-block message X, a second preimage for $h(X)$ can be found in $2^n/r + r$ computations of the round function f and with a storage requirement*

of r n-bit quantities, under the restriction

$$1 \leq r \leq \min(t, 2^{n/2}) .$$

Proof: One calculates and stores the first r intermediate values H_i of the computation of $h(X)$. After computing $2^n/r$ values for $H'_1 = h(IV, X'_1)$ with randomly chosen X'_1, the probability for a match between H'_1 and some H_i, say for $i = i^*$ is approximately 63% (cf. appendix B). Then the message $X' = (X'_1, X_{i^*+1}, \ldots, X_t)$ hashes to the same value as the message X (note that the probability that $X' = X$ is negligible). It is clear from the expression for the number of computations that if $t > 2^{n/2}$ it does not make sense to choose r larger than $2^{n/2}$: this would imply that evaluation of the intermediate values of $h(X)$ requires more computations of the round function than finding a second preimage. ∎

If there are no storage limitations, r will be chosen equal to its maximal value. This generalization is motivated by the fact that in general 2^n computations is easier to achieve than storage (and sorting) of 2^n n-bit quantities. The main problem is the access time of the memory (cf. section 2.5.1.3). It is clear that this attack is not possible if the length of X is added as the last block.

A second problem with long messages is the "loss of memory" problem. It was posed by H. Block in an oral communication at Eurocrypt'84, and was mentioned by D. Davies in [72]. Assume that for a fixed X_i, f is a random injective mapping. If a large number of variations of the first blocks are chosen, all 2^n states will be reached at some point. However, if the next message blocks are kept constant, it can be shown that the fraction of states $y[i]$ at stage i is given by the recursion

$$y[i+1] = 1 - \exp(-y[i]) .$$

With $y[0] = 1$ (for convenience) one finds that

$$y[i] \approx \frac{2}{i + \frac{1}{3}\ln i + \frac{9}{5}} .$$

One can conclude that this is clearly an argument to let f be a bijection for fixed X_i. If this is not the case, it has to be checked to what extent this effect occurs (f might behave even worse than a random mapping). Moreover the message size has to be limited, or the size of the chaining variables has to be increased. Observe that this is only a problem if n is relatively small, which might happen in the case of a MAC.

As a conclusion one can state that one should either fix the initial value IV or add the total length of the message in the padding, and that it is strongly recommended to take both measures. Moreover, one should specify an upper bound on the size of the input.

In case of a MAC, it is a common mistake to choose the IV equal to the secret key. If this is the only way the key is involved in the MAC calculation, one can append an arbitrary number of blocks to the message and update the hashcode without knowledge of the secret key. Therefore the secret key should be involved both at the beginning

and at the end of the computations, but it is recommended that f depends on the secret key.

Research on hash functions has been focussed on the question: what conditions should be imposed on f to guarantee that h satisfies certain conditions ? This approach is based on the fact that one of the problems in assessing the security of a hash function is caused by the arbitrary size of the input. It is clear that weaknesses of f will generally result in weaknesses of h, but the converse does not hold in general. The main problem is to derive sufficient conditions on f. An important result that was achieved is that under certain conditions a CRHF or OWHF can be derived from a fixed size compression function that is collision resistant or one-way.

2.4.2 Conditions on the function f for a OWHF

The "composition theorem" by M. Naor and M. Yung [233] shows that a specific OWHF (namely a Universal One-Way Hash Function or UOWHF) can be constructed if a specific one-way function can be constructed that compresses a single bit. Their result will be discussed in detail in chapter 4.

Below four conditions are discussed that have to be verified to assess the security of a OWHF. These conditions have to be met to thwart certain attacks. It is assumed that the function f is not trivially weak, which means that it is linearly dependent or even not dependent on one of its inputs.

Direct Attack: the most straightforward attack is to simply invert f w.r.t. X_i. This can be thwarted by imposing the requirement that f is one-way w.r.t. X_i, or given H_{i-1}, H_i, (and X_i) it must be "hard" to find an $X_i' \neq X_i$ such that $f(X_i', H_{i-1}) = H_i$. The expression "for a given ..." implies that H_{i-1} and H_i can not be selected directly by an attacker. This attack can be used to construct either a preimage or a second preimage. If a direct attack requiring 2^s operations can be mounted such that only $n' < n$ bits of H_i are matched, finding a (second) preimage will require $2^{n-n'+s}$ operations, which is more efficient than exhaustive search if $s < n'$.

Forward Attack: an attacker can always replace X_j by X_j'. At a later stage, say iteration i with $i > j$, he will have to bring the two chains together again. This implies that given H_{i-1}, H_{i-1}', and X_i, it must be "hard" to find an X_i' such that $f(X_i', H_{i-1}') = f(X_i, H_{i-1}) = H_i$. This attack can only be used to construct a second preimage. If only n' bits can be matched in 2^s operations, one finds again that $2^{n-n'+s}$ operations will be necessary for a complete match.

Backward Attack: it should be hard for an attacker to go backwards through the chain, i.e., given H_i to produce a pair (X_i, H_{i-1}) such that $f(X_i, H_{i-1}) = H_i$. The motivation to forbid this is the existence of meet in the middle attacks (cf. section 2.5.2.1), that can help an attacker as follows [183]:

Proposition 2.2 *If it takes 2^s operations to go one step backwards in the chain, then a pseudo-preimage can be found in 2^s operations, and a (second) preimage for a fixed IV in $2^{1+\frac{n+s}{2}}$ operations.*

It is clear that if $s = n$ this attack does not yield any improvement. The following generalization can be made if only n' bits of H_i can be matched:

Proposition 2.3 *If it takes 2^s operations to go one step backwards in the chain, while providing a match for only $n' < n$ bits of H_i, a pseudo-preimage can be found in $2^{n-n'+s}$ operations, and a (second) preimage for a fixed IV in $2^{1+n+\frac{s-n'}{2}}$ operations.*

Proof: The number of operations for the preimage follows from the fact that the remaining $n - n'$ bits will be correct with a probability of $2^{-(n-n')}$.
For the (second) preimage attack one computes $H_1' = f(X_1, H_0)$ for $2^{n+\frac{s-n'}{2}}$ randomly chosen values of X_1. Then one constructs $2^{n-\frac{s+n'}{2}}$ pairs of (X_2, H_1'') for which $f(X_2, H_1'')$ agrees in n' positions with H_2. One expects that all n positions of H_2 will be correct for 1 value in $2^{n-n'}$, yielding $2^{\frac{-s+n'}{2}}$ useful pairs (X_2, H_1''). The attack succeeds if a match occurs between an H_1' and an H_1'' (cf. section 2.5.1.3). The probability that this happens is approximately equal to 63% (cf. appendix B). The number of operations for both phases is equal, which results in a total of $2^{1+n+\frac{s-n'}{2}}$ operations. ∎

Fixed Point Attack: a fixed point is a pair (X_i, H_{i-1}) such that $f(X_i, H_{i-1}) = H_{i-1}$. An extension of this concept is a fixed point of the r-fold iteration of f. If an attacker can easily find an X_i for a chosen H_{i-1}, it is a special case of a backward attack. On the other hand, if X_i can be chosen, and subsequently a corresponding value of H_{i-1} is obtained, or if the pair (X_i, H_{i-1}) is obtained from a random process, the value is much more limited: it can only be used to find a pseudo-preimage or a second preimage of a hashcode equal to H_{i-1}. If the values of H_{i-1} are uniformly distributed, this attack is not better than a random search for a preimage.

The first three conditions are clearly necessary, and the first two are a special case of the backward attack. They have been described because they will form the basis of a classification of hash functions based on block ciphers in section 5.3.1.4. In [183], it was shown that if the padding contains the length of X, and if the message (without padding) contains at least 2 blocks, ideal security of f against a backward attack is necessary and sufficient for ideal security of h with fixed IV (cf. section 2.2).

Proposition 2.4 *Assume that the padding contains the length of the input string, and that the message X (without padding) contains at least 2 blocks. Then finding a second preimage for h with a fixed IV requires 2^n operations if and only if finding a second preimage for f with arbitrarily chosen H_{i-1} requires 2^n operations.*

The proof is based on the fact that if IV/H_{i-1} can be chosen freely, the security of f is equivalent to the security of h.

2.4.3 Conditions on the function f for a CRHF

A construction for a CRHF based on a collision resistant function (**CRF**) was presented at Crypto'89 both by R. Merkle [212] and (independently) by I. Damgård [66]. R. Merkle called his construction the "meta method", and presented it together with a scheme based on DES (cf. section 5.3.2.2). I. Damgård [66] gave a more formal construction and proof that will be discussed in chapter 4. The idea of the construction is to select for f a CRF with input the concatenation of its arguments X_i and H_{i-1}. It can then be shown that the fact that f is a CRF implies that h is a CRHF. The proof will only work if the padding contains the length of the input string X.

The "meta method" takes an elegant approach, but might be too restrictive for practical applications. The converse of the theorem is certainly not true: it might be that f is not a collision resistant function, while h clearly is. The main observation is that the role of the information blocks X_i and the chaining variables H_i in the hashing process is essentially different: in the construction of a collision for h the attacker can completely control X_i, but H_i is either the fixed initial value or the image of X_i and H_{i-1} under f. Note that this is not true if IV can be chosen freely. The main problem to determine a set of conditions for f and to prove that these are necessary and sufficient for h to be CRHF, is that nothing can be said on the distribution of the H_i's: an attacker can not simply select them, but the designer can not show that the attacker can not control their distribution in a way that helps him to produce a collision.

Again four types of attacks can be identified:

Direct Attack: the most straightforward attack is to simply replace X_i by X_i'. This can be thwarted by imposing the requirement that f is collision resistant w.r.t. X, or for a given H_{i-1} it must be "hard" to find a pair (X_i, X_i') with $X_i \neq X_i'$ such that $f(X_i, H_{i-1}) = f(X_i', H_{i-1})$. The expression "for a given H_{i-1}" implies that H_{i-1} can not be selected directly by an attacker. If only n' bits of H_i can be matched, one obtains the following proposition:

Proposition 2.5 *If it takes 2^s operations for a direct attack, while providing a match for only $n' < n$ bits of H_i, a collision can be found in $2^{1+\frac{n-n'}{2}+s}$ operations.*

Forward Attack: an attacker can always replace X_j by X_j'. At a later stage, say iteration i with $i > j$, he will have to bring the two chains together again. This implies that for a given pair (H_{i-1}, H_{i-1}'), it must be "hard" to find a pair (X_i, X_i') such that $f(X_i, H_{i-1}) = f(X_i', H_{i-1}')$. This is not equivalent to imposing the condition that f should be collision resistant: this would be the case if the attacker is able to select (H_{i-1}, H_{i-1}'). It is clear that proposition 2.5 can be extended to this attack.

Backward Attack: it should be hard for an attacker to go backwards through the chain, i.e., for a given H_i, it must be "hard" to find a pair (X_i, H_{i-1}) such that $f(X_i, H_{i-1}) = H_i$. The motivation to forbid this is the existence of meet in the middle attacks (cf. section 2.5.2.1). These attacks are not generally applicable, as they can only yield a pseudo-collision. Nevertheless it should not be too easy to go backwards, as one requires from a CRHF that it is also one-way. Proposition 2.5 can be extended to this case if collision is replaced by pseudo-collision.

Fixed Point Attack: the definition of a fixed point is discussed in the previous section. If an attacker can easily find an X_i for a chosen H_{i-1}, it is a special case of a backward attack. On the other hand, if X_i can be chosen, and subsequently a corresponding value of H_{i-1} is obtained, or if the pair (X_i, H_{i-1}) is obtained from a random process, the applications are much more limited: it can only be used to find a collision for an $IV = H_{i-1}$. If the values of H_{i-1} are uniformly distributed, the probability that the exact IV will be matched is smaller than the probability for a random collision. Moreover this type of collisions can be avoided easily by adding the length at the end of the message.

The two first conditions are clearly necessary, and the third and fourth condition will thwart a certain class of attacks. We have for the time being no idea whether or not these conditions are sufficient to trust a practical hash function.

A possible solution for the problem concerning the distribution of the chaining variables could be to introduce the assumptions that f is a random mapping, which would imply that all intermediate values H_i will be random. The expression "for a given H_i" implies then that the chaining variables H_i are uniformly distributed and independently selected random variables. It is not obvious that in this case the four conditions are sufficient, because other attacks are applicable, that operate on more than one block at a time. The opponent can try to solve the following problem: find for a given H_{i-2} two different pairs (X_{i-1}, X_i) and (X'_{i-1}, X'_i) such that

$$f(X_i, f(X_{i-1}, H_{i-2})) = f(X'_i, f(X'_{i-1}, H_{i-2}))\,.$$

This type of attack can of course be extended to more steps. It seems that the randomness of the intermediate values makes this attack as difficult as directly attacking the hashcode with a birthday attack (cf. section 2.5.2.1), but no formal proof of this has been given.

The "meta method" also requires some trust: one has to trust that f is a collision resistant function. The main advantage is that one has to study only a function with a fixed size argument. For constructions where this method is not applicable, one has to trust that the function f satisfies less stringent requirements and that the conditions are sufficient to guarantee that h is a CRHF.

2.4.4 Tree approach to hash functions

The linear structure for a hash function can also be replaced with a tree structure for different reasons:

- to speed up the evaluation of h with a time-processor trade-off,
- to authenticate only a single leaf from the input: this requires knowledge of only $O(\log n)$ entries and only $O(\log n)$ computations instead of $O(n)$ computations to verify the whole tree [85, 211, 212].

In this case it should be hard to find a pseudo-preimage or a pseudo-collision for the round function f. The first argument was independently discovered by the author [252] and by I. Damgård [66], who also gave a formal proof of the correctness in case that the building function f is collision resistant. For universal hash functions (cf. section 3.3.2), a similar construction was suggested by M. Wegman and J. Carter in [325].

If one disposes of a suitable round function f that hashes m bits to n bits, the evaluation of h for a k-bit input can be done with $\frac{k}{2n}$ processors with

$$O\left(\frac{n}{m-n} \cdot log_2\left(\frac{k}{n}\right)\right)$$

evaluations of f. For the simple case where $k = 2^q$ for some integer q and $m = 2 \cdot n$, this results in

$$\begin{aligned} H_i^1 &= f(X_{2i-1}, X_{2i}) \qquad i = 1, \ldots, 2^{q-1} \\ H_i^j &= f(H_{2i-1}^{j-1}, H_{2i}^{j-1}) \quad i = 1, \ldots, 2^{q-j} \qquad (j = 2, \ldots, k-1) \\ H &= f(H_1^{k-1}, H_2^{k-1})\,. \end{aligned}$$

The time to compute the result is $O(\log k)$ instead of $O(k)$. A trade-off between chaining and tree approach allows for a speedup with about a factor c when c processors are available. Note that with this approach a different hash function is obtained for every value of the parameter k.

2.4.5 Cascading of hash functions

This section contains two simple observations concerning cascading hash functions, inspired by similar results for stream ciphers [205]. If $h_1()$ and $h_2()$ are hash functions, and $g()$ is a one-way function, then

- $h(X_1, X_2) = g(h_1(X_1) \parallel h_1(X_2))$ is a CRHF if $h_1()$ is CRHF and $g()$ is a CRF,
- $h(X_1) = g(h_1(X_1) \parallel h_2(X_1))$ is a CRHF if either $h_1()$ or $h_2()$ is a CRHF and $g()$ is a CRF,
- $h(X_1) = h_1(X_1) \parallel h_2(X_1)$ is a CRHF if either $h_1()$ or $h_2()$ is a CRHF.

Here $\parallel$ denotes the concatenation of two strings. The first part is equivalent to the tree construction. The second part constructs a CRHF that is at least as strong as the strongest of two collision resistant hash functions, provided a CRF is available. Note however that in both cases one could replace $g()$ by a OWF with a result of 128 bits. The resulting hash function will be a CRHF if finding collisions for $g()$ is not too easy.

For the third part the function $g()$ is omitted, but the hashcode will now be at least 256 bits long. The two last constructions increase the security level at the cost of a decreased performance. Note that this result can easily be extended to more than two functions.

2.5 Methods of attack on hash functions

The goal of this section is to give an overview of the known methods of attack on hash functions. The large number of possible attacks will be classified in five types:

1. attacks independent of the algorithm,
2. attacks dependent on the chaining,
3. attacks dependent on an interaction with the signature scheme,
4. attacks dependent on the underlying block cipher,
5. high level attacks.

Before treating these attacks in detail, the assumptions on the information available to an attacker will be discussed. In case of an MDC, all information is public, and the attacker simply faces the task to produce a preimage or a different element with the same image in case of a OWHF or a collision in case of a CRHF. One can make a distinction between the following cases, depending whether the value of IV is different from the specified value:

Preimage: here an attacker tries to find a preimage for a given hashcode.

Second preimage: here an attacker tries to find a second preimage for a given hashcode.

Pseudo-preimage: here an attacker tries to find a preimage for a given hashcode, with $IV' \neq IV$.

Second pseudo-preimage: here an attacker tries to find a second preimage for a given hashcode, with $IV' \neq IV$ (note that this case will not be discussed in the following).

Collision: here an attacker tries to find a collision.

Collision for different IV: here an attacker tries to find a collision for $IV' \neq IV$.

Pseudo-collision: here an attacker tries to find for some IV' and IV'' a pair X', X'', such that $h_{IV'}(X') = h_{IV''}(X'')$.

It is clear from the definitions that finding a pseudo-collision can be not harder than finding a pseudo-preimage, and that finding a collision can be not harder than finding a (second) preimage. A similar taxonomy was suggested in [183], but they make no distinction between second preimage and preimage. Their terminology for second (pseudo-)preimage is "(free-start) target attack", and the two last collision attacks are called "semi-free-start" respectively "free-start collision attack".

For a MAC the situation is more complicated. The proposed taxonomy is equivalent to the taxonomy of [129] for digital signature schemes. Depending on the information available to an attacker, the following types of attacks are distinguished:

Known plaintext attack: here an attacker is able to examine some plaintexts and their corresponding MAC.

Chosen plaintext attack: here an attacker is able to select a set of plaintexts, and subsequently he will obtain a list of MAC's corresponding to these plaintexts.

Adaptive chosen plaintext attack: this is the most general attack where an attacker will choose a plaintext and immediately receive the corresponding MAC: the choice of a plaintext can depend on the outcome of previous questions.

"Breaking" a MAC can have different meanings:

Total break: this means that an attacker can determine the secret key K.

Universal forgery: in this case an attacker can find an algorithm that is functionally equivalent to the MAC evaluation algorithm.

Selective forgery: here an attacker can determine the correct MAC for a particular plaintext chosen a priori by him.

Existential forgery: here an attacker can determine the MAC for at least one plaintext. As he has no control over this plaintext, it may be random or nonsensical.

The fourth requirement in definition 2.3 can now be restated as follows: it should be "hard" to perform an existential forgery with an adaptive chosen plaintext attack. Note that obtaining a MAC for a plaintext from the owner of the secret key is not considered as a forgery.

An evaluation of a scheme for message authentication or a digital signature strongly depends on the information at the disposal of an adversary, the actions he can undertake and finally on the consequences of both a successful and an unsuccessful attack. In general, a conservative approach is recommended. This implies that one assumes that a MAC will be considered to be broken if an attacker can commit an existential forgery based on an adaptive chosen message attack with the only restriction on the number of plaintexts coming from limited storage and computation capacities.

2.5.1 Attacks independent of the algorithm

This class of attacks depends only on the size of the hashcode n and the size of the secret key k (for a MAC), and is independent of the nature of the algorithm. It is assumed that the hashcode is a uniformly distributed and independent random variable: if this is not the case this class of attacks will be even more successful.

2.5.1.1 Random attack

The opponent selects a random message and hopes that the change will remain undetected. If the hash function has the required random behavior, his probability of success equals $1/2^n$ with n the number of bits of the hashcode. A major difference between a MAC and an MDC is that for a MAC the attack has to be carried out on-line, and hence the attack depends on two elements:

- The number of trials T that can be carried out, which depends on the speed of the implementation and on the time an attacker has access to the system with the key in place. The number of trials can be limited by undertaking a special action (e.g., perform manual verifications) if the number of erroneous results exceeds a certain threshold. An example where a large number of trials is possible, is the control of a satellite authenticated by a MAC.
- The expected value V of a successful attack. In wholesale banking this can be on the order of 100 million $ or even more.

The expected value of an attack equals then $T \cdot V/2^n$. If the number of trials can be limited, or if the expected value is limited like in retail banking, a value of $n = 32$ is sufficient. However, for most applications it is recommended that the size of the hashcode is at least 64 bits.

For an MDC the attack can be carried out off-line and in parallel. This means that the length of the hashcode n should be at least 64 bits. If a significant number of messages can be attacked at the same time, it is advisable to select a larger value of n. In section 2.4.1 it has been shown that finding a preimage for long messages is easier, unless the length of the message is included in the padding.

2.5.1.2 Exhaustive key search

An exhaustive search for a key is only applicable to a MAC. It is a known plaintext attack, where an attacker knows M plaintext-MAC pairs for a given key. He will precompute the MAC for every possible key in order to eliminate wrong keys. The size of the key space is equal to k bits, and the expected number of keys that remain will be denoted with K_{exp}. If M is sufficiently large, it is possible to determine the key uniquely or $K_{exp} \approx 1$. The relation between K_{exp}, M and n can be determined [236] under the assumption that the MAC is a random mapping, and that no key clustering occurs, i.e., that there are no equivalent keys. For the correct key, an attacker will perform M MAC calculations, while for a bad key the probability that exactly i trials are performed is equal to

$$\left(1 - \frac{1}{2^n}\right) 2^{-n(i-1)} .$$

The expected number of trials is given by the following expression:

$$\left(1 - \frac{1}{2^n}\right) \sum_{i=1}^{M} \frac{i}{2^{n(i-1)}} < \frac{1}{1 - 2^{-n}} .$$

The total number of trials to identify the key is upper bounded by

$$M + \frac{2^k - 1}{1 - 2^{-n}}\,, \tag{2.1}$$

and the number of keys that remains is expected to be

$$K_{exp} = 1 + \frac{2^k - 1}{2^{Mn}}\,. \tag{2.2}$$

This means that the number of plaintext-MAC pairs to determine the key uniquely is slightly larger than k/n. After the birthday attack it will be discussed how large k has to be in order to offer sufficient security for the next decades.

2.5.1.3 Birthday attack

The idea behind this attack is that for a group of 23 people the probability that at least two people have a common birthday exceeds 1/2 [106] . Because this number of people is significantly smaller than what one would expect, this has also been called the "birthday paradox". For some applications a related problem is relevant: if two groups of people have 17 persons each, the probability that two people in the two different groups have a common birthday will also exceed 1/2. Note that these results assume that birthdays are randomly distributed over the year; as this is not the case the probability will be even higher. This can be generalized as follows. If two samples of size r_1 and r_2 are drawn from a set of n elements, and if $r_1 r_2 = n$ with $r_1, r_2 = O(\sqrt{n})$, then the probability of a match equals $1 - 1/e$ or 63%. Note that if the attacker is unlucky, it is sufficient to increase the size of r_1 and r_2 slightly, which will increase the success probability significantly. If $r_1 + r_2$ has to be minimized, one can show that this corresponds to $r_1 = r_2 = \sqrt{n}$. This explains why attacks based on this property have also been called "square root" attacks. For a more detailed discussion of the probabilities the reader is referred to appendix B.

The first attack based on this property was proposed by G. Yuval [335]. He showed how to attack a digital signature scheme of Rabin [274], more in particular he shows that it is easier to construct collisions for a one-way function than to find a preimage of a given element in the range. A collision can be produced in the following way.

- The adversary generates r_1 variations on a bogus message and r_2 variations on a genuine message. This is very easy, even if r_1 and r_2 are large: it is sufficient to have $\log_2(r_1)$ respectively $\log_2(r_2)$ positions where one has two alternatives or synonyms. If $r_1 = r_2 = r = \sqrt{n}$ the probability of the existence of a match will be 63%. Note that in case of a MAC the opponent is unable to generate the MAC of a message. He could however obtain these MAC's with a chosen plaintext attack. A second possibility is that he collects a large number of messages and corresponding MAC's and divides them in two categories, which corresponds to a known plaintext attack.

- The search for a match does not require r^2 operations: after sorting the data, which requires $O(r \log r)$ operations, comparison is easy.

An algorithmic improvement has been the collision search algorithm proposed by J.-J. Quisquater [267, 270]. It is based on Pollard's ρ-method for finding cycles [300] in periodic functions on a finite domain. It eliminates almost completely the storage requirements if the attacker is able to call the function (it does not work if a match has to be found in stored data). If a MAC is attacked this corresponds to an adaptive chosen text attack. The basic idea is that if a random mapping is iterated (the output is fed back to the input), it will arrive after a tail with length λ into a cycle with period μ. At the point where the tail enters the cycle (the point of contact), one has found two values x and x' such that $f(x) = f(x')$. A graphical representation of this process will correspond to the Greek letter ρ. The storage requirements can be reduced to a negligible quantity by only storing points with specific characteristics (distinguished points). The expected number of function evaluations is equal to $\rho = \lambda+\mu = \sqrt{\pi n/8}+\sqrt{\pi n/8} = \sqrt{\pi n/2}$ (for a proof, see [113]). This result is also applicable to several other attacks: it is possible to produce pseudo-collisions for a single iteration step or collisions with initial values chosen from a small set. Other extensions will be discussed in the rest of this section.

Feasibility An important problem is to decide which computations should be considered feasible for the time being and within 10 and 20 years from now. This discussion is partially based on [273]. In terms of computations, one can start from the following facts (mid 1992):

- a single PC or workstation is able to perform a function evaluation in about 25 μsec, which corresponds to 2^{40} function evaluations per year,
- a supercomputer like the Cray-3 or the Cray Y-MP C90 (both with 16 processors) is capable of performing 16 Gigaflops on 64-bit words [348, 349]. If one function evaluation takes 64 operations, this corresponds to 2^{52} function evaluations per year.

Based on the observation that the speed of computers is multiplied by four every three years this means that 21 years from now (which seems a reasonable time scale for a number of applications) a single super computer will be able to perform 2^{66} function evaluations per year. It will require 4096 simple processors to perform the same number of operations. However, it can be expected that in the near future even inexpensive computers will have many processors built in, as increasing the number of processors is the most effective way to increase the computational power without excessive increase of the cost (economy of scale) [349]. Hence it is realistic to assume that this computing power will be available in any small size organization. It can be shown that many problems in cryptanalysis can be easily adapted to such a distributed environment [272]: it are probably the applications that will achieve the highest performance on massive parallel machines. These predictions can be extended to dedicated cryptanalytic hardware, if one accepts the assumption that hardware will remain about two

orders of magnitude faster. This corresponds to a factor of $2^6 \ldots 2^7$. The disadvantage of dedicated hardware is the higher cost.

For memory requirements, the situation is more complex, as the size of the available memory depends on the access time [265]. Moreover the access time to memory decreases much slower than the cycle time of the processor, and this can be solved only partially by using cache memories. An efficient attack will balance the use of different types of memories such that the access times are comparable to the calculations that have to be done in between. An example of such an attack using one central hardware board that is connected to a large number of PC's with a few Megabytes of memory has been described in [169]. Predictions can be based on the observation that memory devices increase in capacity by a factor of four every three years. Today's supercomputers have a main memory of up to 32 Gigabytes, a disk capacity of $50-100$ Gigabytes and a high-performance mass storage system of 200 Gigabytes [349]. For storage with still slower access, like tapes, capacity in the order of several Terabytes is currently available, and in the next decade this will become Pentabytes. The memory available in workstations is much smaller. Fast cache memories have currently a capacity between 256 and 512 Kbytes. For dynamic RAMs, 4 Mbit chips are currently in mass production, which means that main memory of 32 Megabytes is becoming the standard in workstations. For disk storage, 1 Gigabyte can be considered state of the art.

One can conclude that for attacks that require no storage, a size of 128 bits corresponding to 2^{64} operations is sufficient for the next 10 years, but it will be only marginally secure within 20 years. One can predict that a storage of about 64 Gigabytes with an acceptable access time will be available on a single workstation within 10 years. If one has 1024 machines of this type available, this amounts to 64 Terabytes. With this constraint, attacking a 64-bit hashcode requires only 2^{21} operations, but probably the access time to the memory would be dominant. For a 96-bit hashcode this amounts to 2^{54} operations corresponding to a few years on these machines, and to a few months if dedicated hardware is available for the computations. For a 128-bit hashcode this would require 2^{86} operations, which is probably not realistic for the next 20 years (in fact the storage capacity will be a factor 64 larger by then, which yields 2^{80} operations). It is clear that a hashcode of 160 bits offers a sufficient security level for 20 years or more.

2.5.2 Attacks dependent on the chaining

This class of attacks depends on some high level properties of the elementary function f.

2.5.2.1 Meet in the middle attack

This attack is a variation on the birthday attack, but instead of the hashcode, intermediate chaining variables are compared. The attack enables an opponent to construct a message with a prespecified hashcode, which is not possible in case of a simple birthday

attack. Hence it also applies to a OWHF. The opponent generates r_1 variations on the first part of a bogus message and r_2 variations on the last part. Starting from the initial value and going backwards from the hashcode, the probability for a matching intermediate variable is given by the same formula. The only restriction that applies to the meeting point is that it can not be the first or last value of the chaining variable. The probability to find a match as a function of r_1 and r_2 is described in appendix B. The cycle finding algorithm by J.-J. Quisquater can be extended to perform a meet in the middle attack with negligible storage [270, 273]. The attack can be thwarted by avoiding functions f that are invertible to the chaining variable H_{i-1} and to the message X_i (cf. section 2.4.2 and 2.4.3).

2.5.2.2 Constrained meet in the middle attack

This type of attack is based on the same principles as the meet in the middle attack, but it takes into account certain constraints that have to be imposed on the solution. Examples of restrictions are that the sum modulo 2 of all blocks should be constant, or that a block of the CBC encryption of the solution with a given initial value and key should take a prespecified value.

2.5.2.3 Generalized meet in the middle attack

This attack was extended [56, 123] to break the p-fold iterated schemes. In these schemes the message is repeated p times or p hash values are computed corresponding to p initial values. With the extended attack, breaking these schemes does not require $O(2^{\frac{pn}{2}})$ but only $O(10^p \cdot 2^{\frac{n}{2}})$ operations. The size of the message in this construction is $2 \cdot 10^{p-1}$ blocks. Modest trade-offs between time, storage, size of the message and processing are possible.

2.5.2.4 Correcting block attack

This attack consists of substituting all blocks of the message except for some block X_j. This block is then calculated such that the hashcode takes a certain value, which makes it also suitable to attack a OWHF. It often applies to the last block and is then called a correcting last block attack, but it can also apply to the first block or to some blocks in the middle. The hash functions based on modular arithmetic are especially sensitive to this attack.

A correcting block attack can also be used to produce a collision. One starts with two arbitrary messages X and X' and appends one or more correcting blocks denoted with Y and Y', such that the extended messages $X\|Y$ and $X'\|Y'$ have the same hashcode.

One can try to thwart a correcting block attack by adding redundancy to the message blocks, in such a way that it becomes computationally infeasible to find a correcting block with the necessary redundancy. The price paid for this solution is a degradation of the performance.

2.5.2.5 Fixed point attack

The idea of this attack is to look for a H_{i-1} and X_i such that $f(X_i, H_{i-1}) = H_{i-1}$. If the chaining variable is equal to H_{i-1}, it is possible to insert an arbitrary number of blocks equal to X_i without modifying the hashcode. Producing collisions or a second preimage with this attack is only possible if the chaining variable can be made equal to H_{i-1}: this is the case if IV can be chosen equal to a specific value, or if a large number of fixed points can be constructed (if e.g., one can find an X_i for every H_{i-1}). Of course this attack can be extended to fixed points that occur after a number of steps. This attack can be prevented easily: one can append a block count to the data or one can (for theoretical constructions) encode the data with a prefix-free code [66].

2.5.2.6 Key collisions

This type of attack can only be applied to hash functions based on block ciphers. If the chaining mode is poorly designed, attacks can be launched based on key collisions. A key collision is a pair of keys K_1, K_2 such that $E(K_1, P) = E(K_2, P)$ for a plaintext P. The number of collisions for a given plaintext can be obtained from theorem B.2. In the case of DES [8, 108], with a block length of 64 bits and a key size of 56 bits, the number of k-fold collisions for a given P is indicated in table 2.2. Key collisions can

k	2	3	4	5	6	7
	47.0	37.4	27.4	17.1	6.5	-4.3

Table 2.2: Binary logarithm of the expected number of k-fold key collisions for a given plaintext in the case of DES.

be constructed with an ordinary birthday attack, but J.-J. Quisquater has shown how the efficient cycle algorithms combined with the method of the distinguished points can produce a collision in about 2^{33} operations [267, 270] and with negligible storage. An important observation is that doubling the number of operations yields a squaring of the number of different collisions.

The attack can be extended to the case of double encryption [270]. In this case a key collision consists of two pairs of keys (K_1, K_2) and (K_1', K_2') (with $K_i \neq K_i'$) such that

$$E(K_2, E(K_1, P)) = E(K_2', E(K_1', P)) .$$

It is also possible to produce a single key pair such that $E(K_2, E(K_1, P)) = C$ for a given plaintext P and ciphertext C.

The collision search is feasible for any block cipher that behaves as a random mapping if the key size is significantly smaller than 128, but a good design of the hash function can make the collisions useless. There is however no easy way to guarantee this, and every scheme has to be verified for this attack (cf. chapter 5).

2.5.2.7 Differential attacks

Differential cryptanalysis is based on the study of the relation between input and output differences and is applicable to both block ciphers and hash functions [20, 21, 22, 23, 24, 25, 182]. The attack is statistical as one searches for input differences that are likely to cause a certain output difference. If one is looking for collisions this output difference should be equal to zero. In case of hash functions based on block ciphers, the situation is slightly different: depending on the mode one requires that the output difference is zero or that the output difference is equal to the input difference (in case of feedforward of the plaintext). It applies only to iterated ciphers that satisfy particular conditions, the so-called Markov ciphers. It turns out that most known iterated ciphers (DES [8, 108], FEAL [225, 228], LOKI [33, 34], PES or IDEA [181, 182, 184], etc.) are of this nature. For well designed block ciphers this attack will find the key based on a large number of plaintexts with a chosen difference, or an even larger number of known plaintexts. One can remark that this class of attacks is in fact more natural in case of an MDC, where there is no secret information. A chosen message attack is the standard way of attacking an MDC, and in this case all calculations can be performed off-line and in parallel.

2.5.2.8 Analytical weaknesses

Some schemes allow manipulations like insertion, deletion, permutation and substitutions of blocks. A large number of attacks have been based on a blocking of the diffusion of the data input: this means that changes have no effect or can be cancelled out easily in a next stage. This type of attacks has been successful for dedicated hash functions [12, 22, 61, 62, 321] and for hash functions based on modular arithmetic [252].

2.5.3 Attacks dependent on an interaction with the signature scheme

In some cases it is possible that even if the hash function is a CRHF, it is possible to break the signature scheme. This attack is then the consequence of a dangerous interaction between both schemes. In the known examples of such an interaction both the hash function and the signature scheme have some multiplicative structure. Examples are the attack by D. Coppersmith on a hash function based on modular arithmetic [58] and the attack by B. den Boer that is discussed in [65]. It was shown in [64] that the security of a digital signature scheme which is not existentially forgeable under a chosen message attack will not decrease if it is combined with a CRHF.

2.5.4 Attacks dependent on the underlying block cipher

Certain weaknesses of a block cipher are not significant when it is used to protect the privacy, but can have dramatic consequences if the cipher is used in one of the special modes for hashing. These weaknesses can be exploited to insert special messages or to carry out well chosen manipulations without changing the hashcode. The discussion

will be limited to the weaknesses of DES [8, 108], LOKI [33] and its improved variant LOKI91 [34], and PES [181] and its improved variant IDEA [182, 184].

2.5.4.1 Complementation property

One of the first properties that was known of DES was the symmetry under complementation [146]:

$$\forall\, P, K : C = \mathrm{DES}(K, P) \Longleftrightarrow \overline{C} = \mathrm{DES}(\overline{K}, \overline{P})$$

It can reduce an exhaustive key search by a factor 2 but it also allows to construct trivial collisions.

A more extended set of related properties of LOKI was described independently in [22, 180] and by B. den Boer [82]. They can be exploited to attack several hash modes, and also to speed up an exhaustive key search with a factor 256. In the new version of LOKI [34] it can be shown that only the complementation property holds.

2.5.4.2 Weak keys

Another well known property of DES is the existence of 4 weak keys [74, 231]. For these keys, encryption equals decryption, or DES is an involution. These keys are also called palindromic keys. This means that $E(K, E(K, P)) = P$, $\forall\, P$. There exist also 6 pairs of semi-weak keys, for which $E(K_2, E(K_1, P)) = P$, $\forall\, P$. This property can be exploited in certain hash functions to construct fixed points after two iterations steps. Compared to DES, LOKI had more weak keys, but LOKI91 has the same number of weak and semi-weak keys [34].

It was remarked by B. den Boer that a similar property holds for PES and IDEA: for the all zero key the cipher is an involution.

2.5.4.3 Fixed points

Fixed points of a block cipher are plaintexts that are mapped to themselves for a certain key. As a secure block cipher is a random permutation, it will probably have fixed points (for every key there is a probability of $1 - e^{-1}$ that there is at least a single fixed point). However, it should be hard to find these. Under some conditions it is easy to produce fixed points:

- For DES, this can be done based on a property of the weak keys [231]: for every weak key K_p, there exist 2^{32} values of P that can be easily found for which $\mathrm{DES}(K_p, P) = P$. A similar property holds for the anti-palindromic keys: these are 4 semi-weak keys for which there exist 2^{32} values of P that can be easily found for which $\mathrm{DES}(K_{ap}, P) = \overline{P}$.
- The block cipher LOKI has 256 simple fixed points where the key is of the special form $ggggggghhhhhhhh_x$, and the plaintext is equal to $iiiiiiiiiiiiiiii_x$, with $i = g \oplus h$ [22]. Here g, h and i are 4-bit numbers in hexadecimal notation. For every weak key there exist 2^{32} fixed points.

2.5.5 High level attacks

Even if the above attacks would not be feasible, special care has to be taken to avoid replay of messages and construction of valid messages by combining others.

For authentication of transmitted messages, attacks at this level can be thwarted by adding a nonce, this is a quantity that is never transmitted twice in a given context, and through the use of sound cryptographic protocols. It is essential to authenticate the integrity of the nonces together with the message.

Timestamps: the date and time of the moment at which the message is sent. If the resolution of the time is sufficiently high, it will provide a unique identifier of the message. For a resolution of one second, 5 to 6 bytes are sufficient. The two main problems are the cost of maintaining reasonably well synchronized clocks at both ends of the communication line and of delays in communication channels.

Serial numbers: a unique number is assigned to every message. A size of 4 bytes should be sufficient for most applications, depending on the lifetime of the key. If every user keeps a different sequence number for every user he communicates with, the serial numbers should be consecutive, and the deletion of a message can be detected. If every user has only one sequence number for all his communications, one has to check that the serial numbers form an increasing sequence. This is only possible if every user stores the highest sequence number of every communication. This system does not allow for checking for deleted messages. A serial number is less expensive than a time stamp, but the timeliness of the information can not be checked. This should be no problem for applications like electronic mail.

Random numbers: a sufficiently long random number is added to the message. To thwart a birthday attack on the number, it has to be larger than the square of the maximal number of messages that will be sent with a key. For most applications this means a size of about 8 bytes. A random number is not very useful if all previous random numbers have to be stored to detect a replay. However, if the random number is used in the next step of the protocol, it can offer an adequate protection.

In the case of stored information, a 'replay' attack becomes a 'restore' attack [74]. The serial numbers have to be replaced by version numbers, and a separate file is necessary that contains a single date and time stamp and for every file the current version number. If rearrangements of units that are protected by a different MAC is a problem, the address in the memory space can be protected together with the stored information.

2.6 Conclusion

In this chapter several types of cryptographic hash functions have been defined, with the emphasis on the system based or practical approach. It has been shown how

cryptographic hash functions provide an efficient way to protect integrity and to speed up digital signatures. A general model has been introduced that allows for a compact description of iterated hash functions and attacks.

Chapter 3

The Information Theoretic Approach

Faith is an island in the setting sun. But proof is the bottom line for everyone. *Paul Simon*

3.1 Introduction

The information theoretic approach results in a characterization of unconditionally secure solutions, which implies that the security of the system is independent of the computing power of the opponent.

This chapter starts with a brief summary of the basic theory of unconditionally secure authentication schemes and discusses some optimal schemes. Subsequently some schemes that are close to optimal are treated, with the emphasis on universal hash functions, as this concept will also be used in the next chapter. Finally it will be discussed whether these schemes are practical and how other, more practical schemes can be derived from them. The main contribution of this chapter is the comparison of the efficiency of the different schemes.

3.2 Basic theory

The first result on unconditionally secure authentication appeared in 1974 in a paper by E. Gilbert, F. MacWilliams, and N. Sloane [118]. Subsequently the theory has been developed by G. Simmons, analogous to the theory of secrecy systems that was invented by C. Shannon [303]. An overview of this theory can be found in [309, 310]. From the brief but clear summary by J. Massey in [202] we can cite the following statement *"The theory of authenticity is in many ways more subtle than the corresponding theory of secrecy. In particular, it is not at all obvious how "perfect authenticity" should be*

defined. This is caused by the fact that there are different bounds that can be met with equality. This section will give the basic definitions and will introduce a taxonomy. The most important theoretical bounds will be discussed (without proof).

3.2.1 Definitions and notations

According to the definitions that are used in the literature on this subject, the information to be communicated is called the *source state*, while the information that will be sent to the receiver is called the *message*. A mapping between the space of source states and the message space is called an *encoding rule*. The set of source states, messages and encoding rules is called an *authentication code*. In the following we will keep the more usual terminology of plaintext, ciphertext and key. The set of plaintext, ciphertexts, and keys will be denoted with $\{P\}$, $\{C\}$, and $\{K\}$ respectively. The size of plaintext, ciphertext, and key space will be denoted with p, c, and k respectively.

In the theory of authenticity, a game with three players is considered like in the theory of secrecy systems. First one has the sender Alice, who wants to send information to the receiver Bob under the form of a cryptogram. The opponent of Alice and Bob is the active eavesdropper Eve, who can perform three types of attacks:

- Eve can send a fraudulent cryptogram to Bob as if it came from Alice (*impersonation attack*).
- Eve can wait until she observes a cryptogram and replace it by a different cryptogram (*substitution attack*).
- Eve can choose freely between both strategies (*deception attack*).

The probability of success (when the strategy of Eve is optimal) will be denoted with P_i, P_s, and P_d respectively. A first result which follows from Kerckhoff's assumption (namely that the strategy to choose the key is known by Eve) is that

$$P_d = \max(P_i, P_s)\,.$$

This model can be extended in several ways [310]:

- If one also assumes that Alice and Bob are mutually distrustful, one needs a fourth person to resolve their disputes: the arbiter. This can be more than one person, who may have access to secret information. In this case the situation is much more complex, and one should also consider attacks by Alice, Bob, and by one or more arbiters. The difference with digital signatures is that a signature can in principle be verified and disputes can be resolved by any other person.
- Eve can wait until she observes l cryptograms, and subsequently perform a substitution or impersonation. It is obvious that here one should take care of the special case of simply replaying an old cryptogram or reordering the cryptograms (this can be solved by inserting a sequence number into the plaintext). The schemes that are resistant to this type of attack are called l-fold secure. If this extension is not considered, the authentication schemes are, just like the Vernam scheme,

one-time schemes: the secret key can be used only once. Only a few results will be described in this more general setting.

- The mapping from plaintext to ciphertext can be probabilistic. In this case one speaks of an authentication code *with splitting*.

In [88, 90] the model is extended to take into account the economical value of the plaintexts.

A basic distinction that can be made in this theory is between authentication codes with and without secrecy. In the latter case the plaintext can be derived easily from the ciphertext. The corresponding codes are called *Cartesian.* In this case one can write the ciphertext as the concatenation of the plaintext P with an authenticator, Message Authentication Code or MAC. The number of authenticators is denoted with r.

3.2.2 Bounds on authentication codes

The simplest bound that can be given is for an impersonation attack: if the enemy selects C completely at random from the c cryptograms that occur with nonzero probability in the authentication code, this probability of success can be lower bounded by the following expression:

Theorem 3.1 (Combinatorial Bound for Impersonation)

$$P_i \geq \frac{p}{c}\,. \tag{3.1}$$

One can show that this bound can not be met with equality if splitting occurs. For Cartesian codes this bound reduces to $P_i > 1/r$.

If no splitting occurs, a similar bound can be given for substitution.

Theorem 3.2 (Combinatorial Bound for Substitution)

$$P_s \geq \frac{p-1}{c-1}\,. \tag{3.2}$$

The next bound is based on the concept of mutual information. Therefore some definitions have to be introduced:

Definition 3.1 *Let $\{p(x)\}_{x\in X}$ be a probability distribution on a finite set X. The* **entropy** *$H(X)$ is defined as*

$$H(X) = -\sum_{x\in X} p(x)\log p(x)\,.$$

Definition 3.2 *Let X and Y be two sets and let $\{p(x,y)\}_{x\in X, y\in Y}$ be a joint probability distribution on their Cartesian product. The* **conditional entropy** *$H(X \mid Y)$ (or equivocation) of X given Y is defined as*

$$H(X \mid Y) = -\sum_{x\in X, y\in Y} p(x,y)\log p(x \mid y)\,.$$

Definition 3.3 *Let X and Y be two sets. The* **mutual information** *$I(X;Y)$ is defined as*

$$I(X;Y) = H(X) + H(Y) - H(XY)\,.$$

This can also be written as $I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$.

The following information theoretic bound can now be given on the probability of impersonation.

Theorem 3.3 (Authentication Channel Capacity)

$$P_i \geq 2^{-I(C;K)}\,. \tag{3.3}$$

For the shortest proof known until now and a discussion of the recent improvements on this bound by R. Johannesson and A. Sgarro the reader is referred to [202]. A formulation in words is that the difference between the amount of information transmitted through the channel and that needed by the receiver to resolve his uncertainty about the plaintext can be used to authenticate the plaintext, and conversely no better result can be achieved [310]. One can now show the following corollary, which was first proven by E. Gilbert, F. MacWilliams, and N. Sloane [118].

Corollary 3.1

$$P_d \geq \frac{1}{\sqrt{k}}\,. \tag{3.4}$$

For l-fold secure schemes where an opponent can first observe l ciphertexts, this can be extended to [103]

Theorem 3.4

$$P_d \geq \frac{1}{\sqrt[l+1]{k}}\,. \tag{3.5}$$

The next step is to define *perfect authenticity* to mean that equality holds in (3.3). However, even if a system is perfect, P_d will only be small if the cryptogram C reveals much information about the key K. Two simple examples of perfect authentication codes are given in table 3.1 [202].

K	$P=0$	$P=1$
00	00	10
01	01	11
10	00	11
11	01	10

K	$P=0$	$P=1$
00	00	10
01	01	00
10	11	01
11	10	11

Table 3.1: A perfect Cartesian authentication code and a perfect authentication code that also provides perfect secrecy.

One can conclude that perfect authenticity schemes are less efficient in terms of key bits than schemes that provide perfect secrecy [201]: if a p'-bit plaintext is mapped

to a c'-bit ciphertext, the number of key bits per plaintext bit for perfect authenticity with P_i and P_s equal to the combinatorial bound is at least $(c' - p')/p'$, which is much larger than 1 if p' is small. Note that $P_i = 1/2^{c'-p'}$, so $c' - p'$ can not be small.

3.2.3 Characterization of perfect Cartesian authentication codes

A large number of authentication codes have been proposed in the literature that meet these bounds (or a subset of them). We will not attempt to overview all these constructions, but we will concentrate on characterizations. A characterization of an authentication code with certain properties shows that these codes can be obtained essentially in one way, which means that all other constructions are equivalent. For the time being no general characterization of perfect authentication codes is known. For Cartesian codes some characterizations exist in terms of combinatorial structures.

For Cartesian authentication codes that meet the bound (3.4) with equality, a characterization was given by M. De Soete, K. Vedder, and M. Walker [93] based on nets. This result was in fact a generalization of the work by E. Gilbert, F. MacWilliams, and N. Sloane [118].

For Cartesian authentication codes that meet the combinatorial bound for impersonation and substitution with equality, a characterization was given by D. Stinson [313] based on orthogonal arrays. It should be noted that only for a subset of these codes the bound (3.4) is met with equality. For general codes that meet the two combinatorial bounds, D. Stinson has derived characterizations based on balanced incomplete block designs.

3.3 Practical Cartesian authentication codes

In the following more practical authentication schemes will be discussed that are still unconditionally secure. To be more in line with the other chapters, the notation will be changed: an m-bit message will be authenticated with a n-bit MAC. The problem with the perfect constructions is that a key of at least $2m$ bits is required. The basic idea to make these schemes more efficient is to allow that P_s increases with an acceptable factor under the assumption that this will reduce the size of the key.

In [91] a different approach was also explored: it was shown how the description and calculations for the scheme can be made extremely simple at the cost of increasing the key size to $(n + 1)m$ bits.

It will be discussed how the perfectly secure schemes behave in a practical context, and several schemes based on universal hash functions will be compared. The theory of universal hash functions will be discussed in more detail, as this concept will turn out to be useful in chapter 4 on the complexity theoretic approach.

3.3.1 The perfect schemes

The property of the perfect schemes with minimal key size based on nets [93] is that $P_s = P_i = 2^{-n}$, and the key size is $2n$ bits. One can show that in this case $2^m \leq 2^n + 1$

or $m < n$, and hence the key size is larger than $2m$. In a practical problem, one designs a system with given P_d and size of the plaintext. One can distinguish between three situations:

- For very small messages ($m \ll n$), these schemes are very inefficient because the key size is much larger than twice the message size.
- For medium sized messages ($m \approx n$) the key size will be equal to $2m$, and hence the schemes make optimal use of the key.
- For large messages ($m \gg n$), it follows that the key size will be equal to about $2m$, but P_d will be much smaller than required. This can be avoided by splitting the message in n bit blocks and authenticating these separately: the efficiency in terms of key bits will remain the same, but the calculations will be on smaller blocks, and hence the scheme will be more efficient [91].

3.3.2 Universal hash functions

Universal hash functions have been proposed by J. Carter and M. Wegman [42, 43] and were studied further by D. Sarwate, [293], M. Wegman and J. Carter [325] and D. Stinson [312, 314]. J. Carter and M. Wegman proposed to use universal hash functions for unconditionally secure authentication schemes and for probabilistic algorithms for testing set equality. Ten years later universal hash functions were shown to be a useful building block for provably secure hash functions, as will be shown in chapter 4.

3.3.2.1 Definitions

A universal hash function is a mapping from a finite set A with size a to a finite set B with size b. For a given hash function g and for a pair (x, x') with $x \neq x'$ the following function is defined:

$$\delta_g(x, x') = \begin{array}{l} 1 \text{ if } g(x) = g(x') \\ 0 \text{ otherwise.} \end{array}$$

For a finite set of hash functions G (in the following this will be denoted with a *class* of hash functions), $\delta_G(x, x')$ is defined as $\sum_{g \in G} \delta_g(x, x')$, or $\delta_G(x, x')$ counts the number of functions in G for which x and x' collide. When a random choice of g is made, then for any two distinct inputs x and x', the probability that these two inputs yield a collision equals $\delta_G(x, x')/ \mid G \mid$. In the case of a universal hash function, the goal is to minimize this probability together with the size of G.

A lower bound on $\delta_G(x, x')/ \mid G \mid$ has been proven in [293] (improving slightly a bound in [325]):

Theorem 3.5 *For any class of hash functions from A to B, there exist distinct elements x and x' such that*

$$\delta_G(x, x') \geq \frac{a - b}{b(a-1)} \cdot \mid G \mid \ .$$

Moreover, equality will hold only if a is a multiple of b. For $a \gg b$ the right hand of the equation can be approximated by $1/b$.

Definition 3.4 *Let ϵ be any positive real number. An* **ϵ-almost universal$_2$** *class (or $\epsilon - AU_2$ class) $\mathcal{G}$ of hash functions from a set A to a set B is a class of functions from A to B such that for any distinct elements $x, x' \in A$*

$$\mid \{g \in G : g(x) = g(x')\} \mid = \delta_G(x, x') \leq \epsilon \cdot \mid G \mid \ .$$

This definition states that for any two distinct inputs the probability for a collision is at most ϵ. Because of theorem 3.5 the lowest possible value for ϵ is $\frac{a-b}{b(a-1)}$. This class of functions is called optimally universal [293]. In [42] the case $\epsilon = 1/b$ is called universal.

Definition 3.5 *Let ϵ be any positive real number. An* **ϵ-almost strongly universal$_2$** *class (or $\epsilon - ASU_2$ class) $\mathcal{G}$ of hash functions from a set A to a set B is a class of functions from A to B such that*

- *for every $x \in A$ and for every $y \in B$,*

$$\mid \{g \in G : g(x) = y\} \mid = \frac{\mid G \mid}{b} \, ,$$

- *for every $x_1, x_2 \in A$ $(x_1 \neq x_2)$ and for every $y_1, y_2 \in B$ $(y_1 \neq y_2)$,*

$$\mid \{g \in G : g(x_1) = y_1, g(x_2) = y_2\} \mid \leq \epsilon \cdot \frac{\mid G \mid}{b} \, .$$

The first condition states that the probability that a given input x is mapped to a given output y equals $1/b$. The second condition implies that if x_1 is mapped to y_1, then the conditional probability that x_2 (different from x_1) is mapped to y_2 is upper bounded by ϵ. The lowest possible value for ϵ equals $1/b$ and this class has been called strongly universal$_2$ functions in [325].

If more than two inputs at the same time are considered, the following generalizations can be made.

Definition 3.6 *Let ϵ be any positive real number, and let r be an integer with $r > 2$. An* **ϵ-almost strongly universal$_r$ class** *(or $\epsilon - ASU_r$ class) $\mathcal{G}$ of hash functions from a set A to a set B is a $\epsilon - ASU_2$ class of hash functions from A to B such that for any distinct elements $x_1, x_2, \ldots x_r \in A$ and for any (not necessarily distinct) elements $y_1, y_2, \ldots y_r \in B$ if*

$$\mid \{g \in G : g(x_1) = y_1 \wedge g(x_2) = y_2 \wedge \ldots \wedge g(x_r) = y_r\} \mid \leq \epsilon \cdot \mid G \mid \ .$$

Note that this is equivalent to saying that the random variables $\{g(x) \mid x \in A\}$ are uniform and r-wise independent. The case $\epsilon = 1/b^r$ has been called strongly universal$_r$ in [325]. More work has to be done to characterize universal$_r$ hash functions. To simplify the notation, the index 2 will be omitted from universal$_2$ from hereon.

3.3.2.2 Constructions

Universal and strongly universal hash functions can be constructed both with simple direct methods and with recursive methods, i.e., by composition of other hash functions. The examples of direct constructions that have been given for any prime power q [312, 314] are summarized in table 3.2. Other constructions can be found in [42, 43].

type	a	b	$\|G\|$	ϵ	expression
$\epsilon - AU_2$	q^2	q	q	$1/q$	$b - ax$
$\epsilon - ASU_2$	q^2	q	q^3	$1/q$	$x + ay + bz$
$\epsilon - ASU_2$	q	q	q^2	$1/q$	$x + ay$

Table 3.2: Summary of direct constructions of universal hash functions. Here q is a prime power; a, b are elements of $GF(q)$; x, y, and z are independent variables.

It will be shown that for authentication codes it is required that $|G|$ is small. For the $\epsilon - ASU_2$ constructions in table 3.2, the number of functions is larger than the product of a and b, which is a corollary from theorem 3.10. M. Wegman and J. Carter showed how to obtain an ϵ'-ASU$_2$ [325] from a $2/b$-ASU$_2$ with $\epsilon' = (2/b) \cdot \log_2 \log_2 a$[1]. The idea is to use a universal hash function that maps $2s$-bit strings to s-bit strings (with $s = \log_2 b + \log_2 \log_2 \log_2 a$) in a tree structure. The number of functions is equal to

$$|G| = 4s \log_2 \log_2 a \,.$$

The following theorems of D. Stinson [314] show how (strongly) universal hash functions can be used in recursive constructions. These theorems generalize similar constructions that were given before. These theorems can be further generalized to universal$_r$ hash functions.

Theorem 3.6 (Cartesian Product) *If there exists an $\epsilon - AU_2$ class G of hash functions from A to B, then, for any integer $i \geq 1$, there exists an $\epsilon - AU_2$ class G^i of hash functions from A^i to B^i with $|G| = |G^i|$.*

Theorem 3.7 (Composition 1) *If there exists an $\epsilon_1 - AU_2$ class G_1 of hash functions from A to B and an $\epsilon_2 - AU_2$ class G_2 of hash functions from B to C, then there exists an $\epsilon - AU_2$ class G of hash functions from A to C, where $\epsilon = \epsilon_1 + \epsilon_2$ and $|G| = |G_1| \cdot |G_2|$.*

Theorem 3.8 (Composition 2) *If there exists an $\epsilon_1 - AU_2$ class G_1 of hash functions from A to B and an $\epsilon_2 - ASU_2$ class G_2 of hash functions from B to C, then there exists an $\epsilon - ASU_2$ class G of hash functions from A to C, where $\epsilon = \epsilon_1 + \epsilon_2$ and $|G| = |G_1| \cdot |G_2|$.*

[1] In fact they claim that $\epsilon' = (2/b)$, but this has been corrected by D. Stinson.

Based on these recursive constructions, the schemes described in table 3.3 can be obtained [314].

type	a	b	$\|G\|$	ϵ
$\epsilon - AU_2$	q^{2^i}	q	q^i	i/q
$\epsilon - ASU_2$	q^{2^i}	q	q^{i+2}	$(i+1)/q$
$\epsilon - ASU_2$	q^{2^i}	q	q^{2i+3}	$i/q^2 + 1/q$

Table 3.3: Summary of recursive constructions of universal hash functions. Here q is a prime power, and i is an integer ≥ 1.

A simple example of a universal$_r$ family is obtained by chopping the first n bits of a polynomial of degree $r-1$ over the finite field $GF(2^m)$ [325]:

$$G_{a_0,a_1,\ldots,a_{r-1}} =$$
$$\left\{ g_{a_0,a_1,\ldots,a_{r-1}}(x) = \mathrm{chop}^n\left(a_0 + a_1 x + \cdots + a_{r-1}x^{r-1}\right) \mid a_0, a_1, \ldots, a_{r-1} \in GF(2^m) \right\} .$$

Here $\mathrm{chop}^n()$ returns the first n bits of its m-bit argument. Note that exactly 2^{n-m} elements in the domain have the same value in the range.

3.3.2.3 Authentication codes based on universal hash functions

The following theorem indicates how ϵ-ASU$_2$ classes of hash functions can be used to construct an authentication code [314].

Theorem 3.9 *If there exists an ϵ-ASU_2 class G of hash functions from A to B, then there exists a Cartesian authentication code with a plaintexts, b authenticators and $k = |G|$ keys, such that $P_i = 1/b$ and $P_s \leq \epsilon$.*

A lower bound on the number of keys can be obtained with the following theorem [314].

Theorem 3.10 *If there exists an ϵ-ASU_2 class G of hash functions from A to B, then*

$$|G| \geq 1 + \frac{a(b-1)^2}{\epsilon\, b(a-1) + b - a} .$$

Schemes that achieve this bound will be called optimal ϵ-ASU$_2$ schemes. No constructions are known that achieve this lower bound, except when $\epsilon = 1/b$, but this corresponds to the well known case where $P_s = P_i$.

In the following it will be assumed that $a = 2^m$ and $b = 2^n$. The scheme proposed by M. Wegman and J. Carter results in an authentication code with $P_s = 2\log_2 m/2^n$ and with a key of $4(m+\log_2\log_2 n)\log_2 n$ bits. The recursive construction by D. Stinson (the second scheme in table 3.3) yields an authentication code with $P_s = (\log_2 m - \log_2 n + 1)/2^n$ and with a key of $(\log_2 m - \log_2 n + 2)n$ bits.

In order to obtain schemes that are l-fold secure, one can construct an authentication code based on an ϵ-ASU$_{l+1}$ [325]. This construction will be very inefficient in terms of use of key material: the key size will grow as the $l+1$th power of $1/P_d$. A more efficient and more general solution will be described in the following section.

In [49] a very simple scheme is suggested to produce an ϵ-ASU$_2$ with $\epsilon = 3(m+n)/2^n$ under the condition that $m \geq n \geq 64$. The key consists of two elements: an n-bit prime p that is larger than 2^{n-1} and an n-bit integer ν. This means that the key size is equal to $2n - \log_2(2n \ \ln 2)$. The function is defined as

$$g(x) = [(x \cdot 2^n) \bmod p + \nu] \bmod 2^n \, .$$

Based on theorem 3.9 one can show that this generates an authentication code with $P_i = 2^{-n}$ and $P_s = 3(m+n)/2^n$.

An even better and more elegant scheme was proposed recently by B. den Boer [84]. It produces an ϵ-ASU$_2$ with $\epsilon = (m/n)/2^n$. The key consists of 2 elements of $GF(2^n)$ denoted with μ and ν. The argument x is split in elements of $GF(2^n)$ denoted with x_1, x_2, ..., x_t, hence $m = t \cdot n$. The function is then defined as follows:

$$g(x) = \mu + \sum_{i=1}^{t} x_i \cdot \nu^i \, ,$$

where the addition and the multiplication are in $GF(2^n)$. It is easy to prove that this function is an ϵ-ASU$_2$. Given x, $g(x)$, and ν, there is exactly one corresponding value of μ. Moreover, if one knows a pair $g(x), g(x')$ with $x \neq x'$, one can solve for μ and ν as follows. Subtract the two equations in order to eliminate μ. This yields an equation in ν of degree t, that has at most t solutions, which results in the claimed value for ϵ. From theorem 3.9 it follows that one has an authentication code with $P_i = 2^{-n}$ and $P_s = (m/n)/2^n$. It is clear that this scheme can be generalized to any finite field, but the arithmetic will be more complicated. B. den Boer has indicated other fields for which efficient implementations exist [84]. Observe that if $m = n$ this construction reduces to the construction by E. Gilbert, F. MacWilliams, and N. Sloane [118].

3.3.3 A comparative overview

Table 3.4 gives an overview of the probability for impersonation and substitution, and the key size for the authentication codes that have been discussed. It is assumed that an m-bit message is mapped to an n-bit hashcode. For all schemes the probability of impersonation $P_i = 1/2^n$.

A concrete example for an authenticator with $n = 64$ bits is worked out in table 3.5. It can be seen that the scheme by D. Chaum et al. is most efficient in terms of key bits, at the cost of a significant increase in P_s. For large values of m, this factor is equal to $3m$. If this factor is substituted for t in the equation for the optimal ϵ-ASU$_2$ schemes, one finds that the scheme by D. Chaum et al. has a key that is about $\log_2(3m)$ bits larger than for the optimal construction. The scheme by B. den Boer yields a smaller

	P_s	key size k (bits)
perfect	$1/2^n$	$2m$
opt. ϵ-ASU$_2$	$t/2^n$	$\log_2(1 + (2^m(2^n-1)^2)/(t(2^m-1)+2^n-2^m))$
Wegman-Carter	$(2\log_2 m)/2^n$	$(n + \log_2\log_2 m)4\log_2 m$
Stinson	$(\log_2 m - \log_2 n + 1)/2^n$	$(\log_2 m - \log_2 n + 2)n$
Chaum et al.	$3(m+n)/2^n$	$2n - \log_2(2n\ \ln 2)$
den Boer	$(m/n)/2^n$	$2n$

Table 3.4: Probability of substitution P_s and key size k of the perfect scheme, the optimal scheme with minimal key size based on ϵ-ASU$_2$, and the schemes by M. Wegman and J. Carter, D. Stinson, D. Chaum et al., and B. den Boer.

scheme	$m = 64$		$m = 2^{20}$	
	P_s	key size k (bits)	P_s	key size k (bits)
perfect	$1/2^{64}$	128	$1/2^{64}$	2^{21}
opt. ϵ-ASU$_2$	$t/2^{64}$	$128 - \log_2 t$	$t/2^{64}$	$128 - \log_2(t-1)$
Wegman-Carter	$12/2^{64}$	1,598	$1/2^{58.7}$	5,466
Stinson	$1/2^{64}$	128	$1/2^{60.1}$	1,024
Chaum et al.	$1/2^{55.4}$	122	$1/2^{42.4}$	122
den Boer	$1/2^{64}$	128	$1/2^{50}$	128

Table 3.5: P_s and key size k for the perfect schemes, the optimal scheme with minimal key size based on ϵ-ASU$_2$, and the schemes by M. Wegman and J. Carter, D. Stinson, D. Chaum et al., and B. den Boer. The size of the authenticator n is equal to 64 bits.

increase of P_s, while the key is only about $\log_2(m/n)$ bits larger than for the optimal construction.

In fact one could also think of a more realistic scenario where not n and m are given, but where the size of the message m and P_d are imposed. In this case one has to determine n and k. For large m ($m \gg n$) one obtains the following approximate formulas for the scheme by D. Stinson:

$$\begin{aligned} k &= (-\log_2 P_d + \log_2 \log_2 m) \cdot (\log_2 m + 2 - \log_2 (-\log_2 P_d + \log_2 \log_2 m)) \\ &\approx (-\log_2 P_d + \log_2 \log_2 m) \cdot (\log_2 m + 2) \\ n &= -\log_2 P_d + \log_2 \log_2 m \,. \end{aligned}$$

Under these conditions the scheme by D. Chaum et al. yields:

$$\begin{aligned} k &= -2\log_2 P_d + 2\log_2 m + 2\log_2 3 \\ n &= -\log_2 P_d + \log_2 m + \log_2 3 \,. \end{aligned}$$

For the scheme by B. den Boer one finds approximately:

$$\begin{aligned} k &= -2\log_2 P_d + 2\log_2 m - 2\log_2(-\log_2 P_d + \log_2 m) \\ n &= -\log_2 P_d + \log_2 m - \log_2(-\log_2 P_d + \log_2 m) \,. \end{aligned}$$

From these equations one can conclude that the authenticator of the scheme by B. den Boer will be at most $\log_2 m$ bits larger than for the scheme by D. Stinson, which approximates the perfect schemes very closely. On the other hand the key size for the scheme by D. Stinson is about $(-\log_2 P_d) \cdot \log_2 m$, while the key size for the scheme by B. den Boer is approximately equal to $-2\log_2 P_d + 2\log_2 m$. The scheme by D. Chaum et al. performs slightly worse than the scheme by B. den Boer.

From an implementation point of view, one can note that the constructions by M. Wegman and J. Carter, by D. Stinson, by D. Chaum et al., and by B. den Boer require only very simple operations (multiplication and addition in a finite field $GF(q^n)$ or $GF(q)$ with q about n bits in size, followed by chopping of bits, or a modular reduction modulo an n-bit prime). Note however that this will be about five to ten times faster than a conventional MAC based on the CBC mode of the DES algorithm, that will be discussed in chapter 5. The basic practical problem however is that a new secret key has to be exchanged for every message. This is mainly a problem if there are many small messages. In that case one could however use the following approach: protect the individual messages with a practical and fast MAC, and apply from time to time a provably secure scheme to all collected messages. Other solutions are to derive the key from the from the previous key, or to design the scheme such that every key can be used l times.

In the first case the key can be generated from a small seed with a computationally or practically secure scheme, which implies that the scheme is no longer unconditionally secure. For a computationally secure scheme one will use a secure pseudo-random string generator. For a practically secure scheme one can choose between a stream

cipher and a mode (e.g. OFB mode) of a block cipher. In both cases the exchange and storage of the key material will decrease, but the computational overhead will increase. This will probably make the schemes of B. den Boer and of D. Chaum et al. more efficient. On the other hand these schemes are more expensive in terms of storage for the authenticator (at least if the messages are larger than 100 Kbytes), which might be a small disadvantage for some applications. A second disadvantage, which only applies for the scheme by D. Chaum et al., is that the generation of a new key, that comprises primality tests, is more than three orders of magnitude slower than performing the key scheduling for the DES algorithm.

In the second case an l-fold secure scheme is designed. This means that a particular key can be used to authenticate l messages. This is possible if the authenticator is encrypted with a good cipher. Again a distinction between three cases can be made:

1. M. Wegman and J. Carter suggest in [325] to use the Vernam scheme, which implies that the scheme will be still unconditionally secure. The number of additional key bits to authenticate l messages will be equal to ln.
2. A second proposal by G. Brassard [29], is to use a secure pseudo-random string generator (cf. chapter 4), which implies that the security is now based on complexity theoretic assumptions.
3. A practical and efficient scheme can be obtained by simply encrypting the hash-code with a 'good' block cipher or stream cipher.

In all three cases the messages should contain a sequence number to avoid replay and reorder attacks. In the scheme by D. Chaum et al. one can also modify part of the key, but this implies that after $l+1$ messages the probability of substitution will increase to

$$P_s^{(l+1)} = \frac{1}{\frac{1}{P_s} - l},$$

from which it follows that l should not be too large. This can be avoided by modifying the complete key after every message. Similarly, one could modify the scheme by B. den Boer such that only μ is replaced.

3.4 Conclusion

In this chapter an overview has been given of schemes that can protect the authenticity of information unconditionally. The theory of these schemes is rather subtle, and therefore it is developing more slowly than the theory of unconditionally secure secrecy. A characterization in terms of combinatorial constructions can only be given for a small subclass of schemes. The schemes that come closest to practical schemes are unconditionally secure Message Authentication Codes or MAC's based on strongly universal hash functions. One can always derive both complexity theoretic and practical constructions from these schemes, but for the time being it seems more efficient to design this type of schemes directly.

Chapter 4

The Complexity Theoretic Approach

Out of intense complexities intense simplicities emerge. *Winston Churchill*

4.1 Introduction

The background of the complexity theoretic approach is the definition of a model of computation. In the uniform model this is a Turing machine [5], while a Boolean function is the model of computation in the non-uniform model. No detailed discussion will be given on the difference between the two models, but it is important to remark that only in the non-uniform model precomputations, that are carried out by an adversary before the start of the protocol are included. All computations in this model are now parameterized by a security parameter and the asymptotic behavior of algorithms is studied. In the uniform model, only algorithms that require time and space polynomial in the size of the input are considered to be feasible. Algorithms that require exponential time and/or space in the size of the input are considered to be infeasible. Along the same lines, an exponentially small fraction is considered to be negligible. Note that in this context a birthday attack does not make sense: it reduces the number of operations to the square root of the number of operations for a brute force attack. However, the square root of a polynomial is still a polynomial and the square root of an exponentially growing function still grows exponentially.

Before a complexity theoretic treatment of a OWHF and a CRHF is possible, some basic definitions have to be discussed, together with some background on pseudo-random string generators and one-way functions. Subsequently, a definition and some important constructions for a OWHF and a CRHF will be given. An extension of the concept of a OWHF will be studied, and it will be discussed how computationally secure perfect authentication codes can be constructed.

Finally it is remarked that some complexity theoretic constructions can be obtained from unconditionally secure schemes, as it has been discussed in the previous chapter.

4.2 Complexity theoretic definitions

4.2.1 Basic definitions

The set of all integers will be denoted with $\mathbb{N}$. The alphabet considered is the binary alphabet $\Sigma = \{0, 1\}$. For $n \in \mathbb{N}$, Σ^n is the set of all binary strings of length n. The set of all strings of arbitrary length will be written as Σ^*. The concatenation of two binary strings x and y will be denoted with $x \| y$. Let $l(n)$ be a monotone increasing function from $\mathbb{N}$ to $\mathbb{N}$, and f be a function from D to R where $D = \bigcup_n D_n$, $D_n \subseteq \Sigma^n$ (in most cases $D_n = \Sigma^n$), and $R = \bigcup_n R_n$, $R_n \subseteq \Sigma^{l(n)}$. D is called the domain and R is called the range of f. The restriction of f to Σ^n will be denoted with f_n. The function f is polynomial time computable if there is a polynomial time algorithm computing $f(x)$, $\forall x \in D$. In the following it will be assumed that there is a description of f_n of length polynomial in n and that f_n is polynomial time computable. The composition $f \circ g$ of two functions f and g is defined as $(f \circ g)(x) = f(g(x))$, and the k-fold composition of f is denoted by $f^{(k)}$. The size of a set S is denoted with $|S|$.

A probability ensemble E, with length $l(n)$, is a family of probability distributions $\left\{E_n : \Sigma^{l(n)} \longrightarrow [0,1], n \in N\right\}$. The uniform ensemble U with length $l(n)$ is the family of uniform probability distributions U_n, where each U_n is defined as $U_n(x) = 1/2^{l(n)}$, $\forall x \in \Sigma^{l(n)}$. By $x \in_E \Sigma^{l(n)}$ we mean that x is randomly selected from $\Sigma^{l(n)}$ according to E_n, and in particular by $x \in_R \Sigma^{l(n)}$ we mean that x is chosen from the set $\Sigma^{l(n)}$ uniformly at random. E is samplable if there is an algorithm M that on input n, outputs an $x \in_E \Sigma^{l(n)}$, and polynomially samplable if the running time of M is also polynomially bounded.

The siblings of x under a function f is the set S_x of elements that are mapped to $f(x)$, and s_n is the number of strings in Σ^n for which $|S_x| > 1$. A function f is called an injection if each f_n $(n > n_0)$ is a one-to-one function, which is equivalent to $|S_x| \leq 1, \forall x$ (with $|x| > n_0$), or $s_n = 0$, $n > n_0$. In some cases a slightly more general concept is necessary. A function f is called a quasi-injection if the following holds: for any polynomial Q and for sufficiently large n: $s_n < 1/Q(n)$. An even more general concept is a function with small expected preimage size. This implies that there exists a polynomial Q such that for $x \in_R \Sigma^n$, the expected size of $f^{-1}(f(x)) < Q(n)$. A function f is called a permutation if each f_n is a one-to-one and onto function.

In this chapter chop_t denotes the function from Σ^{n+t} to Σ^n that drops the t rightmost bits of its argument.

Let $\{n_{0_i}\}$ and $\{n_{1_i}\}$ be two (increasing) sequences such that $n_{0_i} < n_{1_i}$ for all i, but there exists a polynomial Q such that $Q(n_{0_i}) > n_{1_i}$, then these two sequences are polynomially related.

The important concept of a function family has been defined in [65]. A simplified version will be given here.

Definition 4.1 *A* **function family** $\mathcal{F}$ *is an infinite family of finite sets* $\{F_n\}_{n=1}^{\infty}$, *where* F_n *is the set of instances of size* n. *An instance* $f \in F_n$ *is a tuple,*

$$S = (f_n, D, R)\,,$$

where f_n *is a function* $f_n : D_n \longrightarrow R_n$. *The following requirements have to be imposed:*

- F_n *is accessible, or there is a polynomial time algorithm, which on input* n *outputs an instance chosen uniformly from* F_n.
- D_n *is samplable, or there is a polynomial time algorithm, which selects an element uniformly from* D_n.
- f_n *is polynomial time computable, or given* $x \in D_n$ *there is a probabilistic polynomial time algorithm (polynomial in* n *and in* $|\,x\,|$*) that computes* $f_n(x)$.

4.2.2 Pseudo-random string generators

Definition 4.2 *A* **statistical test** *is a probabilistic polynomial time algorithm* T *that on input* x *outputs a bit* $0/1$.

Definition 4.3 *Let* P *be a polynomial, and* E^1 *and* E^2 *be ensembles both with length* $l(n)$. E^1 *and* E^2 *are called* **indistinguishable** *from each other, iff for each statistical test* T, *for each polynomial* Q, *and for all sufficiently large* n,

$$\mid \Pr\{T(x_1) = 1\} - \Pr\{T(x_2) = 1\} \mid < \frac{1}{Q(n)}\,,$$

where $x_1 \in_{E^1} \Sigma^{l(n)}$, $x_2 \in_{E^2} \Sigma^{l(n)}$.

Definition 4.4 *A polynomially samplable ensemble* E *is* **pseudo-random** *if it is indistinguishable from the uniform ensemble* U *with the same length.*

Definition 4.5 *A* **string generator** *extending an* n*-bit input into an* $l(n)$ *bit output (here* $l(n) > n$*), is a deterministic polynomial time computable function* $g : D \longrightarrow R$.

In the following g will also be denoted by $\{g_n \mid n \in \mathbb{N}\}$. Let $g_n(U)$ be the probability distribution defined by the random variable $g_n(x)$, where $x \in_R \Sigma^n$, and let $g(U) = \{g_n(U) \mid n \in \mathbb{N}\}$. $g(U)$ is polynomially samplable. The definition of a pseudo-random string generator (PSG) can now be given [331].

Definition 4.6 $g = \{g_n \mid n \in \mathbb{N}\}$ *is a* **pseudo-random string generator (PSG)** *iff* $g(U)$ *is pseudo-random.*

Another important aspect of pseudo-random sequences is that given a part of the sequence, it should be hard to predict the remaining part.

Definition 4.7 *Let l be a polynomial, and E be an ensemble with length $l(n)$. Then E* **passes the next bit test** *iff for each statistical test T, for each polynomial Q, and for all sufficiently large n the probability that on input of the first i bits of a sequence x randomly selected according to E and $i < l(n)$, T outputs the $i+1$th bit of x is polynomially close to $1/2$, or*

$$\Pr\{T(x_1, \ldots, x_i) = x_{i+1}\} < \frac{1}{2} + \frac{1}{Q(n)},$$

where $x \in_E \Sigma^{l(n)}$.

A key result is the equivalence between the unpredictability and indistinguishability [331].

Theorem 4.1 *Let E be a polynomially samplable ensemble, the following statements are equivalent:*

- *E passes the next bit test.*
- *E is indistinguishable from the uniform ensemble U.*

4.2.3 One-way functions

An intuitive definition of a one-way function is that it is a function that should be hard to invert. Of course one can always take a polynomial number of inputs S and evaluate the function for the values in S: for all values in Image(S) it is now easy to invert the function with a simple table lookup. The solution is to require that it should be hard to invert the function almost everywhere. Many flavors of one-way functions exist, and the subtle differences can only be captured in a formal definition. In practice it can be shown that in the most important cases these definitions are equivalent.

Definition 4.8 *A* **one-way** *function family $\mathcal{F}$ is a function family that satisfies the following condition. Let x be selected uniformly in D_n and let M be a probabilistic polynomial time algorithm that takes as input $f_n(x) \in R_n$ and outputs $M(f_n(x)) \in D_n$. For each M, for each polynomial Q, and for all sufficiently large n*

$$\Pr\{f_n(M(f_n(x))) = f_n(x)\} < \frac{1}{Q(n)}.$$

If $D = R$, one has a one-way permutation family.

Observe that x can also be selected according to a different distribution, but in order to avoid unnecessary complications, it will always be assumed that x is selected uniformly in D_n.

A more complex definition can be given if it is only required that inverting more than a negligible fraction of the instances of a given size is hard [65]. It can be shown that the existence of one-way permutations according to this definition is in fact implied by the existence of permutations that are one-way in a much weaker sense. Moreover

if the function involves some group structure, it can be shown that if the function can be inverted on a non-negligible fraction of its images, it can be inverted everywhere with non-negligible probability.

The fact that f is a one-way function implies that given $f(x)$ there are at least $O(\log n)$ bits of x that are hard to predict. If such a bit is not biased, it is called a hard bit of f. A formal definition of a hard bit is given below:

Definition 4.9 *Let f be a one-way function. Let $i(n)$ be a function from $\mathbb{N}$ to $\mathbb{N}$ with $1 \leq i(n) \leq n$. The $i(n)$th bit is a* **hard bit** *of f iff for each probabilistic polynomial time algorithm M, for each polynomial Q, and for sufficiently large n,*

$$\Pr\left\{M(f_n(x)) = x'_{i(n)}\right\} < \frac{1}{2} + \frac{1}{Q(n)}\,,$$

where $x \in_R \Sigma^n$ and $x'_{i(n)}$ is the $i(n)$th bit of an $x' \in \Sigma^n$ satisfying $f(x) = f(x')$.

If a sufficient number of bits of x are given, it is always possible to find the remaining bits by exhaustive search. This concept is captured in the definition of a computing resource.

Definition 4.10 *One has a* **computing resource for k bits** *if given the output of a one-way function and $n-k$ bits of the input string, one can find the remaining k bits of the input string by exhaustive search.*

It can be shown that all the hard bits of f are independent. Given only $f(x)$, any string of hard bits is indistinguishable from a random string. This is formally stated as follows.

Definition 4.11 *Let f be a one-way function, and let $I = \{i_1, \ldots i_t\}$ be a subset of $\{1, 2, \ldots n\}$ with $t < n-k$. Denote by E_n^I and E_n^R the probability distributions defined by the random variables $x_{i_t(n)} \ldots x_{i_1(n)} \| f(x)$ and $r_t \ldots r_1 \| f(x)$ respectively, where $x \in_R \Sigma^n$ and $r_j \in_R \Sigma$. Let $E^I = \left\{E_n^I \mid n \in \mathbb{N}\right\}$ and $E^R = \left\{E_n^R \mid n \in \mathbb{N}\right\}$. Then the bits $x_{i_t(n)} \ldots x_{i_1(n)}$ are* **simultaneously hard** *if E^I and E^R are indistinguishable.*

The collection of all simultaneously hard bits of a one-way function f is called the **hard core** of f. In the following it will be assumed that the size of the hard core is at least $k+1$ bits. The maximal number of simultaneously hard bits is smaller than $n-k$. It will be denoted with $n-k^+$, where $k^+(n)$ is a function that grows slightly faster than $k(n)$.

Finally it is remarked that for any one-way function one can construct $O(\log n)$ hard core predicates: O. Goldreich and L. Levin [127] showed that for any one-way function f, given $f(x)$ and $p \in_R \Sigma^{|x|}$, the inner product of x and p is a hard core predicate of f, which means that it can not be predicted with a success probability that is better than $1/2 + 1/Q(n)$ for any polynomial Q.

4.3 Complexity theoretic constructions

4.3.1 Universal hash functions and uniformizers

For the definition of universal hash functions and for an overview of some constructions, the reader is referred to section 3.3.2. If these functions have to be defined correctly in a complexity theoretic setting, one has to state that they should be polynomial time computable. Moreover one has to replace the concept class of functions by the concept of a function family.

J. Carter and M. Wegman suggest to use universal hash functions for unconditionally secure authentication schemes (cf. section 3.3.2) and for probabilistic algorithms for testing set equality. Ten years later universal hash functions were shown to be a useful building block for provably secure hash functions. However, their use in these constructions requires an additional property, namely that it should be easy to find collisions. This property was first suggested in [233], but here the slightly extended version of [87] will be given.

Definition 4.12 *A (strongly) universal$_r$ family $\mathcal{G}$ has the* **collision accessibility property** *iff, given a requirement $g_i(x_1) = y_1 \wedge \ldots \wedge g_i(x_r) = y_r$, it is possible to generate in polynomial time a function g_i uniformly among all functions in $\mathcal{G}$ that obeys the requirement.*

For the constructions in [233, 342] the requirement is of the form $g_i(x_1) = g_i(x_2) = \cdots = g_i(x_r)$. This will be called the "weak collision accessibility property".

A concept closely related to a strongly universal hash function is that of a pair-wise independent uniformizer, defined in [341].

Definition 4.13 *A* **pair-wise independent uniformizer** *family $\mathcal{V}$ from n-bit strings to n-bit strings is a collection $\{v_i\}$ of permutations of Σ^n such that:*
$\forall n$, $\forall(x_1, x_2)$ with $x_1, x_2 \in \Sigma^n$ and $x_1 \neq x_2$ and $\forall(y_1, y_2)$ with $y_1, y_2 \in \Sigma^n$ and $y_1 \neq y_2$, there are exactly

$$\frac{|\,V_n\,|}{2^n(2^n-1)}$$

permutations in V_n that map x_1 to y_1 and x_2 to y_2.

4.3.2 Universal One-Way Hash Functions (UOWHF)

The concept of a UOWHF was introduced by M. Naor and M. Yung [233]. They suggested the definition and gave a provably secure construction based on a strongly universal hash function and a one-way permutation. They use the concept to construct a provably secure digital signature scheme based on a one-way injection, which is a less stringent requirement than a trapdoor one-way function which was necessary for the scheme proposed in [15].

4.3.2.1 Definition

A formal definition of a UOWHF can be given as follows.

Definition 4.14 *A* **Universal One-Way Hash Function** *family $\mathcal{H}$ is a function family and a polynomially bounded function $l : \mathbb{N} \longrightarrow \mathbb{N}$.*
A member of H_n is a function $f : \Sigma^n \longrightarrow \Sigma^{l(n)}$.

A collision string finder F is a probabilistic polynomial time algorithm that on input n outputs an initial value $x \in \Sigma^n$, and then given a random $h \in H_n$ outputs either "?" or an $x' \in \Sigma^n$ such that $h(x') = h(x)$.

$\mathcal{H}$ must satisfy the following condition:
for all collision string finders F, for all polynomials Q, and for sufficiently large n holds that

$$\Pr\{F(h, x) \neq \text{"?"}\} < \frac{1}{Q(n)},$$

where the probability is taken over all $h \in H_n$ and the random choices of F.

The collision string finder first selects an input string x and subsequently gets a randomly selected hash function. The philosophy behind a UOWHF is that if first the input is selected and subsequently the hash function, it does not help an opponent to find collisions for the hash function. Collisions are only useful if first the function is fixed and subsequently one can search for two colliding inputs.

This definition was generalized in [341], where a $UOWHF$ is defined as a three party game with an initial string supplier S, a hash function instance generator G and a collision string finder F. Here S is an oracle with unlimited computing power, and G and F are probabilistic polynomial time algorithms. The game consists of three moves:

1. S outputs an initial string $x \in \Sigma^n$ and sends it to both G and F.
2. G chooses an $h \in_R H_n$ independently of x and sends it to F.
3. F outputs either "?" or an $x' \in \Sigma^n$ such that $h(x') = h(x)$.

F wins the game if its output is not equal to "?". The input x is selected by S according to a certain distribution. In the most general case this is the collection of all ensembles with length n. If a different ensemble is introduced, a different definition is obtained. In the original definition of M. Naor and M. Yung the initial string supplier and the collision string finder were the same algorithm, which imposes the unnecessary restriction that x should be selected according to all polynomially samplable ensembles (the collision string finder has to be a polynomial time algorithm). The construction by M. Naor and M. Yung also satisfies this more general definition. On the other hand their definition is less complicated: in fact it does not really make sense for S to send x to G, as G chooses subsequently h independent from x. In [341, 342] the hierarchy between different types of UOWHF has been studied.

First two general methods will be described to construct a UOWHF. Subsequently some specific schemes will be given, namely the scheme of Naor and Yung based on

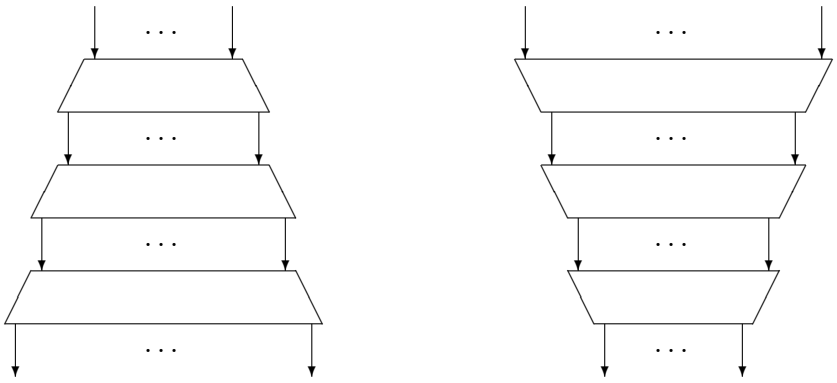

Figure 4.1: Method 1 for serial extending and compressing (M. Naor and M. Yung).

strongly universal hash functions, the first scheme of Zheng, Matsumoto, and Imai, the schemes by De Santis and Yung, the scheme by Rompel, the second scheme of Zheng, Matsumoto, and Imai, and the scheme of Sadeghiyan and Pieprzyk.

4.3.2.2 General construction methods

Two general methods have been described to construct a UOWHF from a more simple UOWHF. It has been pointed out by Y. Zheng, T. Matsumoto, and H. Imai [341] that these two general constructions are related to similar constructions for pseudo-random string generators. This can be explained by the duality between a PSG and a UOWHF: the first extends an input of fixed length to a polynomial size result, while the second compresses an input string of polynomial length into a string of fixed length. The construction of a UOWHF by M. Naor and M. Yung is related to the construction of a PSG by repeatedly applying an extension operation to the input string (cf. figure 4.1), while the construction of I. Damgård is the dual of the PSG scheme due to Boppana and Hirschfeld (according to [7]) (cf. figure 4.2).

The basic idea behind the construction by Naor and Yung is that the composition of two or more UOWHF families is again a UOWHF family. In fact a UOWHF was defined such that this property would hold.

Definition 4.15 *Let $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_l$ be families of functions such that for all i and for all $h_i \in \mathcal{H}_i$ (with $n_{i-1} < n_i$), $h_i : \Sigma^{n_i} \longrightarrow \Sigma^{n_{i-1}}$. The* **$l$-composition** *of $\mathcal{H}_1, \mathcal{H}_2, \ldots \mathcal{H}_l$ is the multiset $\mathcal{H}^{(l)} = \{h \mid h = h_1 \circ h_2 \circ \cdots \circ h_l\}$.*

Lemma 4.1 (composition lemma) *Let $\mathcal{H}^{(l)}$ be an l-composition. If there exists a collision string finder F which, when given an initial value x and a uniformly random $h \in \mathcal{H}^{(l)}$, produces an output $\neq$ "?" with probability $\Pr\{F(h,x) \neq \text{"?"}\} > \epsilon$, then there exists an i with $1 \leq i \leq l$ and an algorithm F' such that*

- *F' produces an initial value $x_i \in \Sigma^{n_i}$.*

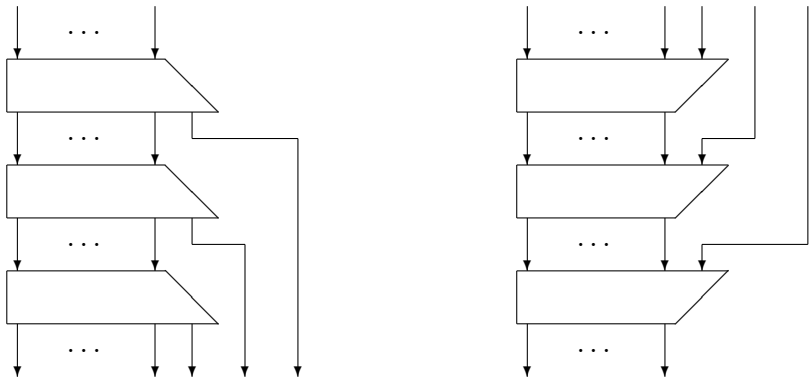

Figure 4.2: Method 2 for serial extending and compressing (I. Damgård).

- *Then on input* $h_i \in \mathcal{H}_i$ *tries to find a* x_i' *such that* $h_i(x_i) = h_i(x_i')$.
- $\Pr\{F'(h_i, x_i) \neq \text{“?”}\} > \epsilon/l$, *where the probabilities are taken over* $h_i \in \mathcal{H}_i$ *and the random choices of* F'.
- *The running time of* F' *is polynomially related to that of* F.

Theorem 4.2 (composition theorem) *Let* $\{n_{0_i}\}$, $\{n_{1_i}\}$, $\{n_{2_i}\}$, ... *be a sequence of increasing sequences, let* $\mathcal{U}_1, \mathcal{U}_2, \ldots$ *be a sequence of families of UOWHF such that* $\mathcal{U}_i = \{H_{i,m}\}_{m=1}^{\infty}$, *where* $\forall h \in H_{i,m}$; $h : \Sigma^{n_{i_m}} \longrightarrow \Sigma^{n_{i-1_m}}$, *let* $l : \mathbb{N} \longrightarrow \mathbb{N}$ *be a polynomial time computable function such that* $Q(n_{0_m}) > n_{l(m)_m}$ *for some polynomial* Q. *Let* $\mathcal{H}_m^{(l(m))}$ *be the* $l(m)$*-composition of* $\mathcal{H}_{1,m}$, $\mathcal{H}_{2,m}$, ..., $\mathcal{H}_{l(m),m}$, *and let* $\mathcal{U} = \{\mathcal{U}_i\}_{i=1}^{\infty}$.

Then $\mathcal{U}$ *is a family of UOWHF if the* $\mathcal{U}_i$ *are simultaneously hard, that is for every polynomial* P *and for every probabilistic polynomial time algorithm* F, *there is an* m_0 *such that* $\forall m > m_0$, F *can not succeed in finding collisions with probability* $1/P(n_{i_m})$ *for all* $i \geq 1$.

It is interesting to note that this composition theorem also holds for a CRHF.

The general construction by I. Damgård [66] as illustrated in figure 4.2 was intended for the design of a CRHF based on a Collision Resistant Function. Therefore it will be discussed in more detail in the corresponding section. It will become clear that if specific conditions are imposed on a compression function with a fixed size input, this general construction can also yield a UOWHF.

4.3.2.3 The scheme of Naor and Yung

As a result of the general construction method by M. Naor and M. Yung, the construction of a UOWHF is reduced to the construction of a UOWHF that compresses one bit. This is achieved with the composition of a strongly universal hash function with a one-way permutation.

Theorem 4.3 (Naor-Yung) *Let f be a one-way permutation on Σ^n.*
Define $H_n = \{h = g \circ f \mid g \in G_n\}$, where G_n is a strongly universal$_2$ family of hash functions from Σ^n to Σ^{n-1}, which has the weak collision accessibility property. Then $\mathcal{U} = \{H_n\}_{n=1}^{\infty}$ is a UOWHF family compressing n-bit input strings into $n-1$-bit output strings.

Note that the notations are slightly simplified here: if the formalism is followed completely, f should be replaced by a function family, where f_n is an instance of size n.

Based on the composition theorem one can obtain a UOWHF for which the size of domain and range are two increasing sequences that are polynomially related. This construction can be extended for the case that f is a one-way injection. The problem with this construction is that it is not very practical: the size of the hash function is $O(n_1^2)$ where n_1 is the size of the input. This can be improved with a factor $\log(n_1)$ by using a strongly universal$_{\log(n_1)}$ hash function. A more efficient approach will be to use a block hashing technique in a tree configuration: if the primitive function compresses $2t$ bits to t bits, the size of the function is $O(t^2 \log(n_1))$ bits. The number of blocks has to be polynomial in t to keep the proof valid. The original construction requires $O(n_1)$ applications of the one-way function, and the first improvement reduces this with a factor $\log(n_1)$. The block hashing technique needs $O(n_1)$ applications of the one-way function. Moreover for every message that has to be hashed a new hash function has to be generated and stored.

4.3.2.4 The first scheme of Zheng, Matsumoto, and Imai

This construction is based on the combination of a quasi injection, a pair-wise independent uniformizer, and a strongly universal hash function [342]. The basic motivation is to reduce the complexity assumptions.

Theorem 4.4 (Zheng-Matsumoto-Imai) *Let f be a quasi injection with input length n and output length $l(n)$. Define $H_n = \{h = g_n \circ v_n \circ f \mid g_n \in G_n, v \in V_n\}$, where G_n is a strongly universal family from $\Sigma^{l(n)}$ to Σ^{n-1}, and V_n is a pair-wise independent uniformizer. Then $\mathcal{U} = \{H_n\}_{n=1}^{\infty}$ is a UOWHF family compressing n-bit input strings into $n-1$-bit output strings.*

4.3.2.5 The schemes of De Santis and Yung

The major contributions of these schemes are the simplification of the original scheme of Naor and Yung, the improvement of the efficiency, and the reduction of the complexity assumptions. The simplification is based on the composition lemma for strongly universal hash functions. The construction is given for one-way injections [87].

Theorem 4.5 (De Santis-Yung) *Let f be a one-way injection with input length n and output length $l(n)$. Define $H_n = \left\{h = g_n \circ g_{n+1} \circ \cdots \circ g_{l(n)} \circ f \mid g_i \in G_i\right\}$, where G_i is a strongly universal family from Σ^i to Σ^{i-1}. Then $\mathcal{U} = \{H_n\}_{n=1}^{\infty}$ is a UOWHF family compressing n-bit input strings into $n-1$-bit output strings.*

The authors also give new constructions based on the following weaker complexity assumptions:

- The existence of a function with small expected preimage size or the property that the expected size of the preimage of an element in the range is small when an element in the domain is randomly chosen. An example of such a function is squaring modulo an RSA modulus.
- The existence of a function where for a given element in the range, an estimate with polynomial uncertainty on the size of the preimage set is easily computable. A particular case is a regular function, i.e., a function where every image of an n-bit input has the same number of preimages of length n. Other examples are the decoding of random linear codes and the subset sum (cf. section 7.2.8.1).

For the formal definitions and constructions the reader is referred to [87].

4.3.2.6 The scheme of Rompel

J. Rompel [284] describes an interesting scheme to turn any one-way function in a UOWHF. The idea is to construct a series of functions, each one closer to the goal of a UOWHF. First a function is constructed for which most siblings are easy to find, but a non-negligible fraction are provably hard to find. Next a function is constructed such that most siblings are provably hard to find. Subsequently a length increasing function is constructed for which it is almost always hard to find any sibling. Finally this is turned into a length-decreasing function with the same properties. The construction is only of theoretical interest: it can be shown that one-way functions are necessary and sufficient for secure digital signatures. The number of applications of f is polynomial in the size of the input, but the scheme is completely impractical as the exponents are larger than 30.

4.3.2.7 The second scheme of Zheng, Matsumoto, and Imai

The goal of this construction by Y. Zheng, T. Matsumoto, and H. Imai [341, 343] is to improve the efficiency rather than to reduce the complexity assumptions. This is the first construction that does not rely on universal hash functions, which implies that the description of the hash function is more compact. This construction processes the message in s-bit blocks. Therefore the $l(n)$-bit message x has to be divided into s-bit blocks denoted by $x_1, x_2, \ldots, x_t$, where $t = \lceil \frac{l(n)}{s} \rceil$ and $x_i \in \Sigma^s$ for each $1 \leq i \leq t$. (note that if $l(n)$ is not a multiple of s a padding procedure has to be specified, cf. section 2.4.1).

Theorem 4.6 (Zheng-Matsumoto-Imai) *Let f be a one-way permutation with input length $n + s$. One can assume w.l.o.g. that the hard core of f are the s rightmost bits of f. Let l be a polynomial with $l(n) > n$, and H_0 be an initial value $\in \Sigma^n$. Then define*

$$H_i = \mathrm{chop}_t\left(f\left(H_{i-1} \| x_{t-i+1}\right)\right) \qquad \text{for } i = 1, 2, \ldots, t\,.$$

Define $H_n = \{h \mid h(x) = H_t\}$. *Then* $\mathcal{U} = \{H_n\}_{n=1}^{\infty}$ *is a UOWHF family compressing* $l(n)$*-bit input strings into* n*-bit output strings.*

4.3.2.8 The scheme of Sadeghiyan and Pieprzyk

The goal of this scheme is to further increase the efficiency. Assume that one has a computing resource of k bits (cf. definition 4.10). The idea of this scheme is to construct from a one-way permutation with $k+1$ hard bits a one-way permutation for which each bit is a hard bit. This can be done through combination with permutations that have specific properties. The resulting permutation can then be used in an efficient construction of a UOWHF and of a PSG [289, 290].

Definition 4.16 *Let* v *be a permutation of* Σ^n. *Then* v *is a* $k+1$**-bit perfect permutation** *iff*

- v *is complete, or each output bit of* v *depends on all input bits.*
- $k+1$ *bits of* v *are pair-wise independent, or their correlation equals zero.*

Definition 4.17 *A* **strong one-way permutation** w *is a permutation with the maximal number of* $t = n - k^+$ *simultaneously hard bits.*

Clearly this is equivalent to stating that each bit of w is a hard bit of w and given $w(x)$ and any $t < n-k$ bits of the preimage x, it is hard to find the complete preimage x.

One can show (cf. section 4.2.3) that given a one-way function f one can find a function g such that $f = O(\log n)$ bits of $f \circ g$ are simultaneously hard. A simple example for g is the inner product with a random string. To turn a one-way permutation into a strong one-way permutation, it is sufficient to find a permutation for which it is hard, given $t < n-k$ bits of its input and $t' < n-k$ bits of its output, to guess any additional output bit. Such a permutation effectively hides any k bits of its output, hence it is called a hiding permutation.

Definition 4.18 *Let* h *be a permutation of* Σ^n. *Let* $i_1, \ldots i_t$ *and* $j_1 \ldots j_k$ *be functions from* $\mathbb{N}$ *to* $\mathbb{N}$, *where* $1 \leq i_l(n), j_l(n) \leq n$. *Then* h *is a* **hiding permutation** *iff for each probabilistic polynomial time algorithm* F, *for each* $t \leq n-k^+$, *for each polynomial* Q, *and for sufficiently large* n

$$\mid \Pr\left\{F(x_{i_t}, \ldots x_{i_1} \| y_{j_n}, \ldots y_{j_{k^+}}) = y_{j_k}, \ldots, y_{j_1}\right\} - \frac{1}{2^k} \mid < \frac{1}{Q(n)},$$

where $x \in_R \Sigma^n$, $y = h(x)$, *and* $k = O(\log n)$.

Any one-way permutation that acts on all its bits is a hiding permutation.

The following theorems, given in [289] and [290] respectively, show how to turn a one-way permutation into a strong one-way permutation. Both constructions use a hiding permutation. The first construction is based on a $k+1$-bit perfect permutation, while the second requires only a linear function over $GF(2^n)$.

Theorem 4.7 *Let f be a one-way permutation, let v be a $k+1$-bit perfect permutation, and let h be a hiding permutation. Define $w = f \circ v \circ h$. Then w is a strong permutation.*

Theorem 4.8 *Let f be a one-way permutation, let $g = px + q$ where $p, q \in_R GF(2^n)$, and let h be a hiding permutation. Define $w = f \circ g \circ h$. Then w is a strong permutation.*

The concept of a strong one-way permutation can be used to construct a UOWHF as follows.

Theorem 4.9 (Sadeghiyan-Pieprzyk) *Let f be a strong one-way permutation on Σ^n. Define $H_n = \{h = \mathrm{chop}_t \circ f\}$, where $t = n - k^+$. Then $\mathcal{U} = \{H_n\}_{n=1}^{\infty}$ is a UOWHF family compressing n-bit input strings into $n - t$-bit output strings.*

A variant of this method is to replace the chop function by a universal hash function. Parameterization of this UOWHF is possible through a parameterization of f or of the chopping function (other bits than the last t bits might be chopped). The efficiency of this method relies on the fact that it requires only one application of a strong one-way permutation for the processing of $n - k - 1$ bits, where a strong one-way permutation consists of 2 one-way permutations and a $k + 1$-bit perfect permutation or a linear function over $GF(2^n)$.

A final remark is that if a strong one-way permutation is used as primitive function f, the second scheme of Zheng, Matsumoto, and Imai reduces to the general construction method of Damgård [289].

4.3.3 Collision Resistant Hash Functions (CRHF)

The formal definition of a Collision Resistant Hash Function (CRHF) (or Collision Free Hash Function) has been introduced by I. Damgård [64, 65]. He also suggested the first provably secure constructions.

4.3.3.1 Definition

A formal definition of a CRHF can be given as follows.

Definition 4.19 *A* **Fixed Size Collision Resistant Hash Function** *family $\mathcal{H}'$ is a function family and a function $l : \mathbb{N} \longrightarrow \mathbb{N}$, such that $l(n) < n$.*
A member of H'_n is a function $h' : \Sigma^n \longrightarrow \Sigma^{l(n)}$.

A collision string finder F is a probabilistic polynomial time algorithm that on input n and a function $h \in_R H_n$ outputs either "?" or a pair $x, x' \in \Sigma^n$ with $x' \neq x$ such that $h(x') = h(x)$.

$\mathcal{H}'$ must satisfy the following condition:
for all collision string finders F, for all polynomials Q, and for sufficiently large n holds that

$$\Pr\{F(h) \neq \text{"?"}\} < \frac{1}{Q(n)},$$

where the probability is taken over all $h \in H_n$ and the random choices of F.

Definition 4.20 *A* **Collision Resistant Hash Function** *family $\mathcal{H}$ is a function family and a polynomially bounded function $l : \mathbb{N} \longrightarrow \mathbb{N}$.*
A member of H_n is a function $h : \Sigma^* \longrightarrow \Sigma^{l(n)}$.

$\mathcal{H}$ must satisfy the following condition:
for all collision string finders F, for all polynomials Q, and for sufficiently large n holds that

$$\Pr\{F(h) \neq \text{“?”}\} < \frac{1}{Q(n)},$$

where the probability is taken over all $h \in H_n$ and the random choices of F.

The main difference between definition 4.19 and definition 4.20 is that the second one imposes no restrictions on the lengths of the inputs of the functions. Of course a polynomial time algorithm can only hash messages of polynomial length.

The practical advantage of a CRHF over a UOWHF is that in case of a CRHF one is not forced to generated a new hash function for every input. A CRHF is however harder to design. From the definition of a CRHF it follows that it is also a UOWHF. A more difficult question is whether a CRHF is also a one-way according to definition 4.8. The answer is rather complicated, but a lemma of I. Damgård [66] that was slightly corrected by J.K. Gibson [119], gives some orientation:

Lemma 4.2 (Damgård) *Let $\mathcal{H}'$ be a fixed size collision resistant function family, and let h be an instance of size n. Let $E_{h'}$ be the probability distribution on $\Sigma^{l(n)}$ generated by selecting $x \in_R \Sigma^n$ and outputting $h'(x)$.*

Assume that for all but an exponentially small fraction of $x \in \Sigma^n$ there exists a $x' \in \Sigma^n$ with $h'(x) = h'(x')$.

Then no algorithm inverting h' on images selected according to $E_{h'}$ succeeds with probability larger than $1/2 + 1/Q(n)$ for any polynomial Q.

If $E_{h'}$ is the uniform distribution over the image of h' or if $n - l(n)$ is $O(n)$, then no inversion algorithm succeeds with probability larger than $1/Q(n)$, for any polynomial Q.

Note that in cryptographic applications an opponent will not select an $x \in_R \Sigma^n$ (as required by the definition of a one-way function) but an $h'(x) \in_R R_n$: the property that can be proven is not really the property that one would like to have. However, the selection of an $x \in_R \Sigma^n$ will result in a uniform distribution for $h'(x)$ if h' is a t to 1 mapping for constant t (which is the case for a fixed size one-way function, but not necessarily for a hash function), or if $n - l(n)$ is $O(n)$. Another solution is to assume that $E_{h'}$ is the uniform distribution over the image of h'.

A general construction is based on a fixed size CRHF. It is believed that the existence of one-way functions is sufficient for the existence of CRHF's, but for the time being a proof is lacking. Currently reductions have been given to claw resistant functions, distinction resistant permutations, and claw-resistant pseudo-permutations. Note that in the original papers these were called claw free functions, distinction intractible permutations, and claw free pseudo-permutations. In order to remain consistent, the names had to be changed.

4.3.3.2 Fixed size CRHF

First it is noted that the composition theorem for a UOWHF (theorem 4.2) also holds for a CRHF. One can however also construct of a CRHF based on a fixed size CRHF family [66].

Theorem 4.10 *Let $\mathcal{H}'$ be a fixed size CRHF family mapping n bits to $l(n)$ bits. Then there exists a CRHF family $\mathcal{H}$ mapping strings of arbitrary length (polynomial in n) to $l(n)$ bit strings.*

Because of its practical importance, the construction will be described here. To simplify notations $l(n)$ will be written as l. Two cases have to be distinguished:

$n - l > 1$**:** split the message into t blocks of size $n-l-1$ bits and apply an unambiguous padding rule (cf. section 2.4.1). The sequence H_i is then defined by:

$$\begin{aligned} H_1 &= h'(0^{l+1} \parallel x_1) \\ H_i &= h'(H_{i-1} \parallel 1 \parallel x_i) \qquad \text{for } i = 2, 3, \ldots t\,. \end{aligned}$$

The hashcode $h(x)$ is equal to H_t.

$n - l = 1$**:** here the message is processed bit by bit. First, select uniformly an l-bit string H_0. The sequence H_i is then defined by:

$$H_i = h'(H_{i-1} \parallel x_i) \qquad \text{for } i = 1, 2, 3, \ldots t\,.$$

The hashcode $h(x)$ is equal to H_t.

The second version will also work if $E_{h'}$ is the uniform distribution over the image of h' or if $n - l(n)$ is $O(n)$. It is slightly more efficient as it allows hashing an additional bit per application of h'.

4.3.3.3 Claw resistant permutations

The concept of a function family has to be generalized to allow for an elegant definition of claw resistant permutations [65]. It will only be used in section 4.3.3.

Definition 4.21 *A* **function family** *$\mathcal{F}$ is an infinite family of finite sets $\{F_n\}_{n=1}^{\infty}$, where F_n is the set of instances of size n. An instance $f \in F_n$ is a tuple,*

$$S = (f_0, \ldots f_{r_n-1}, D^0, \ldots, D^{r_n-1}, R)\,,$$

where $\forall i,\ 0 \le i \le r_n - 1$, f_i is a function: $f_i : D^i \longrightarrow R$, and $\bigcup_{i=0}^{r_n-1} \mathrm{Im}(f_i) = R$. Here r_n is called the set size of $\mathcal{F}$. The following requirements have to be imposed:

- *r_n is polynomially bounded as function of n.*
- *F_n is accessible, or there is a polynomial time algorithm, which on input n outputs an instance chosen uniformly from F_n.*

- *D^i is samplable, or there is a polynomial time algorithm, which on input S and i selects an element uniformly from D^i.*
- *f_i is polynomial time computable, or given S, i, and $x \in D^i$ there is a probabilistic polynomial time algorithm (polynomial in i and in $|x|$) that computes $f_i(x)$.*

A family of claw resistant functions can now be defined as follows:

Definition 4.22 *A* **claw resistant** *function family $\mathcal{C}$ is a function family with the property that for any instance $S = (f_0, \ldots f_{r_n-1}, D^0, \ldots, D^{r_n-1}, R)$, f_i is a t to 1 mapping for some constant t and $\mathrm{Im}(f_i) = R$, $\forall\, 0 \leq i \leq r_n - 1$.*

A claw finder is a probabilistic polynomial time algorithm F that on input S outputs either "?" or a tuple (x, x', i, j) such that $x \in D^i$, $x' \in D^j$, $0 \leq i, j \leq r_n - 1$, $i \neq j$ and $f_i(x) = f_j(x')$.
$\mathcal{C}$ must satisfy the following condition:
for all claw finders F, for all polynomials Q, and for sufficiently large n holds that

$$\Pr\{F(S) \neq \text{“?”}\} < \frac{1}{Q(n)}\,,$$

where the probability is taken over all $f \in F_n$ and the random choices of F.

If $D^0 = D^1 = \cdots = D^{r_n-1} = R$, then $\mathcal{C}$ is called a family of claw resistant permutations.

Let Σ_r be an alphabet with cardinality r. The n-bit input block x is then transformed into a prefix free encoding $\overline{x}$ over the alphabet Σ_r. Note that an efficient algorithm exists to transform x to $\overline{x}$ such that the length of $\overline{x}$ is linear in the length of x.

Theorem 4.11 *Let $\mathcal{C}$ be a family of claw resistant permutations with set size r_n, and with domain the set of all finite words over Σ_{r_n}. An instance of size n of h is defined by*

$$h(x) = f_{\overline{x}}(I)\,,$$

where $I \in D$ and $f_{\overline{x}}(I)$ is defined as $f_{x_1}(f_{x_2}(\cdots f_{x_t}(I)\cdots))$, with $\overline{x} = x_1, x_2, \ldots, x_t$. Then $\mathcal{H}$ is a CRHF family.

The previous theorem reduces the existence of collision resistant hash functions to the existence of claw resistant permutations. Several constructions for claw resistant permutations have been proposed:

- Based on one-way group homomorphisms: the basic idea is, given the one-way homomorphism f and a set of a_i selected uniformly from D, to define the functions f_i as

  $$f_i(x) = a_i \cdot f(x)\,.$$

  Examples of one-way group homomorphisms can be constructed based on the hardness of modular exponentiation and the discrete logarithm problem. Note however that this is an *indirect* reduction in the sense that finding a claw does not prove that the one-way function can be inverted in general.

- Based on the difficulty of factoring the product of two large primes. Several schemes have been proposed and will be discussed in more detail in chapter 6. For these constructions a *direct* reduction between the assumption and the claw resistance can be proved.
- Based on a one-way function that permutes the cosets of a subgroup of its domain [242]. Also in this case more concrete schemes will be discussed in chapter 6.
- Based on computing graph isomorphisms [30].

However, it remains an open problem whether claw resistant permutations can be constructed based on one-way functions.

I. Damgård gives in his PhD thesis [65] a heuristic argument that shows that the two flavors of one-way property are equivalent for a CRHF based on a claw resistant permutations.

4.3.3.4 Distinction resistant permutations

The concept of distinction resistant permutations was introduced by Y. Zheng, T. Matsumoto, and H. Imai [341, 343]. It is slightly more general than the concept of claw resistant permutations: it should be hard to find two inputs for which the output differs at a particular position. To simplify notation, the definition is less formal than definition 4.22.

Definition 4.23 *A* **distinction resistant permutation** *family $\mathcal{W}$ is a function family. A member of W_n is a function $w : \Sigma^n \longrightarrow \Sigma^n$.*

A near string finder algorithm F is a probabilistic polynomial time machine that on input w outputs either "?" or a pair $x, x' \in \Sigma^n$ such that $w(x') = w(x) \oplus e_{i(n)}$. Here e_i denotes the ith unit vector $[00\ldots010\ldots00]$.
$\mathcal{W}$ must satisfy the following condition:
for all near string finders F, for all polynomials Q, and for sufficiently large n holds that

$$\Pr\{F(w) \neq \text{“?”}\} < \frac{1}{Q(n)}\,,$$

where the probability is taken over all $w \in W_n$ and the random choices of F.

If the one-way permutation in the second scheme of Y. Zheng et al. is replaced with a distinction resistant permutation, the scheme yields a CRHF:

Theorem 4.12 *Let w be a distinction resistant permutation with input length $n+1$. One can assume w.l.o.g. that the rightmost bit of w is a hard bit of w. Let l be a polynomial with $l(n) > n$, and H_0 be an initial value $\in \Sigma^n$. Then define*

$$H_i = \mathrm{chop}_1(w\left(H_{i-1} \| x_{l(n)-i}\right)) \qquad \text{for } i = 1, 2, \ldots, l(n)\,.$$

Define $H_n = \left\{h \mid h(x) = H_{l(n)}\right\}$. Then $\mathcal{H} = \{H_n\}_{n=1}^{\infty}$ is a CRHF family compressing $l(n)$-bit input strings into $n-1$-bit output strings.

It it easy to show that the existence of distinction resistant permutations implies the existence of a claw resistant pair of permutations. It is an open problem whether it is possible to construct a distinction resistant permutation based on a claw resistant pair of permutations.

4.3.3.5 Claw resistant pseudo-permutations

This concept was introduced by A. Russell [288]. A pseudo-permutation is a function that is computationally indistinguishable from a permutation: it should be hard to find a witness of non-injectivity or a collapse for f, i.e., a pair (x, x') with $x \neq x'$ such that $p(x) = p(x')$.

Definition 4.24 *A* **pseudo-permutation** *family $\mathcal{P}$ is a function family. A member of P_n is a function $p : \Sigma^n \longrightarrow \Sigma^n$.*

A collapse finder algorithm F is a probabilistic polynomial time machine that outputs either "?" or a pair $x, x' \in \Sigma^n$ with $x \neq x'$ such that $p(x') = p(x)$.
$\mathcal{P}$ must satisfy the following condition:
for all collapse finders F, for all polynomials Q, and for sufficiently large n holds that

$$\Pr\{F(p) \neq \text{"?"}\} < \frac{1}{Q(n)},$$

where the probability is taken over all $p \in P_n$ and the random choices of F.

One can now define a function family of claw resistant pseudo-permutations. As pointed out in [288], the two aspects of these functions balance each other: if the pseudo-permutations are close to permutations, there will exist more claws.

The main result in [288] is then the following theorem:

Theorem 4.13 *There exists a collision resistant function family iff there exists a claw resistant pseudo-permutation function family.*

4.3.4 Sibling resistant functions (SRF)

The concept of a SRF was introduced by Y. Zheng, T. Hardjono, and J. Pieprzyk [344]. It is a further generalization of a UOWHF.

4.3.4.1 Definition

Informally a Sibling Resistant Function (SRF) is a function family for which it easy to find a function s under which k given strings x_i collide, but for which it is hard to find an x' that collides under s with the first k strings.

Definition 4.25 *Let k be a fixed integer. A $(k, 1)$* **Sibling Resistant Function** *family $\mathcal{S}$ is a function family that has the weak collision accessibility property, and a function $l : \mathbb{N} \longrightarrow \mathbb{N}$, such that $l(n) < n$. A member of S_n is a function $s : \Sigma^n \longrightarrow \Sigma^{l(n)}$.*

A sibling finder F is a probabilistic polynomial time algorithm that on input $X = \{x_1, x_2, \ldots, x_k\}$ a set of k initial strings $\in \Sigma^n$ and a function $s \in S_n$ that maps the x_i to the same string, outputs either "?" or an $x' \in \Sigma^n$ such that $s(x') = s(x_i)$.

$\mathcal{S}$ must satisfy the following condition:
for all sibling finders F, for all polynomials Q, and for sufficiently large n holds that

$$\Pr\{F(s, X) \neq \text{"?"}\} < \frac{1}{Q(n)},$$

where s is chosen randomly and uniformly from $S_n^X \subset S_n$, the set of all functions in S_n that map $X = \{x_1, x_2, \ldots, x_k\}$ to the same string in Σ^n, and where the probability is taken over S_n^X and the random choices of F.

A SRF is a generalization of a UOWHF, as a UOWHF is a $(1,1)$-SRF. Note that a $(k,1)$-SRF is also a $(k',1)$-SRF with $k' < k$.

A possible application of a SRF is to authenticate k pieces of information at the same time: this could be a selection of k computer programs. It is then possible to authenticate these programs through a single hashcode.

4.3.4.2 Construction

The following theorem [344] shows that a $(1,1)$-SRF can be transformed into a $(2^s - 1, 1)$-SRF for any $s = O(\log n)$.

Theorem 4.14 *Let $l_1(n)$, $l_2(n)$, and $l_3(n)$ be polynomials with $l_2(n) - l_1(n) = O(\log n)$, and let $k = 2^{l_2(n)-l_1(n)} - 1$. Define $S_n = \{s = u \circ h \mid h \in H_n, u \in U_n\}$, where $\mathcal{H}$ is a $(1,1)$-SRF family from $\Sigma^{l_1(n)}$ to $\Sigma^{l_2(n)}$, and $\mathcal{U}$ is a universal$_{k+1}$ hash function family from $\Sigma^{l_2(n)}$ to $\Sigma^{l_3(n)}$, that has the weak collision accessibility property. Then $\mathcal{S} = \{S_n\}_{n=1}^{\infty}$ is a $(k,1)$-SRF from $\Sigma^{l_1(n)}$ to $\Sigma^{l_3(n)}$.*

Combined with the result of J. Rompel, described in section 4.3.2.6, it follows that a $(k,1)$-SRF can be constructed from any one-way function.

4.3.5 Perfect Authentication codes

In chapter 3 it has been shown how unconditionally secure Cartesian authentication codes can be transformed into complexity theoretic constructions based on pseudo-random string generators. For authentication codes with secrecy, direct complexity theoretic constructions have been developed based on the work of M. Luby and C. Rackoff [198] on randomizers. In [340] it was shown how these randomizers can be used to construct provably secure block ciphers. If F_n is defined as the set of all functions from Σ^n to Σ^n, then for a function $f \in F_n$, the permutation $D \in F_{2n}$ is constructed as follows:

$$D_f(R, L) = (R, L \oplus f(R)),$$

where $R, L \in \Sigma^n$. For a sequence of functions $f_1, f_2, \ldots f_i \in F_n$, the permutation $\psi(f_1, f_2, \ldots, f_i)$ is defined as

$$\psi(f_1, f_2, \ldots, f_i) = D_{f_i} \circ \cdots \circ D_{f_2} \circ D_{f_1} .$$

It was shown in [198] that if three function f, g, h are selected randomly and uniformly from F_n, that $\psi(f, g, h)$ can not be distinguished from a random permutation, hence it is called an L-R randomizer. Note however that the permutation $\psi(f, g, h)$ has no polynomial description, and hence it is not polynomial time computable. A pseudo-random permutation can be obtained if f, g, h are pseudo-random functions; in that case ψ is called an L-R module.

J. Pieprzyk and R. Safavi-Naini show in [248] that if redundancy is added to the message before encryption with one or more L-R randomizers, one obtains a perfect authentication code. This implies that these codes are secure even if the opponent observes a large number of plaintext-ciphertext pairs. Depending on the redundancy, the following results are obtained under the assumption that $f, g, h \in_R F_n$ and g^* is a random permutation $\in F_n$:

- If the redundancy is secret and selected independently for each message: $\psi = \psi_1(f, g^*, h)$.
- If the redundancy is a publicly known fixed quantity that is used as the L-input: $\psi = \psi_1(f, g, h) \circ \psi_2(h, g^*, f)$.
- If the redundancy is a publicly known fixed quantity that is used as the R-input: $\psi = \psi_1(f, g, h) \circ \psi_2(h, g^*, f) \circ \psi_3(h, g^*, f)$.

In order to make the authentication code polynomial time computable, one should replace f, g, h and g^* by pseudo-random functions and permutations respectively. In that case the quality of the authentication codes will depend on the quality of the pseudo-random functions. The disadvantage of this method is that the size of the redundancy is at least twice the size of the message.

A second proposal by R. Safavi-Naini [292] consists of replacing the function f in the Luby-Rackoff construction with a generalized function f' or an expansion, which means that the range is larger than the domain. In this case it can be shown that only three rounds (or a single L-R randomizer) are sufficient to obtain a perfect authentication code. This construction has the advantage that it allows for more redundancy to be added to the information. Based on this construction, the author also suggests a new design principle for practical authentication systems, namely the combination of confusion, expansion, and diffusion. It should however be noted that practical systems rather perform a compression than an expansion.

4.4 Conclusion

The main contribution of the complexity theoretic approach is that it yields good definitions of concepts that would otherwise remain vague. A second advantage is that

this approach yields provably secure constructions based on the hardness of specific problems like factoring an RSA modulus or the discrete logarithm problem and even on more general assumptions like the existence of one-way functions. Most constructions in this chapter are completely impractical, and it is generally not possible to derive practical schemes from these constructions. However, in some cases one can obtain interesting design principles for more practical schemes.

Chapter 5

Hash Functions Based on Block Ciphers

All cases are unique and very similar to others. *T.S. Elliot*

5.1 Introduction

Two arguments can be indicated for designers of cryptographically secure hash functions to base their schemes on existing encryption algorithms. The first argument is the minimization of the design and implementation effort: hash functions and block ciphers that are both efficient and secure are hard to design, and many examples to support this view can be found in this thesis. Moreover, existing software and hardware implementations can be reused, which will decrease the cost. The major advantage however is that the trust in existing encryption algorithms can be transferred to a hash function. It is impossible to express such an advantage in economical terms, but it certainly has an impact on the selection of a hash function. It is important to note that for the time being significantly more research has been spent on the design of secure encryption algorithms compared to the effort to design hash functions. It is also not obvious at all that the limited number of design principles for encryption algorithms are also valid for hash functions. The main disadvantage of this approach is that dedicated hash functions are likely to be more efficient. One also has to take into account that in some countries export restrictions apply to encryption algorithms but not to hash functions. Finally note that block ciphers may exhibit some weaknesses that are only important if they are used in a hashing mode (cf. section 2.5.4).

In the first part of this chapter, some older proposals will be reviewed that combine encryption with redundancy to protect the integrity. These proposals are not based on a hash function, but they gave birth to the idea of using hash functions in a cryptographic context. In the next parts existing proposals for MDC's and MAC's

will be reviewed. This review comprises the description in a uniform and compact notation, a classification of known attacks, a new evaluation of the security, and in some cases the description of a new attack. In case of an MDC a distinction will be made between hash functions where the size of the hashcode equals the block length of the underlying block cipher and hash functions where the size of the hashcode is twice this length. This is motivated by the fact that most proposed block ciphers have a block length of only 64 bits, and hence an MDC with a result twice the block length is necessary to obtain a CRHF. Other proposals are based on a block cipher with a large key and on a block cipher with a fixed key. Several MAC proposals are reviewed, and a generalized treatment is presented of the interaction between encryption and MAC calculation if the same key is used for both operations.

The most important new contribution of this chapter is the synthetic approach for the case where the size of the hashcode equals the block length of the underlying block cipher: all existing proposals have been generalized and the secure schemes have been classified. This result was partially presented in [260]. A second contribution is the proposal of three new schemes together with a security evaluation. A third contribution is the application of differential cryptanalysis to hash functions based on block ciphers. A fourth contribution is a generalized treatment of the addition schemes and of the interaction between MAC calculation and encryption. Finally seven new attacks are presented on schemes proposed in the literature.

For a hash function based on a (non-randomized) block cipher, the following notations have to be fixed. The encryption operation with the underlying block cipher will be written as $C = E(K, P)$, where P denotes the plaintext, C the ciphertext, and K the key. The corresponding decryption operation will be denoted with $P = D(K, C)$. If the emphasis lies on the protection of secrecy and authenticity, the plaintext will also be denoted with X. The size of plaintext and ciphertext in bits is n, while the size of the key is k. The description will follow the general model that has been established in section 2.4.1.

The following definition characterizes the efficiency of a hash function based on a block cipher.

Definition 5.1 *The* **rate** R *of a hash function based on a block cipher is the number of encryptions to process a block of* n *bits.*

5.2 Authentication based on encryption and redundancy

Before the introduction of the concept of one-way hash functions, researchers were well aware of the fact that encryption with a block cipher does not offer a sufficient protection against active attacks.

The most natural approach to improve the authenticity protection is to add a simple form of redundancy to the plaintext. The consequence of this redundancy is that only a small part of the ciphertext space corresponds to genuine plaintext. It will however be shown that adding the sum of the plaintext blocks or a secret key is

generally not sufficient. But first two different approaches will be considered, namely randomized encryption and special modes of use.

5.2.1 Authentication based on randomized encryption

The main goal of randomized encryption techniques is to improve the security of privacy protecting schemes against chosen plaintext and dictionary attacks. A system where more than one ciphertext corresponds to a single plaintext is certainly harder to cryptanalyze. Just like in case of the addition of redundancy to the plaintext, this yields a ciphertext space that is larger than the plaintext space. However, it will become clear that this redundancy can generally not help to protect the authenticity of the plaintext. The discussions will be limited to the McEliece public-key cryptosystem and a new mode of use of a block cipher.

In 1978, R. McEliece proposed a new public-key cryptosystem based on algebraic coding theory [207]. The system makes use of a linear error-correcting code for which a fast decoding algorithm exists, namely a Goppa code. The idea is to hide the structure of the code by means of a transformation of the generator matrix. The transformed generator matrix becomes the public key and the trapdoor information is the structure of the Goppa code together with the transformation parameters. The security is based on the fact that the decoding problem for general linear codes is NP-complete [17]. For each irreducible polynomial $g(x)$ over $GF(2^m)$ of degree t, there exists a binary irreducible Goppa code of length $n = 2^m$ and dimension $k \geq n - mt$, capable of correcting any pattern of t or fewer errors. As it is a linear code, it can be described by its $k \times n$ generator matrix G. With the aid of a regular $k \times k$ matrix S and an $n \times n$ permutation matrix P, a new generator matrix G' is constructed that hides the structure of G:

$$G' = S \cdot G \cdot P\,.$$

The public key consists of G', and the matrices S and P together with $g(x)$ are the secret key. The new matrix G' is the generator matrix of another linear code, that is assumed to be difficult to decode if the trapdoor information is not known. The encryption operation consists of the multiplication of the k-bit message vector X by G' and the modulo 2 addition of an error vector e with Hamming weight t:

$$C = X \cdot G' \oplus e\,.$$

The first step of the decryption is the computation of $C \cdot P^{-1}$. Subsequently the decoding scheme makes it possible to recover $X \cdot S$ from

$$C \cdot P^{-1} = (X \cdot S \cdot G) \oplus (e \cdot P^{-1})\,.$$

The plaintext X is finally constructed by a multiplication with S^{-1}. At first sight one would expect that this scheme offers some protection against active attacks: the probability that a random ciphertext is decodable is equal to

$$2^{k-n}\sigma_n^t \qquad \text{with } \sigma_n^t = \sum_{i=0}^{t} \binom{n}{i},$$

which is about $2.5 \cdot 10^{-47}$, for the usual parameters $n = 10$ and $t = 39$. If the variant by F. Jorissen is used [165], a small part of the error correcting capability is retained at the cost of a decreased security level. This implies that a limited number of random modifications can still be corrected by the code. However, if an attacker wants to modify a particular plaintext bit, he simply has to add to the ciphertext the corresponding row of the public generator matrix G'.

A solution to this problem was given by R. Safavi-Naini and J. Seberry in [291]. The idea is to start from a set of k'-bit messages ($k' < k$), and to transform these into a set of k-bit messages with a systematic linear code that is kept secret:

$$X = X' \cdot G_s \,.$$

Note that this method is a MAC as it is based on a secret key. One can show that the probability of impersonation is equal to the combinatorial bound: $2^{k'}/2^k$ (cf. section 3.2.2). If a single cryptogram C_1 is observed, an attacker gets some information on the secret linear code. However, he is not able to exploit this information as he is not able to compute the corresponding X_1, which is equivalent to breaking the McEliece public-key cryptosystem. If two cryptograms are observed, an attacker has all information available to generate a new authentic message: it is sufficient to define $X_3 = X_1 \oplus X_2$ (the authentic messages form a vector space). However, this assumes that the attacker is able to obtain X_1 and X_2, which is again computationally infeasible. However, it is not mentioned in [291] that it is trivial to perform such a selective forgery with a known text attack. The protection of authenticity was also suggested in [189] for the secret key variant of McEliece by T. Rao and K. Nam [275]. However, it is very likely that this will increase the vulnerability of a scheme that has shown to be insecure against a chosen plaintext attack [315].

A second scheme that will be discussed is a new mode of use for a block cipher that was put forward in [188]. The main goal of the **"random code chaining" (RCC)** mode is to make it easier to modify part of the plaintext, while keeping the chaining of blocks and a limited error propagation. The disadvantage is the redundancy of 50%. The ciphertext corresponding to t $n/2$-bit plaintext blocks is equal to

$$C_i = E(K, X_i \oplus R_{i-1} \| R_i) \,,$$

where R_i ($1 \leq i \leq t$) are random $n/2$-bit blocks and R_0 is a function of K. The authors however wrongly claim that RCC makes the ciphertext resistant to authenticity threats: it is true that any modification to the ciphertext will yield random plaintext, but in spite of the redundancy no mechanism to verify the authenticity is present. The problem is that the authenticity of the random information can not be checked (by definition). The same applies for the less secure variations that are discussed in [188].

5.2.2 New modes of use

New modes apart from the 4 standard ones (ECB,CBC,CFB,OFB) (cf. appendix A) have been introduced to improve characteristics like error propagation and integrity protection.

A first mode of use was proposed in 1975 by H. Feistel [105] and was called "block chaining". The basic idea behind the scheme was already suggested by the same author in 1973 [104]. The idea is to append n' bits of the previous ciphertext block to the plaintext block of $n - n'$ bits, or:

$$C_i = E(K, \text{chop}_{n-n'}(C_{i-1}) \| X_i) \,,$$

where C_0 is a secret and time dependent IV, and the function chop_r drops the r least significant (or rightmost) bits of its argument (cf. chapter 4). It is suggested that $n' \leq n/2$, or the redundancy is at most 50%. The designer claims that any modification will be detected with a probability of $1 - 2^{-n'}$. It is however clear that an active attacker will look for two ciphertext blocks with the same n' most significant bits. After he has collected $2^{n'/2+1}$ ciphertext blocks, his success probability is about 63%; this is in fact a birthday attack as described in section 2.5.1.3. If he has found a match, he is able to substitute the corresponding blocks without being detected. Observe that this attack does not require the knowledge of the plaintext blocks.

The mode proposed by S. Matyas and C. Meyer ([215], pp. 100–105) is called "plaintext-ciphertext block chaining" (PCBC):

$$C_i = E(K, X_i \oplus X_{i-1} \oplus C_{i-1}) \,.$$

Some redundancy is added to the plaintext under the form of an additional block X_{t+1}. This block allows for the computation of the 'hashcode' as the last ciphertext block, or $MDC = C_{t+1}$. The authors suggest that X_{t+1} can be either constant, or the initial value IV, or the first block X_1. For completeness it has to be specified that X_0 equals the all zero block. The proposers also discuss the security of this mode. It is stated correctly that if the ciphertext blocks C_1 through C_i are garbled, resulting in C'_1 through C'_i and X'_1 through X'_i, a correct MDC will be obtained if

$$C_i \oplus X_i = C'_i \oplus X'_i \,.$$

They conclude that it is infeasible to find a message $X' \neq X$ resulting in the correct value for C_{t+1}, as *"under normal conditions an opponent will not know the secret cipher key K, and without knowledge of this key it is computationally infeasible to compute the authentication code or to make systematic changes to the ciphertext that would escape detection"*. It will be shown that this is not correct: if the message contains more than n blocks and if the opponent knows the $t - n$ blocks X_i (with $n \leq i < t$) and n plaintext-ciphertext pairs, he can easily modify the message without affecting the authentication code. The equation to be solved can be rewritten as:

$$\bigoplus_{j=1}^{i} C'_j \oplus D(K, C'_j) = C_i \oplus X_i \oplus IV = T_i \,.$$

The attacker will split the n blocks $C'_j \oplus D(K, C'_j)$ in two groups of size $n/2$ and generate $2^{\frac{n}{2}} - 1$ possible linear combinations in every group. From section 2.5.1.3 it

follows that he is likely to find two values with a difference of T_i (note that he can add T_i to one of the two groups and look for a match).

A third mode of use was proposed in [163], namely "OFB with a Non-Linear Function" (OFBNLF):

$$C_i = E(K_i, X_i) \quad K_i = E(K, K_{i-1})\,.$$

It yields an infinite forward error extension in case of the deletion or insertion of a message block, while the modification of a single ciphertext block results in only one garbage plaintext block. It is clear that this approach depends on the internal redundancy of the message, as there is no mechanism provided to distinguish between authentic and garbage plaintext. Moreover, even if redundancy is present, the deletion of the last blocks would not be detected. This can only be thwarted if a block count field is introduced, but this prevents insertion or deletion of blocks for all other modes too.

Finally it is remarked that in case of stream ciphers one can also improve the protection against active attacks by selecting different modes. This can make it hard for an attacker to produce the ciphertext corresponding to a specific plaintext [164], but a receiver has to decide whether the plaintext is authentic based on its internal redundancy.

5.2.3 Addition schemes

A third type of schemes consist of adding some simple redundancy to the plaintext, namely the modulo 2 sum of the plaintext blocks. This construction can be considered as the first simple proposal for a 'hash function'. According to the general model it can be described as follows:

$$f = H_{i-1} \oplus X_i\,,$$

with $IV = 0$. The resulting MDC is appended to the plaintext, which is subsequently encrypted in CBC (Cipher Block Chaining), CFB (Cipher Feedback), OFB (Output Feedback), PCBC (Plaintext Ciphertext Block Chaining) mode, or OFBNLF (Output Feedback with Non-Linear Function) using a *secret* key. It is clear that this scheme can only be used if integrity protection is combined with privacy protection. Moreover its security will depend on the mode of use of the encryption algorithm.

The scheme (combined with the first three modes) was proposed by the U.S. National Bureau of Standards and [167] *"found its way into a number of publications (notably a draft of Federal standard 1026) and books [215], and enjoyed considerable appeal due to its simplicity"*. Because of this simplicity the security depends on the mode of the encryption algorithm, but all proposed modes have been shown to be totally insecure. The attacks that have been described have all been limited to the first three modes [6, 166, 167, 209]. Table 5.1 indicates which manipulations can be carried out at block level. In CFB, insertions are only undetected if the last ciphertext block and the IV are not affected. Also for the CFB mode, one has to remark that if the size of the feedback variable is maximal and equal to n, that permutations of

blocks will also be undetected. This was not observed in the extensive treatment of these schemes in [166, 167]. In case of the two special modes, PCBC and OFBNLF, substitutions can be carried out if a set of plaintext-ciphertext pairs is known for a given key. Of course insertion of one or more blocks can always be thwarted with a

mode	CBC	CFB	OFB	PCBC	OFBNLF
insertion	√	√			
permutation	√	√	√		
substitution			√	√	√

Table 5.1: Manipulations that are not detectable for different modes of operation.

block count field. Note that these manipulations require no access to the secret key. If the attacker has access to the key, these schemes can also be broken with a meet in the middle attack (cf. section 2.5.2.1), as will be discussed in section 5.3.1.1.

For the three conventional modes, only the attack on the CBC mode will be discussed. For more details the reader is referred to [167]. It is sufficient to observe that a manipulation of the ciphertext blocks C_1 through C_t goes undetected if the sum modulo 2 of the resulting plaintext blocks remains equal to H_{t+1}. This can be restated as follows:

$$H_{t+1} = IV \oplus \bigoplus_{i=1}^{n} C_i \ \oplus \ \bigoplus_{i=1}^{n} D(K, C_i)\,.$$

It can be seen that permutations of blocks are not detectable, and the same holds for an even number of insertions of a given block.

For the PCBC mode combined with an addition scheme, the attack described in the previous section can be extended as follows. The modulo 2 sum of all plaintext blocks can be written as

$$\bigoplus_{j=1}^{t-1} \left(C'_j \oplus D(K, C'_j)\right) \oplus IV \oplus D(K, C_t)\,.$$

The substitution leaves the first sum unchanged, and C_t is not modified by assumption, hence C_{t+1} will not be altered.

In case of OFBNLF, the modulo 2 sum of the plaintext blocks is equal to

$$\bigoplus_{j=1}^{t} D(K_i, C_i)\,.$$

If b plaintext-ciphertext pairs are known for a set of a keys K_i, a birthday attack can produce a message with a given MDC if $a \cdot b \simeq 2^{n/2}$. On the other hand, if the ciphertext corresponding to a chosen plaintext can be obtained, it is very easy to cancel any known changes to the sum. If the scheme is used with care, i.e., the starting value for the key is always different, it is very hard to obtain the required data. On the other

hand, it should be noted that it is a scheme with rate equal to 2. It will be shown that more efficient and more secure solutions are available that are based on a secure hash function.

Soon after the discovery of weaknesses a more secure generalization was proposed by S. Matyas and C. Meyer:

$$f = (H_{i-1} + X_i) \bmod 2^m \quad m > 1\,.$$

However, it has been pointed out [167] that not all weaknesses are avoided:

- CBC: inserting 2^i identical blocks results in a probability of detection of $1/2^{m-i}$.
- CFB with maximal size of feedback variables: inserting 2^i identical blocks results in a probability of detection of $1/2^{m-i}$ (again this was not noticed in [167]).
- OFB: several subtle manipulations are possible if $m \leq n$. Two bits that are at a distance of n and have a different value can be both changed. If their value is not known, the probability of not being detected equals $1/2$ if their position in the block is smaller than m and 1 else.

Some of these weaknesses can be avoided by selecting a sufficiently large value of m. However, this class of linear or near-linear schemes is still weak as was observed by M. Smid [167]. If part of the plaintext is known, it is rather easy to calculate the error term for the MDC. Subsequently the encrypted value of the new MDC is obtained using a chosen plaintext attack. An alternative is to insert the ciphertext corresponding to a correcting plaintext block.

Some proposals were made to replace the addition schemes by a simple CRC (Cyclic Redundancy Check). Although it is clear that there will not be a comparable interference with the common modes of block ciphers, we believe that this should not be recommended, as for these schemes it is very easy to modify the message without affecting the checksum. Hence the security of the combination of a CRC and an encryption algorithm should be checked for every algorithm, which implies that this can never be a generally recommended or standardized solution.

5.2.4 A simple MAC

The simplest MAC that one can imagine is used for message authentication in the military environment and has been described by G. Simmons [310]. The authentication key is stored into a sealed tamper resistant box and it is used only once. The MAC is simply equal to a secret key and is appended to the message. The security of the system depends on the fact that subsequently the extended message is encrypted with an algorithm that uses cipher or text feedback. Under the assumption that the encryption algorithm is secure, and that the encryption key is not compromised, the probability for an opponent to impersonate or substitute a message is 2^{-n}, if the size of the MAC is n bits. This system is not practical in an industrial environment, because a different key is required for every message. Moreover its security depends completely on the subsequent encryption. This is not the case if a MAC is used that satisfies the definition given in section 2.2.

5.3 Overview of MDC proposals

In the first class of proposals, the size of the hashcode is equal to the block length n of the underlying block cipher algorithm. At the time of their proposal, the requirements to be imposed on hash function were not as well understood as today. It is not the goal of this thesis to study the evolution of the concept but to indicate whether these construction meet the current requirements for a OWHF or a CRHF. It is clear from the definitions of chapter 2 that a construction where the size of the hashcode equals the block length will only yield a CRHF if the block length n is at least 128 bits. This is not the case for the currently proposed block ciphers like DES [8, 108], FEAL [225, 227, 304, 305], and LOKI [33, 34]. The designers of these scheme had the intention to produce a OWHF with the available block ciphers, and it should be carefully checked whether or not these proposals do or do not yield a CRHF if $n \geq 128$. A second class of proposals describes how to construct a CRHF with size of hashcode equal to $2n$ bits based on a block cipher with block length equal to n bits. A third class of schemes constructs a OWHF and a CRHF based on block ciphers with block length equal to 64 bits and key size twice as large. Examples of such ciphers are PES [181], IPES or IDEA [182, 184], and FEAL-NX [228]. Finally a new class of proposals is made where the key is chosen to be constant.

In order to avoid confusion, the attention of the reader is drawn to the fact that in this chapter n is used for the block length of the block cipher, while this symbol was used previously to denote the size of the hashcode. This is certainly no problem if both are equal, and if two different symbols are required, which is the case in section 5.3.4, the size of the hashcode will be denoted with h.

5.3.1 Size of hashcode equals block length

Three types of schemes can be distinguished: the first class is based on the simple modes of use, the second class uses a chaining of blocks that allow an attacker to go backwards in the chain, while the third class of schemes will not allow such an operation. Subsequently a synthetic approach will be discussed, that explores a large number of schemes and allows to identify a subset of these schemes that are secure.

5.3.1.1 Conventional modes of use

In fact these schemes were never proposed as such, but some confusion arose in some overview papers [230, 254] because of the attacks described in [6]. These attacks are in fact attacks on a MAC using these modes, where the attacker is privy to the key. However, it is of interest to understand why these schemes are insecure if they are used to generate an MDC.

The two modes that exhibit the required error propagation for the generation of an MDC are the CBC and the CFB mode. As the computation may not involve any secret quantities, it has to be assumed that both key and initial value are public.

$$CBC : f = E(K, H_{i-1} \oplus X_i)\,.$$

In case of the CBC mode, it is very easy to change all blocks but the first one, compute the new value H_1', and take $X_1 = D(K, H_1') \oplus IV$. This is a correcting block attack (cf. section 2.5.2.4). For the CFB the following description can be given:

$$CFB : f = E(K, H_{i-1}) \oplus X_i \,.$$

After encryption of the last block, the output H_t is encrypted with key K and the last n bits of ciphertext H_i are obtained. The last step is added to avoid a linear dependence of the MDC on the last plaintext block. If the feedback value is n bits long, the last n bits are simply $E(K, H_t)$, but if it is shorter the hash value will comprise more than one ciphertext block. Here a similar attack applies: modify all blocks but the first one, compute the new value H_1', and take $X_1 = E(K, IV) \oplus H_1'$. The limitation of these attacks is that at least one block of the second plaintext will be random. This can be solved by applying a meet in the middle attack (cf. section 2.5.2.1).

5.3.1.2 Invertible key chaining

This type of hash functions is based on the fact that a good block cipher can withstand a chosen plaintext attack: this implies that it is computationally infeasible to determine the key even if a large number of adaptively chosen plaintext and ciphertext pairs are known. For a larger class of schemes, the function f can be defined by

$$f = E(X_i \oplus s(H_{i-1}), H_{i-1}) \,,$$

with s a public function that maps the ciphertext space to the key space (in the case of DES a simple example is the discarding of the parity bits). The first proposals for s were the zero function [274] and the identity, attributed to W. Bitzer in [71, 86]. In the first case, it is easily seen that the scheme is vulnerable to a meet in the middle attack (cf. section 2.5.2.1), as was pointed out by R. Merkle.

However, R. Winternitz [327] has shown that this is also the case for s the identity function: the idea is to assume a value of $K = X_i \oplus H_{i-1}$. The plaintext resulting from the decryption of H_i with key K is equal to H_{i-1}, and X_i can subsequently be computed as $K \oplus H_{i-1}$. Finally, it was remarked that any choice of s results in the same attack [230].

Because all these schemes were considered to be insecure, the proposals were amended to thwart the attacks. It will however be shown that all these proposals are vulnerable to an extended meet in the middle attack. The first idea [70] was repeating the message p times, another one was repeating the whole scheme for p different initial values. The first suggestion is very weak, as the meet in the middle attack will still work if $H_{it} = IV$, for $1 \leq i \leq p$. However this implies that one can only produce a collision and not a preimage of a given value. In [56] it was explained how the meet in the middle attack can be extended to attack both suggestions for $p = 2$ and $p = 3$, while a further generalization in [123] shows that an attack on a p-fold iterated scheme would not require $O(2^{\frac{pn}{2}})$ but only $O(10^p \cdot 2^{\frac{n}{2}})$ operations. For $n = 64$ this is infeasible for $p \geq 8$, but this would mean that the scheme becomes very inefficient.

A third proposal (attributed to R. Merkle in [70]) is first encrypting the message in CBC or CFB mode (with a random non-secret key K and initial value IV) before applying the hash function. This implies a reduced performance: the rate equals 2. The idea is to thwart a meet in the middle attack by introducing a dependency between the blocks that enter the hash function. We were able to extend the meet in the middle attack to a constrained meet in the middle attack (cf. section 2.5.2.2) to take into account this extension.

- Generate a set of r messages for which the last ciphertext block of the CBC encryption with initial value IV and key K is equal to IV' ; this can be done easily with an appropriate selection of the last plaintext block (or with a meet in the middle attack).
- Generate a second set of r messages and encrypt these messages in CBC mode with initial value IV' and key K.
- As the two message parts are now independent, one can use the set of two 'encrypted' messages in a simple meet in the middle attack.

It should be remarked that for all previous schemes the fact that one can go backwards implies that it is trivial to produce a "pseudo-preimage" for a random initial value.

A fourth possibility suggested in the book of J. Seberry and J. Pieprzyk [299] is to introduce additional redundancy X_{t+1} in the message under the form of the modulo 2 sum of X_1 through X_t. However, a constrained meet in the middle attack can take into account the redundancy. It is sufficient to require that the sum of all first variations is equal to T_1 and the sum of all second variations is equal to T_2, with $X_{t+1} = T_1 \oplus T_2$.

Finally note that when DES is used as the underlying block cipher, every scheme of this type is vulnerable to an attack based on weak keys [327] (cf. section 2.5.4.2): for a weak key K_w DES is an involution which means that $E(K_w, E(K_w, X)) = X, \forall\ X$. Inserting twice a weak key as a message block will leave the hashcode unchanged in all the schemes (even the modulo 2 sum of the blocks remains the same). Note that in the proposal of R. Merkle the actual message block has to be computed by decryption of the weak keys [6]. This attack can be thwarted by putting the second and third key bits to 01 or 10: for all weak keys these bits are equal. As a consequence the rate increase from 1.14 to 1.19, and the security level is at most 54 bits.

5.3.1.3 Non-invertible key chaining

The next type of hash functions is also based on key chaining, but tries to avoid the backward recurrence. To simplify the notation, we define

$$E^{\oplus}(K, X) = E(K, X) \oplus X$$

and

$$E^{\oplus\oplus}(K, X) = E(K, X) \oplus X \oplus s'(K)\,.$$

Here s' is a public function that maps the key space to the ciphertext space.

One scheme is proposed by S. Matyas, C. Meyer, and J. Oseas in [203], and referred to as "Key Block Chaining" in [168]:

$$f = E^{\oplus}(s(H_{i-1}), X_i)\,,$$

with s a public function that maps the ciphertext space to the key space. A second scheme is attributed to D. Davies in [327, 328], and to C. Meyer in [73]. Therefore it is denoted with the Davies-Meyer scheme in [223, 269]:

$$f = E^{\oplus}(X_i, H_{i-1})\,.$$

Also in [203], interleaving of the first two schemes is proposed. A third scheme was proposed by the author and studied by K. Van Espen and J. Van Mieghem in [318]. It appeared independently in [226] as a mode of use for the N-hash algorithm. N-hash is a dedicated hash function that has a Feistel structure [104, 105] and is based on the same design principles as FEAL. It will be discussed in more detail in chapter 7. In 1990 the same scheme was proposed by Japan as a contribution to DP 10118-2, the ISO document that specifies a hash function based on a block cipher [161]. The function can be described as follows:

$$f = E^{\oplus\oplus}(H_{i-1}, X_i)\,.$$

Another variation was proposed as a mode of use for the block cipher LOKI [33]:

$$f = E^{\oplus}(X_i \oplus H_{i-1}, H_{i-1})\,.$$

The goal of adding H_{i-1} is to prevent the key from being weak if X_i is equal to the all zero or all one block. It was shown independently by E. Biham and A. Shamir [22] and by B. den Boer [82] that weaknesses of LOKI (cf. section 2.5.4) can be used to produce a second preimage (in fact 15 different preimages) in constant time. The security of these schemes will be discussed in more detail in the next section.

5.3.1.4 A synthetic approach

A natural question that arises when all the previous schemes are considered is the following: is it possible to find other schemes based on the same principles, or are the described proposals the only ones that are secure? To limit the number of possibilities, it will be assumed that the size of the key is equal to the block length of the algorithm. This is not true for DES, but it could be easily satisfied if the first step of the key scheduling is the discarding of the parity bits.

The block cipher has two inputs, namely the key input K and the plaintext input P, and one output C. One can select for the inputs one of the four values: X_i, H_{i-1}, $X_i \oplus H_{i-1}$ and a constant value V. It is also possible to modify with a feedforward FF the output C by addition modulo 2 of one of these four possibilities. This yields in total $4^3 = 64$ different schemes.

If the function $H_i = f(X_i, H_{i-1})$ is considered, five types of attacks have been identified, extending the classification of section 2.4.2:

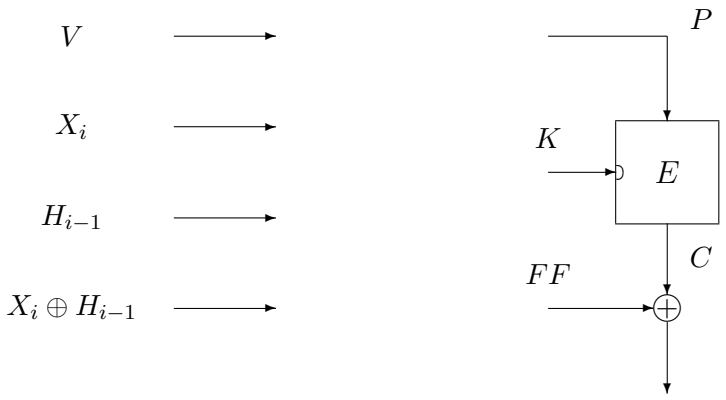

Figure 5.1: Configurations for an MDC where the size of the hashcode is equal to the block length. P, K, and FF can be chosen from the set $\{V, X_i, H_{i-1}, X_i \oplus H_{i-1}\}$.

Direct Attack (D): given H_{i-1} and H_i, it is easy to find X_i. All schemes that are vulnerable to a direct attack can in principle be used for encryption, where the encryption of X_i is given by block H_i. Of course the CBC and CFB mode belong to this class.

Permutation Attack (P): in this case H_i can be written as $H_{i-1} \oplus f'(X_i)$, where f' is a one-way function: X_i can not be recovered from H_i and H_{i-1}, but the hashcode is independent of the order of the message blocks, which means that a second preimage or collision can be found easily. Moreover one can also insert the same message block twice. These attacks are in fact trivial, as H_i depends only linearly on H_{i-1}.

Forward Attack (F): given H_{i-1}, H'_{i-1}, and X_i (note that this means that H_i is fixed), it is easy to find an X'_i such that $f(X'_i, H'_{i-1}) = f(X_i, H_{i-1}) = H_i$.

Backward Attack (B): given H_i, it is easy to find a pair (X_i, H_{i-1}) such that $f(X_i, H_{i-1}) = H_i$.

Fixed Point Attack (FP): find H_{i-1} and X_i such that $f(X_i, H_{i-1}) = H_{i-1}$. As was already pointed out, this attack is not very dangerous: if the OWHF is secure, it is hard to produce a message yielding this specific value H_{i-1} as chaining variable.

The order of these attacks has some importance: the possibility of a direct attack means that a forward and a backward attack are also feasible, but the converse does not hold. In case of a permutation attack, one can also apply a backward attack by first selecting X_i and subsequently calculating H_{i-1}. If a forward attack is possible, one can easily construct a second preimage for a given hashcode, but it is not necessarily easy to construct a preimage of a given element in the range. In case of a backward attack, a preimage (or a second preimage) can be found with a meet in the middle attack. It is easy to show that if both a forward and a backward or permutation attack

are possible, a direct attack is also feasible. Indeed, let us assume that a direct attack does not work. In that case, the feasibility of a forward attack implies that the three inputs are either constant or equal to $X_i \oplus H_{i-1}$. On the other hand, if one can go backwards, or if a permutation attack applies, one is able to find a pair (X_i, H_{i-1}) such that $f(X_i, H_{i-1}) = H_i$. In fact one will in this case only determine $X_i \oplus H_{i-1}$, and hence given H_{i-1}, one is able to find the corresponding X_i, which contradicts our assumption.

A table has been compiled that shows which attacks are possible for a feedforward of a constant (w.l.o.g. V is assumed to be equal to 0), X_i, H_i, and $X_i \oplus H_i$. The attacks are indicated with their first letter(s), while a "–" means that the function f is trivially weak as the result is independent of one of the inputs. If none of these five attacks applies, a $\surd$ is put in the corresponding entry. An attack on the scheme in table 5.2 with a constant key and feedforward and plaintext equal to $X_i \oplus H_{i-1}$ (denoted with a $(*)$ in the table) has also been discussed in [203].

		choice of P			
choice of FF	choice of K	X_i	H_{i-1}	$X_i \oplus H_{i-1}$	V
V	X_i	–	B (Rabin)	B	–
	H_{i-1}	D	–	D	–
	$X_i \oplus H_{i-1}$	B	B (Bitzer)	F	F
	V	–	–	D (CBC)	–
X_i	X_i	–	B	B	–
	H_{i-1}	$\surd$ (Matyas et al.)	D	$\surd$	D
	$X_i \oplus H_{i-1}$	FP	FP	B	B
	V	–	D (CFB)	B	–
H_{i-1}	X_i	P	FP (Davies-Meyer)	FP	P
	H_{i-1}	D	–	D	–
	$X_i \oplus H_{i-1}$	FP	FP (Brown et al.)	B	B
	V	D	–	D	–
$X_i \oplus H_{i-1}$	X_i	P	FP	FP	P
	H_{i-1}	$\surd$ (Miyaguchi-Preneel)	D	$\surd$	D
	$X_i \oplus H_{i-1}$	B	B	F	F
	V	P	D	F $(*)$	D

Table 5.2: Attacks on the 64 different schemes.

This large number of schemes can be reduced by considering linear transformations of the inputs. A class of schemes that is derived from a single scheme by linear transformation of variables will be called an equivalence class.

- In 7 equivalence classes the function depends on two independent inputs (X_i and H_{i-1}), and 6 transformations are possible, as there are 6 invertible 2×2 matrices over $GF(2)$. It can be shown that in 2 cases the function is secure or is

vulnerable to a fixed point attack, and in 5 cases the function is vulnerable to a direct attack, a permutation attack, or a backward attack.

- In 7 equivalence classes the function depends on a single independent input. Hence one has three possible inputs, namely X_i, H_{i-1}, and $X_i \oplus H_{i-1}$, corresponding to the 3 nonzero vectors of length 2 over $GF(2)$. If the function depends on the sum of the two inputs, it is not trivially weak. However, it is vulnerable to a direct attack (2 cases out of 7) or to a forward attack (5 cases out of 7).
- In 1 equivalence class the function is simply constant.

In table 5.3 an overview of the equivalence classes is given. A further classification is made based on the number of constants in the choices CI. To characterize a class, a relation is given between plaintext P, key K, and feedforward FF.

CI	characterization	class size	–	D	P	B	F	FP	$\surd$
0	$FF = P$, $(P \neq K)$	6						4	2
	$FF = P \oplus K$, $(P \neq K)$	6						4	2
	$FF = K$, $(P \neq K)$	6		2		4			
	$P = K$, $(FF \neq P)$	6		2	2	2			
	$FF = P = K$	3	2				1		
1	$FF = V$, $(P \neq K)$	6		2		4			
	$P = V$, $(FF \neq K)$	6		2	2	2			
	$K = V$, $(FF \neq P)$	6		4	1	1			
	$FF = V$, $(P = K)$	3	2				1		
	$P = V$, $(FF = K)$	3	2				1		
	$K = V$, $(P = FF)$	3	2				1		
2	$FF = P = V$	3	2				1		
	$FF = K = V$	3	2	1					
	$P = K = V$	3	2	1					
3	$FF = P = K = V$	1	1						
Total		64	15	14	5	13	5	8	4

Table 5.3: Overview of the 15 variants, sorted according to the number of constant inputs, denoted with CI.

One can conclude that only 4 schemes of the 64 are secure, and that 8 insecure schemes are only vulnerable to a fixed-point attack. These 12 schemes are listed in table 5.4 and graphically presented in figure 5.2. The roles of X_i and H_{i-1} in the input can be arbitrarily chosen, and the dotted arrow is optional (if it is included, the key is added modulo 2 to the ciphertext). For the dash line, there are three possibilities: it can be omitted or it can point from key to plaintext or from plaintext to key. There are two equivalence classes that are secure, and their simplest representatives are the

scheme by Matyas et al. (no. 1 in the table) and the scheme by Miyaguchi and Preneel et al. (no. 3 in the table). The properties of these 12 schemes are given in table 5.5 together with fixed points and some properties if the underlying block cipher is DES. For each of these schemes it is possible to write a "proof" of their security based on a black box model of the encryption algorithm, as was done for the Davies-Meyer scheme (number 5 in the table) in [328]. The basic idea is that finding a (pseudo)-preimage for a given hash value is at least as hard as solving the equation $H_i = f(X_i, H_{i-1})$ for a given value of H_i. The expected number of evaluations of $f()$ is shown to be 2^{n-1}.

The schemes will now be compared in more detail based on their vulnerability to fixed point attacks, to attacks based on weak keys and on the complementation property, and to differential attacks. The efficiency of the schemes will also be compared.

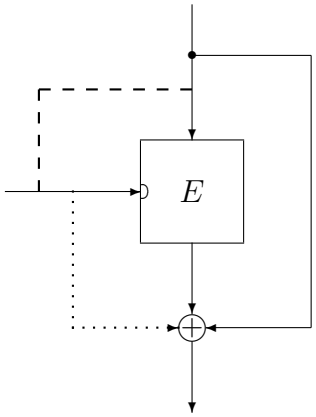

Figure 5.2: Secure configuration for an MDC based on an encryption algorithm and a feedforward (size of hashcode = block length).

Fixed points Fixed points for the function f can only be found easily if X_i is used in the key port. A large set of fixed points can be found in that case, by varying the parameter K. This implies that starting from any initial value, one can easily produce a collision with a birthday attack. The equations are given for scheme 5, as discussed in [227], but can be extended to schemes 5 to 12.

- Generate r values of $D(K, 0)$, by randomly selecting K.
- Generate r values of $E(X_1, IV) \oplus IV$, by randomly selecting X_1.
- The probability of finding a match is about 0.5 if $r \simeq 2^{n/2}$.

This attack provides an efficient way to produce a preimage for the value $D(K, 0)$. If only a pseudo-preimage is required, only constant time is necessary. However, if the attacker gets a **randomly selected** H_t, he will be unable to find the corresponding value of K (this would mean that he can break the cryptosystem), and hence this

no.	function expression
1	$E(H_{i-1}, X_i) \oplus X_i$
2	$E(H_{i-1}, X_i \oplus H_{i-1}) \oplus X_i \oplus H_{i-1}$
3	$E(H_{i-1}, X_i) \oplus X_i \oplus H_{i-1}$
4	$E(H_{i-1}, X_i \oplus H_{i-1}) \oplus X_i$
5	$E(X_i, H_{i-1}) \oplus H_{i-1}$
6	$E(X_i, X_i \oplus H_{i-1}) \oplus X_i \oplus H_{i-1}$
7	$E(X_i, H_{i-1}) \oplus X_i \oplus H_{i-1}$
8	$E(X_i, X_i \oplus H_{i-1}) \oplus H_{i-1}$
9	$E(X_i \oplus H_{i-1}, X_i) \oplus X_i$
10	$E(X_i \oplus H_{i-1}, H_{i-1}) \oplus H_{i-1}$
11	$E(X_i \oplus H_{i-1}, X_i) \oplus H_{i-1}$
12	$E(X_i \oplus H_{i-1}, H_{i-1}) \oplus X_i$

Table 5.4: A list of the 12 secure schemes for a one-way function based on a block cipher.

no.	fixed points		properties if E = DES			differential
	X_i	H_{i-1}	rate	K_w	compl.	attack
1	–	–	1	0	$\surd$	X_i
2	–	–	1	0	$\surd$	X_i
3	–	–	1	K_w	-	X_i
4	–	–	1	K_w	-	X_i
5	K	$D(K,0)$	n/k	0	$\surd$	H_{i-1}
6	K	$D(K,K) \oplus K$	n/k	0	$\surd$	H_{i-1}
7	K	$D(K,K)$	n/k	K_w	-	H_{i-1}
8	K	$D(K,0) \oplus K$	n/k	K_w	-	H_{i-1}
9	$D(K,K)$	$D(K,K) \oplus K$	1	0	$\surd$	X_i, H_{i-1}
10	$D(K,0) \oplus K$	$D(K,0)$	n/k	0	$\surd$	H_{i-1}
11	$D(K,0)$	$D(K,0) \oplus K$	1	K_w	-	X_i, H_{i-1}
12	$D(K,K) \oplus K$	$D(K,K)$	n/k	K_w	-	X_i, H_{i-1}

Table 5.5: Properties of the 12 secure schemes: fixed points, properties if DES is used as the underlying block cipher, and variables to be modified in case of a differential attack.

weakness is of no use to him. In case of collisions, the situation is different. This attack yields in constant time collisions if the initial value can be selected by the attacker. For a given initial value, this attack is not more efficient than a birthday attack on the hashcode. A way to thwart a fixed point attack that was used in the proofs for complexity theoretic constructions is the use of a prefix-free encoding of the message. A more practical approach is adding the length of the message in the padding scheme. The conclusion is that the fixed point attack is in practice not very dangerous.

Weak keys and complementation property A second issue is attacks based on (semi)-weak keys (cf. section 2.5.4.2) and on the complementation property (cf. section 2.5.4.1). They are only relevant if the underlying block cipher is DES or if it has similar properties. An attacker will try to use a weak key and one of the 2^{32} corresponding fixed points. Note that these are fixed points for the block cipher and not for the hash function. This implies that he has to be able to control the key input, which is in general not possible for schemes 1 through 4. Secondly, the plaintext has to be one of the 2^{32} fixed points: this will happen after on average 2^{29} iterations. Indeed, the probability that a random plaintext is a fixed point for a weak key is equal to 2^{-32}, and there are 4 weak keys. In case of a pseudo-preimage attack, one can easily control both key and plaintext, and hence this attack is directly applicable to the 12 schemes. Table 5.5 indicates whether the output will be equal to 0 or to the weak key K_w. For the anti-palindromic keys, a similar attack can be applied using the anti-fixed points. The output will then be 1 or the complement of the anti-palindromic key. This attack is not very dangerous: it only shows that it is relatively easy to find many pseudo-preimages and preimages of these values. It is also possible to extend the pseudo-collision attack in [227] that uses the existence of (semi)-weak key pairs K_1, K_2 such that $E(K_2, E(K_1, P)) = P$. Due to the complementation property one can construct a second input that yields the same output ($\surd$ in table 5.5) or the complement of that output (- in table 5.5). In the case of DES, the problems with (semi)-weak keys and the complementation can easily be solved [216]: fix bits 2 and 3 of the key to "01" or "10". The price paid for this modification is that k will decrease from 56 to 54. This will also limit the security level against random attacks, as will be shown below. The vulnerability to weak keys was remarked by the author (in a 1988 proprietary report) and independently in [216]. In 1990 an extensive study of the use of the combination of (semi)-weak keys and the complementation properties was published [227]. For the Matyas-Meyer-Oseas and the Davies-Meyer scheme they construct collisions for the function f. However, this means no threat to these functions: these constructions only yield a second pseudo-preimage in case of a OWHF or a pseudo-collision for a CRHF.

Differential attacks A third element is the resistance to differential attacks (cf. section 2.5.2.7). The basic idea is to look for differences between the functions that yield the same output with a high probability. The use of differential attacks on this type of hash functions was suggested to the author by I. Damgård [67]. Similar ideas

were developed by E. Biham [24], without looking into a detailed optimization. For a more detailed treatment the reader is referred to appendix C. An attacker will look for a characteristic for the hash function that yields a collision, i.e., an output exor of zero. This will have to be translated into properties for the characteristics of the underlying block cipher. For all 12 schemes that are given in table 5.5, a characteristic is necessary where the exor of the input is equal to the exor of the output. In the last column of the table it is indicated which variables have to be modified. It is clear that it is easier for an attacker to come up with two blocks X_i with a given difference than to find two messages for which the hash value after $i-1$ blocks has a given difference. One can conclude that the first four schemes are more vulnerable to differential attacks.

The following differences with differential cryptanalysis of a block cipher used in ECB mode can be indicated:

- The key is known (in some cases it can be even chosen by the attacker), which can be used in several ways. First, the input data can be selected in such a way that the characteristic will be satisfied trivially in the first 3 rounds (this argument holds only for collisions and not if a second preimage is constructed). A second element is that an early abort strategy will reduce the number of computations: it can be checked after every round whether the characteristic is still satisfied. A third element is that the characteristics with enhanced probability can be used: in the case of DES one can show that if certain relations between key bits are present, the probability of the best known iterative characteristic will increase with a factor 1.6. Finally the attacks can be performed off-line and in parallel, which renders them much more realistic.

- A single right pair is sufficient, i.e., a pair that follows the characteristic. Most differential attacks on block ciphers need a large set of right pairs.

- It is not possible to use a 1R-, 2R- or 3R-attack: in these attacks the characteristic is 1, 2, or 3 rounds shorter than the block cipher. This means that the characteristic has to be 1, 2, or 3 rounds longer and will have a lower probability. Moreover, a characteristic with an even number of rounds (namely 16) is necessary to attack the scheme based on DES, but for the time being no good characteristics with an even number of rounds are known.

The number of operations to construct a collision and a second preimage for one of the 12 hash functions based on DES with t rounds (t odd) is indicated in table 5.6. The second column should be compared to a birthday attack that requires about 2^{33} encryptions, while the third column should be compared to 2^{56}, the expected number of operations to find a second preimage. Note that in case of a birthday attack the number of solutions increases quadratically with the number of trials.

The attacks indicated can further be optimized by a more careful selection of the inputs: if one only tries to look for a second preimage if the input satisfies certain constraints, the number of operations will be comparable to the number of operations to construct a collision. Another possibility is to start the attack in the middle, and to

number of rounds	collision $\log_2$(# encryptions)	second preimage $\log_2$(# encryptions)
7	14.4	21.6
9	19.6	26.8
11	26.6	33.8
13	34.0	41.2
15	41.2	48.4
17	48.3	55.5

Table 5.6: The number of encryptions for a differential attack on one of the 12 DES-based hash functions if DES had an odd number of rounds.

work forwards and backwards: this could reduce the workload to produce a collision with a factor 146. The major problem however is the search for good characteristics with an even number of rounds. It would be interesting to prove a lower bound on the probability of such characteristics, which could show that differential attacks are not feasible.

A second remark is that by symmetry arguments this attack is also applicable to functions where the message block is entered through the key port: keys with a specific difference can be chosen to yield specific outputs to the F-function. However, in the case of DES this will not be very useful, as the same key bit appears at different places in the F-function in every round. Therefore it is expected that only a very small number of rounds can be attacked in this way. A further extension could be an attack where both key and plaintext differences are used.

A third remark is that if FEAL [225] or FEAL-NX [228] are used as the underlying block cipher, schemes 1 to 4 become very weak. In [21] a 4 round iterative characteristic is shown with probability 1/256, for which lower and upper halves of the difference pattern are equal. This means that for FEAL or FEAL-NX with $4r$ rounds, a single encryption will yield a second preimage with a probability of 2^{-8r}. For more details the reader is referred to section 4 of appendix C.

The main conclusion is that if DES is used as the underlying block cipher, for the time being differential attacks pose no serious threat to this type of scheme.

Efficiency A final element in the comparison of the different schemes is the efficiency. If the message is used only in the key port, the rate will be equal to n/k, which is equal to 1.14 in case of DES. If precautions against weak keys and the complementation property have been taken, this increases to 1.19. A second less important issue is the number of additional exors. Scheme 1 and 5 require only a single additional exor, while all other schemes require 2 additional exors (note that in scheme 2 and 6 $X_i \oplus H_{i-1}$ has to be computed only once).

As a conclusion, one can state that the best solution is to avoid the situation where an attacker can control the key input. This requires that the key input be fixed to

H_{i-1}, and leaves the possibilities 1 to 4: the plaintext and the feedforward value can be either X_i or $X_i \oplus H_{i-1}$. The advantage is that it becomes hard to force the key to be a weak key, and block ciphers are designed to be secure for all possible plaintexts (there should be not something like a "weak plaintext"). Hence precautions to avoid weak keys are not very important: it should be difficult for an attacker to find a message for which H_i is a weak or a semi-weak key. Moreover in this case it is not possible to construct fixed points. If $k < n$, like for DES, there is another argument to select as key input H_{i-1}: the rate equals 1 instead of n/k. The main disadvantage of these schemes is that they are more likely to be vulnerable to a differential attack. Finally it should be remarked that scheme 1 is considered for ISO standardization [159], but we feel that the other 3 schemes certainly have the same strength, namely a security level of k bits. In the case of a block cipher that is vulnerable to differential attacks, one of the other schemes should be selected.

If $k \neq n$, the question arises whether it is possible to use variables X_i and H_{i-1} of length $max(n, k)$, in order to maximize the security level. The idea is that bits that are not used in the key or as plaintext might influence the output through some exors. The following proposition shows that this is not possible:

Proposition 5.1 *The security level of the OWHF is determined by the minimum of k and n, with k the size of the key and n the block length.*

Proof: This can be shown by checking systematically all 12 functions.

- For schemes 1 through 4 the situation is simple when $k > n$: as X_i is not added to the key in these schemes, additional bits of X_i do not influence the output. They could be concatenated to the ciphertext, but this yields only a linear dependence of the output on these bits. Therefore n gives an upper bound to the security level. If $k < n$ the situation is more complicated.
  For scheme 1 it is sufficient to find a match for one of intermediate k-bit H_{i-1}'s even if the hashcode has a size of n bits. It is not possible to use a larger chaining variable as H_{i-1} is used only as the key.
  For scheme 2 it is possible to feed the $n - k$ bits of H_{i-1} that are not used as key bits to the input and output of the block cipher. If one of these bits is modified, the plaintext could be kept constant through appropriate changes in X_i. Owing to the feedforward of both H_{i-1} and X_i to the ciphertext, changes will be cancelled out. This yields 2^{n-k} pairs (X_i, H_{i-1}) with the same image!
  For scheme 3 the cancellation argument no longer holds, but the plaintext will remain constant and the output will depend only linearly on the additional bits of H_{i-1}. Therefore this can not be considered to be sufficiently secure.
  For scheme 4 it is possible to keep the plaintext constant through appropriate changes in X_i. This yields again a linear dependence of the outputs on the additional bits of H_{i-1}.

- For schemes 5 through 8 the proofs can be copied by interchanging the role of X_i and H_{i-1} in the previous arguments.

- For scheme 9 the situation is simple if $k > n$. The additional bits of the plaintext X_i could be added to the key and concatenated to the ciphertext. However, finding a preimage is not harder, as these bits can be easily obtained from the output. If $k < n$, the security level can not be increased as the bits of H_{i-1} are used only in the key port. A similar proof holds for scheme 10.
  For scheme 11, if $k > n$, additional bits of X_i are added to the key. However, the key can be kept constant by appropriate changes in H_{i-1}. As H_{i-1} is added to the ciphertext, the output will depend only linearly on the additional bits of X_i. On the other hand if $k < n$, it is clear that the additional bits of H_{i-1} will be added to the ciphertext, yielding again a linear dependence. The arguments for scheme 12 are similar to those for scheme 11.

■

Note that this proof only discusses the number of operations for a second preimage. It is clear that if one does not know a preimage for instance for scheme 1, that determining H_{i-1} and a corresponding X_i will require 2^n operations (one can do no better than guess with a success probability of 2^{-n}).

To conclude this section, it will be discussed what the limitations are of the 12 secure functions. If they are used with a 64-bit block cipher ($n = 64$), it is clear that they can only result in a OWHF. This means that it is always easy to produce for a given IV two inputs yielding the same output. This can be done with a conventional birthday attack or with the more sophisticated collision search attack proposed by J.-J. Quisquater (cf. section 2.5.1.3). In [227] it is shown how the key collisions described in [267, 270] can be used to produce a pseudo-collision for scheme 1. However, this is trivial in the sense that the number of operations for a key collision search attack is comparable to the number of operations for a collision search based on the birthday attack (that requires more storage). Moreover the key collision search algorithm can as well be applied directly to the hash function.

The limitations of a OWHF have been discussed in chapter 2: repeated use of these functions will weaken them. Moreover if they are based on an iterative construction, it is sufficient to find a preimage for one of the chaining variables. Hence if the function has been applied to r t-block messages, the success probability of an attacker will increase with a factor rt. The speed-up with the factor r could be avoided by selecting a different IV for every message, but this requires that the authenticity of the different IV's is protected. As discussed in chapter 2, the speed-up with the factor t could be thwarted by adding the message length in bits at the end of the message. This is clearly more efficient than the proposal in [328] to add a running count to one of the arguments of f that is increased at every iteration [328], and protected together with the hashcode. This running count should not be reset for a new message, which means that in practice it will have to be at least 28 or 32 bits. If DES is used as the underlying block cipher this would not work since the chaining variable would become too small.

The second question that remains to be resolved is whether these schemes yield a CRHF in case $n, k \geq 128$. For all methods that allow a backward attack, it will be trivial to produce pseudo-collisions. Although these collisions do not form a real threat

to the system, it would be preferable to select constructions for which the function f is not easily invertible. Otherwise it is possible with a meet in the middle attack to produce the preimage for a given hashcode in $O(2^{n/2})$ operations, where this is $O(2^n)$ operations in the ideal case. Attacks based on fixed points are not so relevant: for an initial value selected by an attacker they yield a collision in constant time, but for a fixed initial value the construction of a collision is not more efficient than a simple birthday attack. As a conclusion one can state that the best solution is to use one of the schemes that is recommended for a OWHF in case $n = 64$.

5.3.2 Size of hashcode equals twice the block length

This type of functions has been proposed to construct a CRHF based on a block cipher with a block length of 64 bits like DES. Therefore the functions in this section will be intended for CRHF and not as a OWHF: for this class of functions the constructions of the previous section offer a better solution.

5.3.2.1 Iteration of a OWHF

A very weak solution that is discussed in [6] is to keep as the result not only H_t but also H_{t-1}. The scheme can be broken easily by fixing the last block and by subsequently generating collisions for H_{t-1}. A second class of proposals is based on the iteration of a scheme with an n-bit result for two different values of a parameter. All these solutions are straightforward and very appealing, but the vulnerability of the iterated schemes to generalized meet in the middle attacks (cf. section 2.5.2.3) renders them suspected.

- In the first scheme the parameter is the key, as is discussed in [6] for the addition scheme followed by CBC encryption. This scheme will succumb to a generalized meet in the middle attack.
- A second proposal is the initial value, as suggested in [168, 347] for the Matyas-Meyer-Oseas scheme. In this case, one can find a preimage for a given value in only $2 \cdot 2^{64}$ operations, and a second preimage in $2 \cdot 2^{56}$ operations if DES is used as the underlying block cipher. The left and right halves of the hashcode will be denoted with $H1$ and $H2$ respectively, and s is a mapping from the ciphertext space to the key space. The first message block X_1 is obtained from solving the equation $s(E(IV_1, X_1) \oplus X_1) = s(H1)$, which requires 2^{56} encryptions. The next message block X_2 is obtained from the equation $E(s(H1), X_2) \oplus X_2 = H1$; here finding a solution requires about 2^{64} encryptions. All subsequent message blocks will then be chosen equal to X_2, which implies that the left half of the hashcode will always be equal to $H1$. Finding a match for $H2$ requires only 2^{64} encryptions. There is however a twist in this method: the value of $H2_i$ is obtained from an injective mapping, namely $E(s(H2_{i-1}), X_2) \oplus X_2)$. This means that after about 2^{28} operations the values of $H2_i$ will be repeating. If the whole procedure would be restarted, the efficiency of the attack would decrease. If however a second solution is found for X_2, denoted with X_2', a sufficient number of values of $H2_i$

can be obtained from several combinations of X_2 and X_2'. In case of a second preimage, it is sufficient to look for a match of the 112-bit chaining variables, which requires only 2^{56} operations.

If one of the IV's is a weak key, it is known that finding a fixed point X_1 for the block cipher is easy (cf. section 2.5.4). In this case one obtains $H1 = 0$, which is again a weak key in the case of DES. The fact that the value of $H2_i$ will be repeating fast can now be exploited to construct a collision in about 2^{28} operations.

This scheme was replaced by a more secure scheme MDC-2, where an interaction between the chains is introduced. Therefore it will be called MDC-1. In the next section its security will be compared to the security level of MDC-2 and MDC-4.

- A third proposal is BMAC, that was originally described in Internet RFC 1040 [193]. It uses the CBC mode (without the addition), and for the generation of the second part of the hashcode the order of the message blocks is reversed. A constrained meet in the middle attack (cf. section 2.5.2.2) is described in [223, 224]. Let H_t and H_t' be the outcome of the CBC calculations if the message is considered in normal and reverse order respectively. The message is split in two parts: r variations on the first part are constructed such that they yield with initial value IV' a reverse CBC hashcode of H_t', and r variations on the second part are constructed such that they yield with initial value zero a reverse CBC hashcode of IV'. This guarantees that the reverse CBC condition will be fulfilled if a message from the first set is concatenated to one of the second set. Finally an ordinary meet in the middle attack is applied: one looks for matching values of the chaining variable in between the two blocks for the forward CBC calculation. The probability of a match is sufficiently high if $r = 2^{\frac{n}{2}}$.

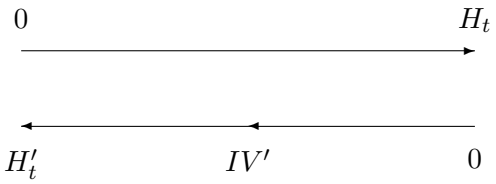

Figure 5.3: Intermediate variables in the CBC computation for the attack on BMAC.

5.3.2.2 Schemes with rate greater than or equal to 2

The following schemes will be reviewed in this section: the proposal by Merkle, MDC-2, MDC-4, the scheme by Zheng, Matsumoto, and Imai, and the Algorithmic Research (AR) hash function.

The proposal by Merkle (Crypto'89) A first proposal [213] has the property that its cryptanalysis requires at least 2^{56} operations (in case of DES). Its construction is based on the 'meta-method' that has been described in section 2.4.3. As a consequence, it can be shown that the scheme is as secure as the underlying block cipher under the assumption that this is a random function (black box model). Of course this assumption does not hold if a specific block cipher is used: several deviations of random behavior have been noted for e.g., the DES. The validity of the proof is limited by the impact of these deviations on the security of the hash function. When the key length equals k bits, the proposed function f reduces the input length of $n+k-1$ bits to $2k$ bits. To simplify notation the expression $E^{\overline{\oplus}}(X)$ will be used, where the first k bits of X will be used for the key and the next n bits for the plaintext. The function chop_r drops the r least significant (or rightmost) bits of its argument. Then the function f is defined as follows:

$$f = \text{chop}_{16}\left[E^{\overline{\oplus}}(0 \parallel X_i) \parallel E^{\overline{\oplus}}(1 \parallel X_i)\right] .$$

The rate of this hash function is very large, namely $2n/(n-k-1)$, which is equal to 18.3 for DES. Another disadvantage in case of DES is the processing of data in blocks of 119 bits.

The performance can be improved by working out the ideas. The security level remains the same (2^k operations), but the security proof becomes much more complicated. A first improvement requires 4 DES applications for hashing $2k+n-4$ bits to $2n$ bits, resulting in a rate of $4n/(2k-n-4)$, which is 5.8 for the DES. X_i is split in 2 parts of length $k+n-2$ and $k-2$, that are denoted with X_i^0 and X_i^1 respectively.

$$f = E^{\overline{\oplus}}(00 \parallel E^{\overline{\oplus}}(10 \parallel X_i^0) \parallel X_i^1) \quad \parallel \quad E^{\overline{\oplus}}(01 \parallel E^{\overline{\oplus}}(11 \parallel X_i^0) \parallel X_i^1)$$

A second variation reduces a $2(n+k-3)$-bit input (234 for DES) to a $2n$-bit output with 6 applications of DES. It can be described as follows: split X_i in 2 parts of length $n+k-3$, denoted with X_i^0 and X_i^1 respectively.

$$\begin{aligned} f = a : E^{\overline{\oplus}}\left(00 \parallel c : \text{chop}_5\left[E^{\overline{\oplus}}\left(100 \parallel X_i^0\right)\right] \parallel e : \text{chop}_5\left[E^{\overline{\oplus}}\left(101 \parallel X_i^1\right)\right]\right) \parallel \\ b : E^{\overline{\oplus}}\left(01 \parallel d : \text{chop}_5\left[E^{\overline{\oplus}}\left(110 \parallel X_i^0\right)\right] \parallel f : \text{chop}_5\left[E^{\overline{\oplus}}\left(111 \parallel X_i^1\right)\right]\right) \end{aligned}$$

The rate equals $3n/(k-3)$, in case of DES about 3.62. The letters a, b, c, d, and f denote intermediate values that will be used in the security proof.

The analysis of this scheme is rather involved. The security proof of the scheme will be sketched, and it will be shown how the results described in appendix B on multiple collisions allow us to tighten Merkle's original bound for the security level from 52.5 to 56 bits. The proof is based on two definitions:

Random linkage map (rlm): two sets of 2^{56} tuples, where every tuple has 2 59-bit elements. The tuple from the first sets are values $\{c, d\}$ generated from a random X^0, and the tuple $\{e, f\}$ are values generated from a random X^1.

Doubly linked quadruple with respect to an rlm is a tuple $\{c, e, d, f\}$, where $\{c, d\}$ appears in the first set of the rlm, and $\{e, f\}$ appears in the second set.

The philosophy behind the proof is that the collision finder algorithm gets an rlm for free. This consists intuitively of all the useful information that any algorithm can ever hope to obtain about the 4 intermediate values. The generation of an rlm requires 2^{58} computations. The algorithm is subsequently only allowed to use function calls to produce values for either a or b. Under these conditions it is shown that the expected running time for an optimal algorithm F to produce a collision for f is as least as long as the expected running time for an algorithm F' to find a pair of doubly linked quadruples such that both quadruples generate the same output.

The next step is to estimate the success probability of an algorithm F'. The idea is first to produce a k-fold collision for the value of b, i.e., k inputs $\{d, f\}$ that yield the same value of b. If this can be done, the success probability of a collision for a is equal to $k/2^{64}$. The security level of 2^{56} is achieved if $k < 256$. A k-fold collision for b can be produced in two ways: a value of d or f might occur more than once in the same set and in the same position of an rlm, and identical inputs yield of course the same value for b; a second possibility is that the same value for b is obtained with either $d \neq d'$ or $f \neq f'$.

R. Merkle uses in his proof the following crude upper bound for the number of k-fold collisions: if r balls are drawn with replacements from a set of n balls ($n > r$), the number of balls that occurs k times is upper bounded by $2^{n-k(n-r)}$. For collisions of d and f ($r = 2^{56}$, $n = 2^{59}$), he obtains $k \leq 20$ and for collisions for b ($r = 2^{56}$, $n = 2^{64}$), this gives $k \leq 7$. The number of inputs that yield this value of b is then given by $20^2 + 7 \cdot 20^2 = 2800$. The corresponding security level is only 52.5 bits.

A better approximation of the number of k-fold collisions has been derived in appendix B. The expected number of collisions is given by equation (B.17):

$$\lambda_k = n \frac{\exp\left(-\frac{r}{n}\right)}{k!} \left(\frac{r}{n}\right)^k .$$

The numerical results for the parameters of the Merkle scheme can be found in table 5.7.

k		2	3	4	5	6	7	8	9	10	11
n	r										
2^{59}	2^{56}	51.82	47.23	42.23	36.91	31.33	25.52	19.52	13.35	7.03	0.57
2^{64}	2^{56}	46.99	37.41	27.41	17.09	6.50	−4.30				
2^{64}	2^{55}	45.00	34.41	23.41	12.09	0.51	−11.30				

Table 5.7: Binary logarithm of the expected number of k-fold collisions for drawings of r out of n with replacements.

If one wants to prove that the security level is at least 56 bits, any algorithm is limited to 2^{56} computations of a or b. The following strategy is now suggested:

select $2^{27.5}$ values of d and f for which the largest number of collisions can be found in the rlm. From table 5.7 it can be seen that $2^{0.57}$ 11-fold collisions are expected, $2^{7.03}$ 10-fold collisions, etc. This continues, until one completes the set with $2^{27.07}$ 6-fold collisions. The average number of collisions in this set is equal to 6.26. Now all values of d and f are combined to produce 2^{55} b-values, and with each value for b will correspond $6.26^2 = 39.2$ trials for a[1]. This operation is repeated with the roles of a and b interchanged such that the total number of function evaluations is equal to 2^{56}. The next step is to look for collisions among the b values. If for a certain b-value a k-fold collision is obtained, the corresponding number of trials for a is equal to $k \cdot (6.26)^2$. Indeed, one can write for this value of b

$$b(d_1, f_1) = b(d_2, f_2) = \ldots = b(d_k, f_k)\,.$$

With every value of d_i and f_i correspond on average 6.26 other values in the rlm. The *maximal* number of trials for a will be obtained if $k = 6$, and this number is equal to 235; this will occur only once. The expected number of collisions for b is approximately equal to

$$2^{-55} \sum_{k=1}^{\infty} k^2 (6.26)^2 \lambda_k\,.$$

This is equal to $1.00195 \cdot 6.26^2 = 39.26$. If the symmetry between a and b is not used (which means that 2^{56} computations for b are made and none for a), the average number of collisions for d and f is equal to 6.18, and the average number of collisions for b is equal to $1.00391 \cdot 6.18^2 = 38.41$. Note that the maximal number of collisions here is 230, and this occurs for 91 values of b. The conclusion is that this attack corresponds to a security level of $64 - \log_2(39.26) = 64 - 5.3 = 59.7$ bits. Hence the security level of the scheme is lower bounded by 56 bits. Other suggestions by R. Merkle to tighten the proof are to distribute the 2^{56} operations over all computations, and to limit the information that is given for free. It is important to remark that this scheme should be modified to take into account attacks based on weak keys and on the complementation property.

MDC-2 and MDC-4 The authors of the first papers that referred to these hash functions are C. Meyer and M. Schilling, C. Meyer, and S. Matyas [216, 217, 204], while the patent for MDC-2 and MDC-4 is on the names of B. Brachtl, D. Coppersmith, M. Hyden, S. Matyas, C. Meyer, J. Oseas, S. Pilpel, and M. Schilling [28]. No security proof has been given for either scheme, but on the other hand no attacks have been demonstrated. The basic philosophy is to extend the Matyas-Meyer-Oseas scheme (cf. section 5.3.1.3). MDC-2 is currently under consideration for standardization as a collision resistant hash function based on a block cipher in ISO-IEC/JTC1/SC27 [159]. Note that a variant on MDC-2 was proposed in [161]. It will be discussed in appendix C.

[1] This is in fact an approximation, as the average of the squares is not equal to the square of the average.

MDC-2 can be described as follows:

$$
\begin{aligned}
T1_i &= E^{\oplus}(H1_{i-1}, X_i) = LT1_i \parallel RT1_i \\
T2_i &= E^{\oplus}(H2_{i-1}, X_i) = LT2_i \parallel RT2_i \\
H1_i &= LT1_i \parallel RT2_i \\
H2_i &= LT2_i \parallel RT1_i \,.
\end{aligned}
$$

The chaining variable $H1_0$ and $H2_0$ are initialized with the values IV_1 and IV_2 respectively, and the MDC is equal to the concatenation of $H1_t$ and $H2_t$. A similar notation will be used for the next schemes. A graphical representation is given in figure 5.4. The rate of this scheme is equal to 2. In order to protect this scheme against attacks based on semi-(weak) keys (cf. section 2.5.4.2), the second and third key bits are fixed to 10 and 01 for the first and second encryption.

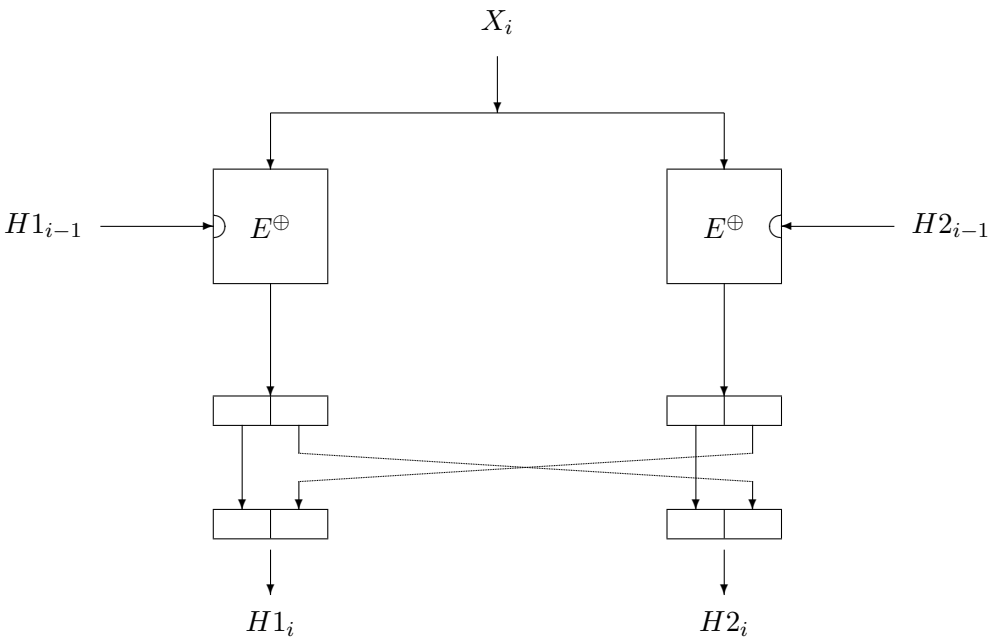

Figure 5.4: One iteration of the MDC-2 hash function.

Part of the following evaluation of MDC-2 and MDC-4 is based on the work by I. Damgård [67] on this scheme. The basic step of MDC-2 is certainly not collision resistant: for any fixed choice of X_i, one can find collisions for both $H1_i$ and $H2_i$ independently with a simple birthday attack. As this scheme is based on the Matyas-Meyer-Oseas scheme, all remarks on this scheme concerning efficiency, weak keys, and complementation property can be transferred to this scheme. If the feedforward would be omitted (the basic scheme would then be the dual of the Rabin scheme [274]), the scheme would become invertible: producing a pseudo-preimage requires constant time, hence a (second) preimage requires only about 2^{54} operations (this follows from

proposition 2.2). If the switch of the left and right part would be omitted, one would have two independent chains and the following attack would apply: compute a large number of inverses for both chains (which requires at least 2 message blocks)—for the time being no feasible algorithm is known to do this—and look for a match between the preimages of the two chains. Switching left and right halves thwarts this attack, as both chains get mixed. Finally one can remark that fixing the key bits does not only avoid weak keys, but this also guarantees that the two encryption keys are different. This would be a serious problem if $IV_1 = IV_2$.

The number of operations (without taking into account constant factors) for several attacks on MDC-1, MDC-2, and MDC-4 (cf. infra) have been indicated in table 5.8. The factors 2 and 4 indicate how many encryptions are required for every step, or how many separate steps have to be performed. If one is looking for a preimage, one has a trade-off between storage and computations: this is the case if a large number of values can be attacked at the same time, or if some precomputations can be done. The number of encryption operations is then the original number divided by the number of stored variables (at most 2^{55}). Note that in case of MDC-2 the size of these variables is 54 bits and not 64 bits as claimed in [216]. The number of operations for a (second) preimage for MDC-2 and MDC-4 is infeasible, but significantly lower than was estimated by the designers.

A pseudo-collision can be found for MDC-1 and MDC-2 by choosing X_1 randomly, taking IV_2 equal to the specified value and independently trying values for IV_1. For a pseudo-preimage one can choose a random X_1 and perform independent trials for IV_1 and IV_2. A pseudo-preimage can be used to compute a preimage faster than exhaustive search based on proposition 2.2. A different way to construct pseudo-collisions for MDC-2 was reported in [227]. First an attacker constructs key collisions (cf. section 2.5.2.6) for a random plaintext P, that satisfy at the same time the constraints on the second and third bit. The one-block message P with initial value K_1, K_2 will yield the same hashcode as the message $\overline{P}$ with initial value $\overline{K_1}, \overline{K_2}$. B. den Boer remarked that if $IV_2 = \overline{IV_1}$, collisions can be produced in the following way: let P be a plaintext such that $E(IV_1, P) = \overline{E(IV_2, P)}$ on the 54 positions that are selected from the output (finding such a P requires about 2^{55} encryptions). In that case P and $\overline{P}$ hash to the same value. Collisions with random IV for MDC-1 can be found with less effort because it is not guaranteed in this scheme that the two chains have different keys. If IV_1 is chosen equal to IV_2, both chains are identical.

The differential attacks against the Matyas-Meyer-Oseas scheme have been extended to MDC-2. In this case the input pair has to be a good pair for both characteristics. Essentially all properties can be transferred, but the overall probability of success will be the product of the probabilities, under the reasonable assumption that both key values are independent. The only step where the approach is more complicated is to satisfy the characteristic automatically in the second round of both encryption instances. Here a careful selection of the inputs is necessary to produce an input that is a right pair for both key values with probability 1. For more details the reader is referred to appendix C. The final results are indicated in table 5.9.

	MDC-1	MDC-2	MDC-4
preimage	$2 \cdot 2^{64}$	$2 \cdot 2^{82}$	$1 \cdot 2^{109}$
2nd preimage	$2 \cdot 2^{56}$	$2 \cdot 2^{82}$	$1 \cdot 2^{109}$
pseudo-preimage	$2 \cdot 2^{56}$	$2 \cdot 2^{54}$	$1 \cdot 2^{90}$
collision (fixed IV)	$2 \cdot 2^{56}$	$2 \cdot 2^{54}$	$4 \cdot 2^{54}$
collision (random IV)	$2 \cdot 2^{28}$	$2 \cdot 2^{54}$	$2 \cdot 2^{54}$
pseudo-collision	$2 \cdot 2^{28}$	$2 \cdot 2^{27}$	$1 \cdot 2^{41}$

Table 5.8: Number of operations for preimage and collision attacks on MDC-1, MDC-2, and MDC-4.

number of rounds	collision $\log_2$(# encryptions)	second preimage $\log_2$(# encryptions)
7	29.8	44.2
9	42.2	56.6
11	55.4	69.8
13	71.0	85.4
15	87.4	101.8
17	99.6	114.0

Table 5.9: The number of encryptions for a differential attack on MDC-2 for an odd number of rounds.

The conclusion of appendix C is that only for 11 rounds or less the collision search is faster than a birthday attack. A second preimage can be found faster than exhaustive search if the number of rounds is less than 17. In order to be of the same order of magnitude as a birthday attack for 15 rounds, a characteristic with probability of 2^{-32} or better would be required (this would be 1/40 per two rounds for an iterative characteristic). Again it should be remarked that no good characteristics are known for an even number of rounds.

The following observation was made on the initial values that are proposed by the designers (the hexadecimal values $\texttt{2525252525252525}_\texttt{x}$ and $\texttt{5252525252525252}_\texttt{x}$): they are chosen in such a way that differential attacks on the first iteration of MDC-2 require 128 more operations than the optimal situation (for an attacker) that is exploited in our attack.

One iteration of MDC-4 is defined as a concatenation of two MDC-2 steps, where the plaintexts in the second step are equal to $H2_{i-1}$ and $H1_{i-1}$:

$$T1_i = E^{\oplus}(H1_{i-1}, X_i) = LT1_i \parallel RT1_i$$
$$T2_i = E^{\oplus}(H2_{i-1}, X_i) = LT2_i \parallel RT2_i$$

$$
\begin{aligned}
U1_i &= LT1_i \parallel RT2_i \\
U2_i &= LT2_i \parallel RT1_i \\
V1_i &= E^{\oplus}(U1_i, H2_{i-1}) = LV1_i \parallel RV1_i \\
V2_i &= E^{\oplus}(U2_i, H1_{i-1}) = LV2_i \parallel RV2_i \\
H1_i &= LV1_i \parallel RV2_i \\
H2_i &= LV2_i \parallel RV1_i\,.
\end{aligned}
$$

The rate of MDC-4 is equal to 4. It is clear that the "exchange" of $H1_{i-1}$ and $H2_{i-1}$ in the second step does not improve the algorithm: after the exchange of right halves, $U1_i$ and $U2_i$ are symmetric with respect to $H1_{i-1}$ and $H2_{i-1}$.

Both finding a preimage and finding a pseudo-collision is harder than in the case of MDC-2, as indicated in table 5.8. On the other hand, collisions for the basic compression function of MDC-2 with the same value of $(H1_{i-1}, H2_{i-1})$ will also yield collisions for MDC-4, but generally this property does not hold for other collisions for the basic function like pseudo-collisions. Hence differential attacks that produce a collision for MDC-2 will also produce a collision for MDC-4, but finding a second preimage with a differential attack will be much harder: the probability that a single message block is sufficient to find a second preimage is very small.

The improved pseudo-preimage attack is based on the fact that $V1_i$ depends only on $H1_{i-1}$ through 26 bits of $LT1_i$, and that 10 bits of $H1_{i-1}$ (namely the 8 parity bits and the 2 bits that are fixed in the key port) only influence the output in the second half of the algorithm (this observation was suggested to the author by B. den Boer).

1. Choose a random value of X_i and a 26-bit constant S.
2. Calculate $T1_i$ for all 2^{54} relevant values of $H1_{i-1}$ (only the bits that are used in the first half). It is expected that in 2^{28} cases the 26 relevant bits of $LT1_i$ will be equal to S (indeed, 4 parity bits in the first half are ignored and 2 more bits are fixed).
3. Calculate $T2_i$ for all 2^{54} relevant values of $H2_{i-1}$. Next, extend $H2_{i-1}$ with all possible values for the 10 unused bits, and compute $V1_i$, under the assumption that $LT1_i = S$ (this blocks the influence of $H1_{i-1}$ on this calculation). This requires in total $2^{64} + 2^{54}$ DES encryptions, hence one expects to find the correct value of $V1_i$. In this way $H2_{i-1}$ is determined.
4. For the 2^{28} cases with $LT1_i = S$, extend $H1_{i-1}$ with all possible values for the 10 unused bits, and compute $V2_i$. This requires in total 2^{38} DES encryptions, and the correct value is obtained with probability 2^{-26}.
5. If no match is obtained, one chooses a new value of X or for S; in the latter case, one can avoid recomputing $T1_i$ and $T2_i$ at the cost of more storage.

One can conclude that finding a preimage requires $2^{64} \cdot 2^{26} = 2^{90}$ DES encryptions and a storage of 2^{28} 54-bit quantities (if for every trial a new value for X is chosen). With proposition 2.2, one finds that a (second) preimage requires about 2^{109} DES encryptions.

The improved pseudo-collision attack is based on the same principles:

1. Choose a random value of X_i and of $H2_{i-1}$ (this can be the specified value).
2. Calculate $T1_i$ for $2^{40.5}$ values of $H1_{i-1}$. It is expected that there will be a 26-bit integer S such that in $2^{14.5}$ cases the 26 relevant bits of LT_i will be equal to S (in fact for 50% of the integers S there will be $2^{14.5}$ cases or more).
3. For these $2^{14.5}$ cases with $LT1_i = S$, extend $H1_{i-1}$ with all possible values for the 10 unused bits, and compute $V2_i$ for the $2^{14.5+10} = 2^{24.5}$ different inputs. The probability to find a collision for $V2_i$ is equal to $2^{2 \cdot 24.5}/2^{65} = 2^{-16}$ (cf. section 2.5.1.3).
4. If no match is obtained, one chooses a new value for S; one can avoid recomputing $T1_i$ if one stores $2^{16} \cdot 2^{14.5} = 2^{30.5}$ 54-bit quantities (about 10 Gigabytes).

One can conclude that finding a pseudo-collision requires $2^{41.5}$ DES encryptions and a storage of $2^{30.5}$ 54-bit quantities.

The scheme by Zheng, Matsumoto, and Imai Another scheme based on a collision resistant function was suggested with some theoretical arguments in [339, 343] (cf. section 4.3.2.7). The weaknesses that have been identified in this scheme have been presented at Auscrypt'92 [263]. The primitive function compresses a 224-bit input to a 128-bit output and is based on xDES^1. This block cipher is one of the extensions of DES that has been proposed in [340]. xDES^1 is a three round Feistel cipher [104, 105] with block length 128 bits, key size 168 bits, and with the F function equal to DES. One round is defined as follows:

$$\begin{aligned} C1_{i+1} &= C2_i \\ C2_{i+1} &= C1_i \oplus DES(K_i, C2_i)\,, \qquad i = 0, 1, 2\,. \end{aligned}$$

Here the variables $C1_i$ and $C2_i$ are 64-bit blocks, and the variables K_i are 56-bit keys. The block cipher is then written as

$$C2_3 \parallel C1_3 = \text{xDES}^1(K_1 \| K_2 \| K_3, C1_0 \| C2_0)\,.$$

Here $C1_0$ and $C2_0$ are the first and second part of the plaintext, and $C2_3$ and $C1_3$ are the first and second part of the ciphertext. The round function that is claimed to be collision resistant consists of 2 xDES^1 operations:

$$f(Y1 \| Y2) = \text{xDES}^1\left(\text{chop}_{72}\left(\text{xDES}^1(IV \| Y1, \alpha)\right) \| Y2, \alpha\right)\,.$$

Here $Y1$ and $Y2$ are 112-bit blocks, α is a 128-bit constant, and IV is a 56-bit initialization variable. This variable has been added to clarify the description of the scheme. The complete hash function has the following form:

$$H_i = f(H_{i-1} \| X_i)\,,$$

where H_{i-1} is a 128-bit block, and X_i is a 96-bit block. The rate of this scheme is equal to 4. Note that one could also start from xDES[1] and classify this as a single length hash function.

If $IV = K_1$ and $Y_1 = K_2 \| K_3$, it is very easy to produce collisions for the atomic operation: every data bit is used only once in the key input of DES, which means that it can be simply replaced by a key collision for the DES plaintext equal to the second part of α with the algorithm described in [267].

A stronger scheme is obtained if IV is equally distributed over K_1, K_2, and K_3, and if the size of IV is increased [345]. However, it will be shown that independently of the size of IV, the security level can not be larger than 44 bits. If the size of IV is equal to v bits (in the original proposal $v = 56$), the number of fixed bits of IV that enter the key port of a single DES block is equal to $v/3$ (it will be assumed that v is divisible by 3). It can be shown that the rate of this scheme is then equal to

$$R = \frac{6 \cdot 64}{208 - 2v} .$$

The number of bits of Y_1 that enter the key port will be denoted with y, hence $y + v/3 = 56$. The following attacks are now considered:

- For the fixed value of the right part of α and of the first $v/3$ bits of IV, one can calculate and store a set of 2^z different ciphertexts. The expected number of collisions in this set is approximately equal to 2^{2z-65} (cf. appendix B). If $y > 32$, implying $v < 72$, a value of $z = 33$ is clearly sufficient to obtain a collision. If on the other hand $y \leq 32$, one will take $z = y$, and the probability of success is smaller than one. One can however repeat this procedure, (e.g., if one attacks a DES block different from the first one, a different value can be chosen for the value of the bits of Y_1 that enter the first DES), and the expected number of operations for a single collision is equal to 2^{65-y}, while the required storage is equal to 2^y. An extension of the Quisquater algorithm could be used to eliminate the storage. If the security level S is expressed in bits, it follows that $S = \max\{65 - y, 33\}$. With the relation between y and v, one obtains $S = \max\{9 + v/3, 33\}$.

- A second attack follows from the observation that only v bits are kept from the output of the first xDES[1] operation (hence the chop operation is chopping $128 - v$ bits). It is clear that finding a collision for the remaining v bits requires only $2^{v/2+1}$ operations, or $S \leq v/2 + 1$ bits. This attack is more efficient than the first attack if $v < 64$ bits.

The relation between S and v can be summarized as follows:

$$\begin{aligned} v < 64 &: S = v/2 + 1 \\ 64 \leq v < 72 &: S = 33 \\ 72 \leq v < 104 &: S = v/3 + 9 . \end{aligned}$$

The following table indicates the relation between S and the rate R. One can conclude

v	56	64	72	84	96	102
S	29	33	33	37	41	43
R	4	4.8	6	9.6	24	96

Table 5.10: Relation between v (the number of bits of IV), the security level S and the rate R.

that producing a collision for the proposed compression function (with fixed size input) requires significantly less than 2^{64} operations. Depending on the allocation of the bits of X_i and H_{i-1} to Y_1 and Y_2, it might also be feasible to produce a collision for the hash function with a fixed initial value: it is certainly possible to produce a collision for the hash function if there is a single DES block where all key bits are selected from X_i.

Note that the scheme proposed by R. Merkle based on a large block cipher (cf. section 5.3.1.3) will succumb to a similar attack if xDES[1] is used as the underlying block cipher.

The Algorithmic Research (AR) hash function The AR hash function was proposed by Algorithmic Research Ltd. and is currently used in the German banking community. It has been circulated as a contribution within ISO [162]. The basic mode of operation is an extension of the CBC mode: the last two ciphertext blocks are added modulo 2 to the message block. This chaining is not secure, but two independent iterations with two different keys are computed in parallel. The hashcode is computed from the outcome of both iterations. The operation of a single chain can be described as follows:

$$f = E(K, X_i \oplus H_{i-2} \oplus H_{i-1} \oplus \eta) \oplus X_i \,.$$

Here $\eta = \texttt{0123456789ABCDEF}_\texttt{x}$ (hexadecimal), and H_{-1} and H_0 are equal to the all zero string. The keys for both iterations are equal to $K = \texttt{0000000000000000}_\texttt{x}$ and $K' = \texttt{2A41522F4446502A}_\texttt{x}$ respectively. The chaining variables of the second iteration will be denoted with H'_i. Two mappings are then defined to combine the output of both iterations:

$$g(K, H1, H2) = E(K, H1 \oplus H2) \oplus E(K, H1) \oplus E(K, H2) \oplus H2$$

and

$$v(K, H1, H2, H3, H4) = g(g(K, H1, H2), g(K, H3, H4)) \,.$$

Then the hashcode is computed as follows:

$$v(K, H_{t-1}, H_t, H'_{t-1}, H'_t) \parallel v(K', H_{t-1}, H_t, H'_{t-1}, H'_t) \,.$$

Although the rate is equal to 2, this scheme will in practice be faster than other schemes with the same rate, as the key remains fixed. In the case of DES, which has

been proposed as the underlying block cipher, this scheme will be 2 to 4.5 times faster in software than MDC-2 (see also section 5.3.4.1).

Several weaknesses have been identified in this scheme. The choice to initialize the 4 chaining variables with 0 will facilitate certain attacks. Another problem is the choice of the all zero key for K, which is a weak key for DES. For a weak key 2^{32} fixed points can be easily found (cf. section 2.5.4.3), and these can be used in the following attacks:

Collision attack: if X_i is chosen such that the input to DES is one of those fixed points, the new value of the chaining variable will be equal to $H_{i-2} \oplus H_{i-1} \oplus \eta$, and thus independent of the fixed point. It is now very easy to use the 2^{32} fixed points to produce a collision for the round function (and hence for the hash function) in the other iteration. The success probability of this attack is 0.39 (cf. section 2.5.1.3); if the attack fails, it can be repeated at a different position in the message. Observe that this attack is independent of the initial values.

Fixed point attack: the fixed points of DES can be used to construct fixed points for the round function after 3 iterations. The chaining variables for the iteration with the weak key will be as follows: H_{-1}, H_0, $H_{-1} \oplus H_0 \oplus \eta$, H_{-1}, H_0. In order to obtain a fixed point for the second iteration (or $H'_2 = H'_{-1}$ and $H'_3 = H'_0$), the following equations have to be solved:

$$\begin{aligned} H'_1 &= X_1 \oplus E(K', X_1 \oplus H'_{-1} \oplus H'_0 \oplus \eta) \\ H'_{-1} &= X_2 \oplus E(K', X_2 \oplus H'_0 \oplus H'_1 \oplus \eta) \\ H'_0 &= X_3 \oplus E(K', X_3 \oplus H'_{-1} \oplus H'_1 \oplus \eta)\,. \end{aligned}$$

Here X_1, X_2, and X_3 are selected such that the plaintexts in the first iteration are the 2^{32} fixed points corresponding to the weak key K. The unknown H'_1 can be solved from the three equations, and every equation will yield 2^{32} values. Under the assumption that DES is a random mapping, a common solution will exist with a very small probability (approximately $1/2^{32}$). If however $H_{-1} = H_0$ and $H'_{-1} = H'_0$ (as specified), the second and third equation are equivalent, and the probability for a common solution is about $1 - e^{-1}$. The total number of operations for this attack equals 2^{33} encryptions. The solution can be used to construct collisions, but also to construct a second preimage for any message (in fact an infinite number of preimages).

One could try to thwart this attack by specifying different initial values; however, finding message blocks such that the chaining variables become equal requires only 2^{34} encryptions. The fixed points can then only be used to produce collisions. Assume that H'_{-1} and H'_0 are different. Then one can choose X_1, X_2, and X_3 such that $H'_3 = H'_2$. The equations to be solved are the following:

$$H'_1 = X_1 \oplus E(K', X_1 \oplus H'_{-1} \oplus H'_0 \oplus \eta) \tag{5.1a}$$

$$H'_2 = X_2 \oplus E(K', X_2 \oplus H'_0 \oplus H'_1 \oplus \eta) \tag{5.1b}$$

$$H'_2 = X_3 \oplus E(K', X_3 \oplus H'_1 \oplus H'_2 \oplus \eta)\,. \tag{5.1c}$$

Choose for X_1 the 2^{32} values that are allowed, and compute from (5.1a) the corresponding value of H'_1. Equation (5.1c) can be rewritten as follows:

$$D(K', X_3 \oplus H'_2) \oplus X_3 \oplus H'_2 = H'_1 \oplus \eta\,.$$

Choose 2^{32} values of $X_3 \oplus H'_2$ and use this equation to obtain a second set of values of H'_1. A matching value for H'_1 will be obtained with probability $1 - e^{-1}$; this also fixes X_1 and $X_3 \oplus H'_2$. Then one chooses for X_2 the 2^{32} possible values, and one computes with (5.1b) the 2^{32} corresponding values of H'_2. This also yields 2^{32} values of X_3, and with high probability one of these values will be a value that is allowed. If this is not the case, one has to choose more values of $X_3 \oplus H'_2$ in the second stage, which will yield more solutions.

In this way one can increase the success probability of the fixed point attack at the cost of increasing the number of operations to $2^{34} \ldots 2^{36}$: if the first attack fails, one looks for identical values of the chaining variables, and one repeats the attack. The solution can now only be used to produce collisions.

I. Damgård and L. Knudsen [68] have identified weaknesses in the g and v function: they become very weak if certain arguments are equal or are equal to 0. They can exploit this weakness together with the fixed points of DES to find 2^{32} messages hashing to the same value (finding a single collision requires only one DES encryption, i.e., the time to find two fixed points of DES). They also have presented different preimage attacks; in order to obtain a success probability that is very close to 1, the attack requires about 2^{66} encryptions.

The AR hash function can be strengthened in several ways: fixed points attacks can be blocked by including the message length in the padding scheme; the key K should be replaced by a different key, that is not weak or semi-weak; four different initial values have to be specified, and g and v should be replaced by stronger mappings. Even with all these modifications the security of the scheme is questionable: the basic chaining mode is very weak, and it is not clear that having two parallel versions of an insecure scheme will yield a secure scheme.

5.3.2.3 Schemes with rate equal to 1

The following schemes will be reviewed in this section: the proposals by Quisquater-Girault, Brown-Seberry-Pieprzyk, and Preneel-Govaerts-Vandewalle.

The scheme by Quisquater and Girault A scheme with rate 1 was proposed in the abstracts of Eurocrypt'89 by J.-J. Quisquater and M. Girault [268]. It was also submitted to ISO as a contribution to DP 10118. It allows no parallel evaluation of the two encryptions:

$$T1_i = E(X1_i, H1_{i-1} \oplus X2_i) \oplus X2_i$$
$$T2_i = E(X2_i, T1_i \oplus H2_{i-1} \oplus X1_i) \oplus X1_i$$

$$H1_i = H1_{i-1} \oplus H2_{i-1} \oplus T2_i$$
$$H2_i = H1_{i-1} \oplus H2_{i-1} \oplus T1_i\,.$$

If DES is used as the underlying block cipher, one still has to specify a mapping from 56 to 64 bits. A weakness of this scheme that was pointed out by the author is that if the scheme is used with DES, complementing X has no effect on the hashcode (due to the complementation property). Independently, the same weakness was reported in [227], together with a number of pseudo-collisions. A more serious problem was an attack by D. Coppersmith [271], that was only published recently in [59]. It resulted in a new version in the proceedings of Eurocrypt'89 [269]. A different attack on the first version by X. Lai and J. Massey [183] shows that finding a preimage with random IV requires only 2^{33} encryptions.

The attack by D. Coppersmith is limited to the case where DES is used as the underlying block cipher. It exploits the existence of 2^{32} fixed points for a weak key (cf. section 2.5.4.3) to produce a collision in about 2^{33} encryptions. The basic observation behind the attack is the following: if $X1_i$ is chosen to be a weak key (e.g., $X1_i = 0$), and $X2_i$ is chosen such that $H1_{i-1} \oplus X2_i$ is a fixed point corresponding to that weak key, $T1_i$ will be equal to $H1_{i-1}$ and hence $H2_i = H2_{i-1}$. The attack can be described as follows. Choose $X1_1 = X1_2 = 0$ (or any other weak key) and choose 2^9 values of $X2_1$ such that $H1_0 \oplus X2_1$ is a fixed point. For each of the resulting values of $H1_1$, on takes 2^{24} values of $X2_2$ such that $H1_1 \oplus X2_2$ is a fixed point. It follows that $H2_2 = H2_1 = H2_0$, and one expects to find two identical values among the 2^{33} values for $H1_2$ (cf. section 2.5.1.3). One still has to prove that one can obtain a sufficient number of values of $X2_2$ and $X2_3$ in the second step. There are 2^{32} fixed points, but because of the mapping from the key space to the plaintext space, on average only one in 2^8 can be obtained. This means that one expects to find 2^{24} fixed points.

D. Coppersmith [59] presents the attack as a correcting block attack with 6 correcting blocks (cf. section 2.5.2.4) that requires 2^{35} encryptions. Assume one starts with 2 t-block messages X and X'. The first step is now to choose 2^{32} plaintext block pairs $(X1_{t+1}, X2_{t+1})$ and $(X1'_{t+1}, X2'_{t+1})$ such that $H2_{t+1} = H2'_{t+1}$. The rest of the attack is similar; the main difference is that here one looks for matches between two sets.

The pseudo-preimage attack by X. Lai and J. Massey will now be described. The basic equations for the scheme are the following (the index i will be omitted in $X1_i$ and $X2_i$):

$$H1_i = H1_{i-1} \oplus H2_{i-1} \oplus X1 \oplus E(X2, X1 \oplus X2 \oplus H2_{i-1} \oplus E(X1, X2 \oplus H1_{i-1}))$$
$$H2_i = H1_{i-1} \oplus H2_{i-1} \oplus X2 \oplus E(X1, X2 \oplus H1_{i-1})\,.$$

First the equations are simplified with the substitution

$$S = X1 \oplus E(X1, H1_{i-1} \oplus X2)\,, \tag{5.2}$$

which leads to:

$$H1_i = H1_{i-1} \oplus H2_{i-1} \oplus X1 \oplus E(X2, X2 \oplus H2_{i-1} \oplus S)$$
$$H2_i = H1_{i-1} \oplus H2_{i-1} \oplus X1 \oplus X2 \oplus S\,.$$

Subsequently the second equation is added to the first, and $X2 \oplus H1_{i-1}$ is eliminated from the second equation using (5.2):

$$\begin{aligned} H1_i \oplus H2_i &= S \oplus X2 \oplus E(X2, S \oplus X2 \oplus H2_{i-1}) \\ H2_i &= S \oplus X1 \oplus D(X1, S \oplus X1) \oplus H2_{i-1}\,. \end{aligned}$$

Both equations can now easily be solved for $H2_{i-1}$:

$$\begin{aligned} H2_{i-1} &= S \oplus X2 \oplus D(X2, S \oplus X2 \oplus H1_i \oplus H2_i) \\ H2_{i-1} &= S \oplus X1 \oplus D(X1, S \oplus X1) \oplus H2_i\,. \end{aligned}$$

A common solution for $H2_{i-1}$ can be obtained by choosing S and selecting 2^{32} random values of $X2$ and $X1$ to evaluate the first and second equation respectively. The probability for a matching value for $H2_{i-1}$ will then be sufficiently large (cf. section 2.5.1.3). Finally $H1_{i-1}$ can be calculated from (5.2). As this attack yields only random IV, it can only produce a pseudo-preimage, or a pseudo-collision. Producing a preimage with this attack requires 2^{81} operations (cf. proposition 2.2), which is still completely infeasible. Finally it is noted that both equations become equivalent if and only if $H1_i = H2_i = 0$. In this case a pseudo-preimage can be found with a single decryption and a preimage in 2^{64} operations, by solving:

$$E(X_i, X_i \oplus H1_{i-1}) \oplus X_i \oplus H1_{i-1} \oplus H2_{i-1} = 0\,,$$

with $X1_i = X2_i = X_i$. If one chooses X_i and $H1_{i-1}$, this equation yields a random $H2_{i-1}$ (pseudo-preimage attack); a preimage can be found if the correct value of $H2_{i-1}$ is obtained.

The improved scheme takes a different approach: the equations are much simpler, but the construction of collisions or of a preimage is made more difficult by inserting additional redundancy to the message: one adds the message blocks $X_{t+1} = \bigoplus_{i=1}^{t} X_i$ and $X_{t+2} = \left(\sum_{i=1}^{t} X_i\right) \bmod 2^k - 1$. The iteration procedure can be described as follows:

$$\begin{aligned} T1_i &= E(X1_i, H1_{i-1}) \\ T2_i &= E(X2_i, T1_i \oplus H2_{i-1}) \\ H1_i &= H1_{i-1} \oplus H2_{i-1} \oplus T2_i \\ H2_i &= H1_{i-1} \oplus H2_{i-1} \oplus T1_i\,. \end{aligned}$$

For this algorithm pseudo-collisions were reported as well in [227]. However, only under special conditions these collisions are real collisions. It is sufficient to construct key collisions for a plaintext IV, i.e., a pair of keys (K_1,K_2) such that $E(K_1, IV) = E(K_2, IV)$ (cf. section 2.5.2.6). If the second initial value IV_2 is equal to $E(K_1, P) \oplus P$ and the first two message blocks are K_1 and K_2, swapping those two blocks does not affect the hashcode. The restriction is that the attacker should be able to choose the second half of the initial value. However, if a random initial value is given the probability that this attack will work is of the order of 2^{-64}.

Without redundancy, the scheme has several weaknesses. Finding a preimage with chosen $H1_{i-1}$ requires only 2^{33} operations. This follows from the equations:

$$H1_i \oplus H2_i = E(X1, H1_{i-1}) \oplus E(X2, H2_i \oplus H1_{i-1})$$
$$H2_{i-1} = H2_i \oplus H1_{i-1} \oplus E(X1, H1_{i-1})\,.$$

After selecting $H1_{i-1}$, one can solve the first equation for $X1$ and $X2$ with a birthday attack. The value of $H2_{i-1}$ follows then from the second equation. A preimage can be found in 2^{57} encryptions, but with a success probability of only 2^{-16}: first $X1$ is determined from the second equation, and subsequently $X2$ can be determined from the first equation. In order to find a collision, one has to solve the following equations for $(X1, X2)$ and $(X1', X2')$:

$$E(X2, E(X1, H1_{i-1}) \oplus H2_{i-1}) = E(X2', E(X1', H1_{i-1}) \oplus H2_{i-1})$$
$$E(X1, H1_{i-1}) = E(X1', H1_{i-1})\,.$$

With a birthday attack, key collisions for the second and subsequently the first equation can be found in only 2^{34} encryptions. However, because of the additional redundancy, this is not sufficient for a collision for the complete hash function. From these equations it can be seen that a pseudo-collision or a second pseudo-preimage can even be found with only 2 decryptions, in which $X1'$ and $X2'$ can be chosen freely. If DES is used as the underlying block cipher, one can again make use of fixed points. Here one will choose blocks $X1_1$ and $X2_1$ such that $H1_1$ is a fixed point for a weak key; this requires on average 2^{29} operations (cf. section 5.3.1.4). If $X1_2$ is the corresponding weak key, one finds again that $H2_2 = H2_1$. Note that here a single block will be sufficient to produce a collision for $H1_2$, since no restrictions are imposed on $X2_2$.

However, D. Coppersmith has recently published in [59] an attack to produce collisions for the second scheme with redundancy in about 2^{39} operations. The idea is to produce key collisions as indicated above for 43 values of i. This means that one has 43 pairs $(X1_i, X1_i')$ and 43 pairs $(X2_i, X2_i')$, where each alternative can be replaced by the other one without affecting the hashcode. This yields in total 2^{86} possible messages. Then one chooses X_{t+1}; in this way only $2^{86}/2^{56} = 2^{30}$ combinations will remain. They can be described as a 30-dimensional affine subspace of $GF(2^{86})$. Within this subspace, one has a sufficiently large probability of finding two matching values of X_{t+2}. Again this attack can be extended to a correcting block attack (cf. section 2.5.2.4). With 2 blocks one can make the chaining variables for both messages equal, from which it follows that in total 88 correcting blocks are necessary.

The scheme by Brown, Pieprzyk, and Seberry A variation on the first scheme of J.-J. Quisquater and M. Girault was proposed by L. Brown, J. Pieprzyk, and J. Seberry, the designers of the block cipher LOKI [33], who called it LOKI Double Hash Function (DBH). They took the scheme from the abstracts of Eurocrypt'89 [268], and modified the two keys with the chaining variables:

$$T1_i = E(H1_{i-1} \oplus X1_i, H1_{i-1} \oplus X2_i) \oplus X2_i$$

$$
\begin{aligned}
T2_i &= E(H2_{i-1} \oplus X2_i, T1_i \oplus H2_{i-1} \oplus X1_i) \oplus X1_i \\
H1_i &= H1_{i-1} \oplus H2_{i-1} \oplus T2_i \\
H2_i &= H1_{i-1} \oplus H2_{i-1} \oplus T1_i \,.
\end{aligned}
$$

This scheme is also vulnerable to a complementation attack, if DES, LOKI, or LOKI91[2] are used as the underlying block cipher. Moreover, weaknesses of LOKI (cf. section 2.5.4) can be exploited to produce pseudo-collisions [22] and even collisions [82].

In [183] an attack that is independent of the block cipher was described to find a pseudo-preimage in 2^{33} operations. The scheme is simplified by means of several linear transformations. We will show however that it follows directly from the equations that this attack applies.

Starting from (again the index i will be omitted in $X1_i$ and $X2_i$)

$$
\begin{aligned}
H1_i &= H1_{i-1} \oplus H2_{i-1} \oplus X1 \oplus E(X2 \oplus H2_{i-1}, X1 \oplus H1_{i-1} \oplus H2_i) \\
H2_i &= H1_{i-1} \oplus H2_{i-1} \oplus X2 \oplus E(X1 \oplus H1_{i-1}, X2 \oplus H1_{i-1}) \,,
\end{aligned}
$$

one chooses $S = X1 \oplus H1_{i-1}$, which results in

$$
\begin{aligned}
H2_{i-1} &= H1_i \oplus S \oplus E(X2 \oplus H2_{i-1}, S \oplus H2_i) \\
H2_{i-1} &= H2_i \oplus X2 \oplus H1_{i-1} \oplus E(S, X2 \oplus H1_{i-1}) \,.
\end{aligned}
$$

A solution for $H2_{i-1}$ can be found with a birthday attack: choose 2^{32} values of $X2 \oplus H2_{i-1}$ and 2^{32} values of $X2 \oplus H1_{i-1}$, and look for a matching value. Once a solution for $H2_{i-1}$ is determined, one finds $X2$, $H1_{i-1}$, and finally $X1$. In a similar way, both equations will become equivalent if and only if $H1_i = 0$. In that case one can find a pseudo-preimage with a single encryption, while a preimage requires 2^{65} encryptions.

The attack by D. Coppersmith [59] that exploits the existence of weak keys and the corresponding fixed points can be extended to this scheme. Note that LOKI has more fixed points than DES, and LOKI91 has the same number of fixed points as DES (cf. section 2.5.4.3). If one chooses $X1_i$ such that $H1_{i-1} \oplus X1_i$ is a weak key, and $X2_i$ such that $H1_{i-1} \oplus X2_i$ is a corresponding fixed point, one obtains again that $H2_i = H2_{i-1}$. The rest of the attack is similar to the attack on the first Quisquater-Girault scheme.

The scheme by Preneel, Govaerts, and Vandewalle This scheme was published in [253]:

$$
\begin{aligned}
H1_i &= c_3 \left[X1_i, E^{\oplus\oplus}\left(c_1\left[X1_i, X2_i\right], c_2\left[H1_{i-1}, H2_{i-1}\right]\right)\right] \\
H2_i &= c_4 \left[X2_i, E^{\oplus\oplus}\left(c_2\left[X1_i, H1_{i-1}\right], c_1\left[X2_i, H2_{i-1}\right]\right)\right] \,.
\end{aligned}
$$

The functions c_1, c_2, c_3, and c_4 have to satisfy the following conditions:

1. they have to compress two n-bit variables to one n-bit variable,
2. their result must be uniformly distributed,

[2]Note that no hash mode was specified for LOKI91 [34].

3. at least one of their output bits must change if a single input bit is modified.

The choice of particular functions thwart attacks that exploit special properties of the block cipher E. A different choice for c_1 and c_2 and for c_3 and c_4 can avoid certain symmetries. A possible choice for the c_i is the function $E^{\oplus}()$, resulting in a rate of 4. A second proposal is to choose the addition modulo 2 for all c_i. This results in the following scheme:

$$H1_i = X1_i \oplus H1_{i-1} \oplus H2_{i-1} \oplus E(X1_i \oplus X2_i, H1_{i-1} \oplus H2_{i-1})$$
$$H2_i = X2_i \oplus H1_{i-1} \oplus H2_{i-1} \oplus E(X1_i \oplus H1_{i-1}, X2_i \oplus H2_{i-1})\,.$$

This scheme is not vulnerable to attacks based on the complementation property. Its analysis is simpler because the two operations are independent: the goal was to allow for an efficient parallel implementation. Several attacks, mainly attacks to find collisions were tried during the design phase (and many other schemes were rejected), but later it was shown that the parallel operation introduces several weaknesses. A first attack finds a pseudo-preimage with a single decryption: the two equations will be identical if and only if $X1_i = H1_{i-1} \oplus H1_i \oplus H2_i$, and $X2_i = H1_{i-1}$. In this case the relation between the chaining variables is given by:

$$E(H1_i \oplus H2_i, H1_{i-1} \oplus H2_{i-1}) \oplus H2_{i-1} = H2_i\,.$$

For a chosen value of $H2_{i-1}$, one obtains quickly $H1_{i-1}$. If the attacker wants to find a preimage, he will choose $H2_{i-1}$ equal to the specified value. If he is very lucky, he obtains the correct value for $H1_{i-1}$ (the probability of this event is 2^{-64}).

A different method will find a preimage in 2^{64} operations: choose $X1_i \oplus X2_i$, then $X1_i$ (and hence $X2_i$) can be obtained easily from the first equation. A match for the second equation will then be found with probability 2^{-64}. Unfortunately, together with proposition 2.5 this fact can be used to produce a collision in only 2^{33} operations. The values of $X1_i$ and $X2_i$ corresponding to a choice of $H1_i$ can be obtained with a single decryption ($s = 0$), which means that $n' = 64$. If this is repeated 2^{33} times, a matching value for $H2_i$ will be found with very large probability (cf. section 2.5.1.3).

A weaker attack on this scheme, that requires 2^{33} operations for a pseudo-preimage was described in [183]. With the substitutions $X = X_1 \oplus X_2$, $S = X_2 \oplus H2_{i-1}$, and $T = H1_{i-1} \oplus H2_{i-1} \oplus X_1 \oplus X_2$, one obtains the following simplified equations (omitting the subscript i for $X1$ and $X2$):

$$H1_i = X2 \oplus T \oplus E(X, X \oplus T)$$
$$H2_i = X2 \oplus T \oplus X \oplus E(T \oplus S, S)\,.$$

It is possible to eliminate X_2, by adding the two equations, which yields:

$$E(X, X \oplus T) \oplus X \oplus H1_i = E(T \oplus S, S) \oplus H2_i\,.$$

This equation can be solved easily by fixing T, and subsequently choosing 2^{32} values for X and S to find a match. Finally $X2$ can be computed from one of the above

equations. With the expressions for X, S, and T one finds X_1, $H2_{i-1}$, and $H1_{i-1}$ respectively.

The main reason why these weaknesses were not identified in the design stage, is that we concentrated on protection against collision attacks: we believed that a preimage attack would automatically be hard. This scheme however shows how a partial preimage attack can be used to produce collisions based on proposition 2.5. The lesson learned from this attack is that finding a solution $(X1_i, X2_i)$ for one of the equations should require at least 2^{32} operations (and preferably 2^{64}).

Finally it is mentioned that the attack by D. Coppersmith [59] that exploits the existence of weak keys and the corresponding fixed points can be extended to this scheme. If one chooses $X1_i$ such that $X1_i \oplus H1_{i-1}$ is a weak key, and $X2_i$ such that $X2_i \oplus H2_{i-1}$ is a corresponding fixed point, one obtains that $H2_i = H1_{i-1}$. The rest of the attack is similar to the attack on the first Quisquater-Girault scheme.

Conclusion To conclude this section, the attacks on the three schemes (excluding the second scheme by J.-J. Quisquater and M. Girault) will first be compared in table 5.11. For the two first schemes, the following remarks have to be made: if DES, LOKI or LOKI91 are used as the underlying block cipher, collisions can be found immediately with the complementation property, and for special values of $H1_i$ and $H2_i$ (depending on the scheme), a pseudo-preimage can be found in time $O(1)$, and a (2nd) preimage can be constructed in time 2^{64}. Moreover for the three schemes a collision can be produced in about 2^{33} encryptions with an attack based on fixed points.

	Q-G	LOKI-DBH	P-G-V
(2nd) preimage	$2 \cdot 2^{81}$	$2 \cdot 2^{81}$	$2 \cdot 2^{64}$
pseudo-preimage	$2 \cdot 2^{32}$	$2 \cdot 2^{32}$	1
collision (fixed IV)	$2 \cdot 2^{64}$	$2 \cdot 2^{64}$	$2 \cdot 2^{33}$
collision (random IV)	$2 \cdot 2^{32}$	$2 \cdot 2^{64}$	$2 \cdot 2^{33}$
pseudo-collision	1	$2 \cdot 2^{32}$	1

Table 5.11: Number of operations and storage for preimage and collision attacks on the three schemes studied in this section.

The main problem with these schemes with rate 1 is that their analysis is very involved: one can try to verify that all 4 criteria of section 2.4.3 are satisfied, but there does not seem to be a way to prove this. On the other hand, one should specify against which attacks one wants to be protected: an attack on a single round implies solving 2 simultaneous equations. In case of a preimage attack, one can choose 2 variables ($X1_i$ and $X2_i$), while in the case of a pseudo-preimage, one can choose 4 variables ($X1_i$, $X2_i$, $H1_{i-1}$, and $H2_{i-1}$). For a collision, the equations are different and will contain at least 4 $E()$ operations. If one searches for a collision, one can choose 4 variables ($X1_i$, $X2_i$, $X1'_i$, $X2'_i$), for a collision with a random IV one can additionally choose

$H1_{i-1}$ and $H2_{i-1}$, and in the case of a pseudo-collision one can also select $H1'_{i-1}$ and $H2'_{i-1}$, hence one has in total 8 variables.

We believe that it is not possible to come up with an efficient scheme with rate 1 that is ideally secure against all these attacks. One suggested line of research is to come up with a synthetic approach like for the single length hash function, in order to find the best scheme under these constraints. The problem here is not easy, as the number of possibilities is equal to $\left(2^4-1\right)^6 = 11,390,625$ if no interaction between the chains is considered, and $\left((2^4-1)(2^5-1)\right)^3 = 100,544,625$ with interaction. The problem size can of course be reduced significantly by excluding trivially weak schemes and by considering linear transformations of input and output, as suggested in [183]. The only result one can hope to achieve is to identify certain patterns that lead to specific weaknesses. A first attempt could be the following:

- Individual functions should be complete.
- It should not be possible to completely separate the chains.
- One of both equations should not be easily solvable for the two data blocks.
- The equations should not be easily solvable if symmetry between the two parts is introduced.
- It should not be possible to make use of key collisions: a simple case is where a message block is just entered once in the key port. It seems that adding some redundancy is not sufficient to thwart this attack.
- It should not be possible to exploit the complementation property and the existence of weak keys and their corresponding fixed points. On the other hand one should note that it seems better to block these possibilities: in the case of DES it is sufficient to fix two key bits (cf. MDC-2). Moreover, for any scheme of this type one can come up with a variant of a complementation property (such as $E(\overline{K}, P) = \overline{E(K, P)}$) that will yield collisions.
- It should not be easy to go backwards (pseudo-preimage).

It would be very interesting to develop a computer program that identifies these patterns in all possible schemes. For several criteria this is certainly rather complicated. Moreover one should keep in mind that it will only be possible to verify attacks that have been considered by the author of the program. The complexity of the evaluation of this type of schemes suggests that further work has to be done in this area.

A second line of research is to check whether the most efficient schemes can be made secure against all these attacks by adding a limited number of redundancy blocks, which influences the rate only marginally. At the end of the message, one can add the message length (as discussed in section 2.4.1), and the addition modulo 2 and/or modulo 2^k of the message bits, as suggested by J.-J. Quisquater and M. Girault. It has however become clear that this will not help against collision attacks. At the beginning of the message, one can derive the first message block from IV, or $X_1 = g(IV)$ (cf. section 2.4.1): then finding a pseudo-preimage is not easier than finding a preimage.

5.3.3 Size of key equals twice the block length

Some block ciphers have been proposed for which the key size is approximately twice the block length. Examples in this class are FEAL-NX [228] (a FEAL version with a 128-bit key) and PES/IDEA [181, 182, 184]. Triple DES with 2 keys has a key size of 112 bits and a block length of 64 bits and could hence also be considered to belong to this class. Note that xDES[1] discussed in section 5.3.2.2 is not of this type, as the key size is equal to 168 bits and the block length is 128 bits. Again one has to make a distinction between single and double length hash functions.

5.3.3.1 Size of hashcode equals block length

A scheme in this class was proposed by R. Merkle in [211]. It can also be classified as "non-invertible chaining":

$$f = E(H_{i-1} \parallel X_i, IV)\,.$$

A parameterized version was proposed by the same author [212]:

$$f_j = E(IV' \parallel H_{i-1} \parallel X_i \parallel i \parallel j\ , IV)\,.$$

Here IV and IV' are two constants that can be zero, and i and j are numbers with about 64 bits: j is a parameter of the hash function and i results in a different function for every iteration. An alternative scheme was suggested by X. Lai and J. Massey in [183]:

$$f = E(H_{i-1} \parallel X_i, H_{i-1})\,.$$

These constructions can only yield a CRHF if the block length is larger than 128 bits (R. Merkle suggested 100 bits in 1979), and if the key size is equal to 256 bits for the first and third scheme and 384 bits for the second scheme. For smaller block lengths, a OWHF can be obtained. The security depends strongly on the key scheduling of the cipher. If triple DES is used with three different keys, which is a block cipher with a 168-bit key size and a 64-bit block length, this construction can only yield a OWHF. This scheme would have a rate of 1.85 (the size of the chaining variable is 64 bits and 104 message bits are processed after 3 encryptions), which is not very efficient. In this case the first scheme by R. Merkle can be broken with a key collision attack on double DES (cf. section 2.5.2.6). If H_{i-1} is used as key for the first DES and as first part of the key of the second DES, a key collision search can be applied to the second and third DES, under the constraint that the first 8 bits of the second key agree with the corresponding value for H_{i-1}.

5.3.3.2 Size of hashcode equals twice the block length

In order to obtain a CRHF based on a 64-bit block cipher, a different construction is required. The first two schemes in this class were recently proposed by X. Lai and J. Massey [183]. Both try to extend the Davies-Meyer scheme. One scheme is called

"Tandem Davies-Meyer", and has the following description:

$$\begin{aligned} T_i &= E(H2_{i-1} \| X_i, H1_{i-1}) \\ H1_i &= T_i \oplus H1_{i-1} \\ H2_i &= E(X_i \| T_i, H2_{i-1}) \oplus H2_{i-1} \,. \end{aligned}$$

The second scheme is called "Abreast Davies-Meyer":

$$\begin{aligned} H1_i &= E(H2_{i-1} \| X_i, H1_{i-1}) \oplus H1_{i-1} \\ H2_i &= E(H2_{i-1} \| X_i, \overline{H2_{i-1}}) \oplus H2_{i-1} \,. \end{aligned}$$

Both schemes have a rate equal to 2, and are claimed to be ideally secure, or finding a pseudo-preimage takes 2^{2n} operations and finding a collision takes 2^n operations. The security of this scheme has to be evaluated: e.g., certain weaknesses might occur depending on the underlying block cipher.

5.3.4 A new scheme based on a block cipher with fixed key

In this section a new hash function will be presented that offers a trade-off between security and performance. The scheme was published at Auscrypt'92 [262]. After motivation of the most important design principles, the function will be described, followed by a description of attacks, and a detailed security evaluation. Subsequently some extensions are discussed, and the results are summarized. Note that in the previous sections several schemes with a fixed key have already been discussed, since it was more natural to treat them there.

5.3.4.1 Background and design principles

It is recalled that the rate of the scheme indicates the number of encryption operations to process a number of bits equal to the block length. An encryption operation consists of two steps: first the installation of a new key and subsequently an encryption with that key. Several reasons can be indicated to avoid a modification of the key during the hashing process:

Performance: in general, the key scheduling is significantly slower than the encryption operation. A first argument to support this is that the key scheduling can be designed as a very complex software oriented process to discourage exhaustive attacks. Here software oriented means that the variables are updated sequentially, which reduces the advantages of a parallel hardware implementation. The advantage of this approach is that the key can remain relatively small (e.g., 64 bits), while an exhaustive attack is still completely infeasible. Even when the key schedule is simple, it can be harder to optimize its implementation. For DES software, the fastest version (code size 100 Kbytes) for the IBM PS/2 Model 80 (16 MHz 80386) runs at 660 Kbit/s without key change and 146 Kbit/s with key change (a factor 4.5 slower). If the code size is increased to 200 Kbytes, a

speed of 200 Kbit/sec is obtained (a factor 3.3 slower). For smaller versions this factor is lower bounded by 2.5. Moreover, encryption hardware is in general not designed to allow fast modification of the key. A key change can cause loss of pipelining, resulting in a serious speed penalty.

Security: another advantage of keeping the key fixed is that an attacker has no control at all over the key. Hence attacks based on weak keys can be eliminated completely in the design stage.

Generality: the hash function can be based on any one-way function with small dimensions (e.g., 64-bit input and output).

The following design principles form the basis of the system:

Atomic operation: the one-way function that is used will be $E^{\oplus}(K, X)$. It has shown to be very useful for the single length hash functions and was also used in MDC-2, MDC-4 and by R. Merkle in his construction for a double length hash function (cf. section 5.3.2.2).

Parallel operation : the one-way function will be used more than once, but the function can be evaluated in parallel; this opens the possibility of fast parallel hardware and software implementations. It is also clear that a scheme in which several instances are used in a serial way is much harder to analyze.

Trade-off between memory, rate, and security level : the rate of the system can be decreased at the cost of a decreasing security level; it will also be possible to decrease the rate by increasing the size of the hashcode. This could also be formulated in a negative way, namely that the security level will be smaller than what one would expect based on the size of the hashcode. Observe that this property is also present in a limited way in MDC-2, MDC-4, and the third scheme of R. Merkle (cf. section 5.3.2.2).

The basic function is not collision resistant: the construction is not based on a collision resistant function, because it can be shown that this would not yield an acceptable performance (the efficiency will decrease with a factor 4 or more). This means that producing collisions for different values of the chaining variable or producing pseudo-collisions is easy. Based on intuitive arguments indicated in section 2.4.3, the data input to the elementary operation will be protected more strongly than the chaining variable.

5.3.4.2 Description of the new scheme

One iteration step consists of k parallel instances of the one-way function, each parameterized with a fixed and different key. In the following, these instances will be called 'blocks'. These k keys should be tested for specific weaknesses: in the case of DES it is recommended that all 16 round keys are different, and that no key is the complement of any other one. If the block length of the block cipher is equal to n, the total number of input bits is equal to kn. These inputs are selected from the x bits of the data X_i and from the h bits of the previous value of the chaining variable H_{i-1}.

Every bit of X_i will be selected α times, and every bit of H_{i-1} will be selected β times which implies the following basic relation:

$$\alpha \cdot x + \beta \cdot h = k \cdot n \,. \tag{5.3}$$

A simple consequence of this equation is that $h < kn/\beta$. The rate R of this scheme is given by the expression:

$$R = \frac{n \cdot k}{x} \,. \tag{5.4}$$

The following considerations lead to additional restrictions on α and β:

- it does not make sense to enter the same bit twice to a single block,
- the data bits should be protected in a stronger way than the chaining variable, which has been discussed as the fourth design principle.

This results in the following constraints:

$$2 \leq \alpha \leq k \quad \text{and} \quad 1 \leq \beta \leq \alpha \,. \tag{5.5}$$

In the largest part of the discussion, the parameter β will be limited to 1.

The output of the functions has also size kn. This will be reduced to a size of h by selecting only h/k bits from every output. Subsequently a simple mixing operation is executed, comparable to the exchange of left and right halves in MDC-2. The goal of this operation is to avoid that the hashing operation consists of k independent chains. If h is a multiple of k^2, this mixing operation can be described as follows. The selected output block of every function (consisting of h/k bits) is divided into k parts, and part j of block i is denoted with H^{ij} $(1 \leq i, j \leq k)$. Then $H^{ji}_{\text{out}} \longleftarrow H^{ij}_{\text{in}}$. Figure 5.5 depicts one iteration for the case $k = 4$ and $\alpha = 4$.

As has been discussed in section 2.4.1, a fixed initial value has to be chosen, the message length in bits has to be added at the end, and an unambiguous padding rule has to be specified.

It will become clear that the complexity of the scheme does not depend on α, but on the difference between k and α. Therefore, the following parameter is defined:

$$\phi \stackrel{\text{def}}{=} k - \alpha \,. \tag{5.6}$$

The next step in the design is the decision on how the data bits are to be distributed over the different blocks if $\phi > 0$. If $\beta > 1$ a similar construction can be applied to the chaining variables. The construction is obtained by considering the following attack. Let $\mathcal{S}$ be a subset of the blocks. For a given value of H_{i-1}, fix the data input of the blocks in $\mathcal{S}$, which means that the output of these blocks will be the same for both input values. Subsequently, match the remaining outputs with a birthday attack (cf. section 2.5.1.3). In order to maximize the effort for this attack, it is required that after fixing the input of the blocks in $\mathcal{S}$, the number of bits that can still be freely chosen by an attacker is minimal (this will be explained in more detail in section 5.3.4.3). This number will be denoted with $A(\mathcal{S})$.

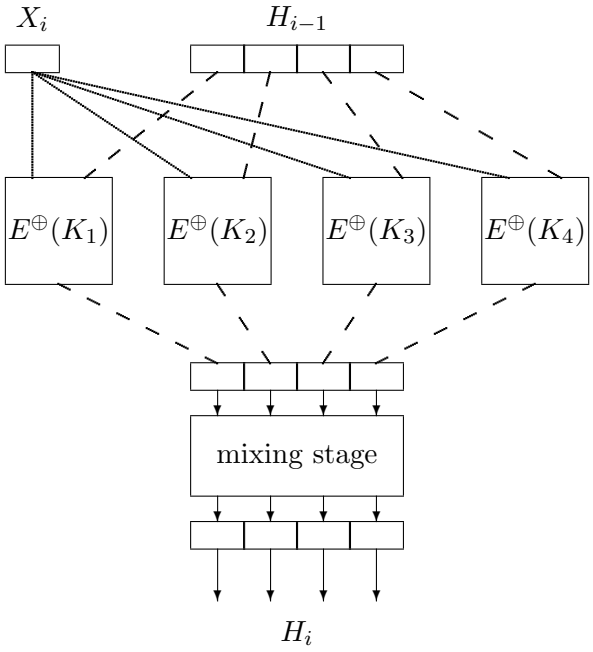

Figure 5.5: One iteration of the new hash function proposal.

Theorem 5.1 *If the data is divided into $\binom{k}{\phi}$ parts, and every part goes to a different combination of $\binom{k}{\phi}$ blocks, an optimal construction is obtained. Let A_s be defined as $\max_{|\mathcal{S}|=s} A(\mathcal{S})$, then this construction results in the following expression for A_s:*

$$A_s = \frac{\binom{k-s}{k-\phi}}{\binom{k}{\phi}} \cdot x = \frac{\binom{\phi}{s}}{\binom{k}{s}} \cdot x \qquad 1 \le s \le \phi \tag{5.7a}$$

$$= 0 \qquad \textit{else} . \tag{5.7b}$$

This construction is optimal in the sense that for a given $s =\mid \mathcal{S} \mid A(\mathcal{S})$ is independent of $\mathcal{S}$, and for any other construction there will be an s (with $1 \le s \le k$) such that $A'_s \ge A_s$. If equality holds for all values of s both constructions are isomorphic.

Proof: The optimality follows from the symmetry of the allocation. Because of this optimality, in the following it can be assumed w.l.o.g. that the set $\mathcal{S}$ consists of the first s blocks. Define a_j as the number of bits that are used for the first time in block number j. It will be shown that

$$a_j = \frac{\binom{k-j}{k-\phi-1}}{\binom{k}{\phi}} \cdot x \qquad 1 \le j \le \phi + 1 \tag{5.8a}$$

$$= 0 \qquad \text{else.} \tag{5.8b}$$

Assume that the k blocks are numbered starting with 1, then the selection of α blocks out of k correspond to an index that is an r digit number in base $k+1$. In order to

make this index unique, it is specified that its digits are in increasing order. Sort the selection based on their index (also in increasing order). Divide the data input into $\binom{k}{\phi}$ parts that are numbered starting with 1. The bits of data part q will be used in the selection of which the index has number q in the sorted list. The first digit of the index indicates in which block this data part will be used first. It is now easy to see that the number of data parts that are used for the first time in block j ($1 \leq j \leq \phi+1$) is given by

$$\binom{k-j}{\alpha-1} = \binom{k-j}{k-\phi-1}.$$

As the number of bits in a single data part is given by $x/\binom{k}{\phi}$, the expression for a_s was proven.
Subsequently, A_s can be computed based on the following relation:

$$A_s = x - \sum_{j=1}^{s} a_j \,.$$

The expression for A_s will be proven by induction on s. It is clearly valid for $s = 1$:

$$a_1 = \frac{k-\phi}{k} \cdot x \quad \text{and} \quad A_1 = \frac{\phi}{k} \cdot x \,.$$

Assume that (5.7) is valid for s (with $s < \phi$). The expression for $s+1$ is then:

$$A_{s+1} = \frac{\binom{k-s-1}{k-\phi}}{\binom{k}{\phi}} \cdot x \,.$$

First A_s will be written as

$$A_s = x - \sum_{j=1}^{s+1} a_j = x - \sum_{j=1}^{s} a_j - a_{s+1} \,.$$

With the induction assumption, and the expression for a_{s+1} this can be reduced to

$$\frac{\binom{k-s}{k-\phi}}{\binom{k}{\phi}} \cdot x - \frac{\binom{k-s+1}{k-\phi-1}}{\binom{k}{\phi}} \cdot x \,.$$

Using the basic identity in Pascal's triangle, this reduces to the correct expression for A_{s+1}. In order to complete the proof, it is sufficient to note that if $s > \phi$, it follows from the expression for a_s that A_s is equal to 0. ∎

An example of the allocation will be given for the case $k = 4$. All relevant parameters are summarized in table 5.12.

ϕ	s	1	2	3	4	block	1	2	3	4
0	A_s/x	0	0	0	0	# parts = 1				
	a_s/x	1	0	0	0	allocation	1	1	1	1
1	A_s/x	0.25	0	0	0	# parts = 4				
	a_s/x	0.75	0.25	0	0	allocation	123	124	134	234
2	A_s/x	3/6	1/6	0	0	# parts = 6				
	a_s/x	3/6	2/6	1/6	0	allocation	123	145	246	356

Table 5.12: Overview of all parameters of the optimal allocation for the case $k = 4$ and $\phi = 0$, 1, and 2.

5.3.4.3 Attacks on the scheme

For the time being, it is not possible to give a security proof for the scheme. This disadvantage is shared with all other schemes (including MDC-2 and MDC-4); the only exceptions are the Merkle schemes. The difference with the other schemes is that the system is parameterized, and that the security level depends on the size of the hashcode h. In the following, a number of attacks will be considered that are faster than the birthday attack. This is possible as it is not the case that all output bits depend in a strong way on the inputs of a single iteration step. Indeed, the data only enter α blocks, and hence if $\alpha < k$, the output of $k-\alpha$ blocks does not depend on these input bits. Even if $\alpha = k$, the fact that $\beta = 1$, implies that the diffusion of the H_{i-1} is limited to one block. Note that this property is shared with MDC-2. This limited dependency is solved by increasing the size of the hashcode. The required number of bits for the hashcode is estimated from studying a set of birthday attacks that exploit the structure of the scheme. The generality of the proposed attacks should form an argument for the security. However, for the time being it is not possible to prove that there does not exist a more sophisticated attack. The advantage of the scheme is that the security level can always be increased at the cost of an increased memory.

It will also be checked whether this type of functions is one-way, and the applicability of differential attacks will be discussed briefly. By construction the scheme is not vulnerable to attacks based on weak keys or based on the complementation property.

For more details on birthday attacks and collisions, the reader is referred to Appendix B. Before discussing the attacks in detail, an additional remark will be made on the number of operations to generate a collision. Assume one has a random function with B output bits and A input bits that can be chosen arbitrarily. The function might have C inputs bits that can not be chosen freely; these input bits will be called parameters. If a collision is to be produced for this function for a certain value of a parameter, i.e., two inputs that result in the same output, two cases have to be distinguished:

$A > B/2$: in this case producing a collision requires $2^{B/2}$ function evaluations.

$A < B/2$**:** in this case, the number of expected collisions after a single trial is equal to $p = (2^A)^2/2^B$. This process will be repeated for several values of the parameter (it is assumed that C is sufficiently large). The expected number of trials is given by $1/p$ and the effort for a single trial is 2^A function evaluations. Hence the expected number of function evaluations is equal to $2^B/2^A$, which is always larger than $2^{B/2}$.

Note that abstraction has been made from the constant in the exponent. The number of operations is graphically depicted in figure 5.6. Note that this argument can be

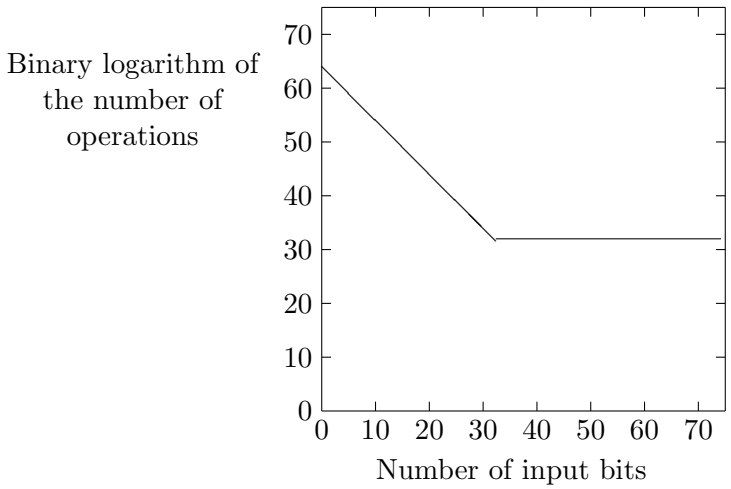

Figure 5.6: The binary logarithm of the number of function evaluations as a function of the number of input bits for the case $B = 64$.

simply extended if one needs more than one collision. However, if multiple collisions are required, the behavior will be different. A detailed discussion of the number of operations is given in section 4 of Appendix B. It should be added that if a k-fold collision is required, with k large (say > 32), the number of operations will grow much faster if the number of inputs in a single trial is smaller than required.

Four types of birthday attacks that exploit the structure of the scheme will be discussed. They are only valid if $\phi > 0$. More details on the relative efficiency of the different attacks together with a discussion of the case $\phi = 0$ are given in the next section.

Attack A: this attack has already been discussed when introducing the optimal allocation of the data bits. It consists of fixing the input bits to the first s blocks and matching the output of the remaining $k - s$ blocks with a birthday attack. The number of output bits of these blocks is denoted with B_s. It is clear that

$$B_s = \frac{k-s}{k} \cdot h \,.$$

The binary logarithm of number of operations for this attack is now given by the following expression:

$$\frac{B_s}{2} + 1 + \log_2(k-s) \qquad \text{if } A_s \geq \frac{B_s}{2} \tag{5.9a}$$

$$B_s - A_s + 1 + \log_2(k-s) \qquad \text{if } A_s < \frac{B_s}{2}\,. \tag{5.9b}$$

With the expressions (5.3), (5.7), and (5.9) for respectively x, A_s, and B_s, this can be written as:

$$\frac{k-s}{2k} + 1 + \log_2(k-s) \text{ if } A_s \geq \frac{B_s}{2}$$

$$h\left[\frac{k-s}{k} + \frac{\beta}{k-\phi}\cdot\frac{\binom{k-s}{k-\phi}}{\binom{k}{\phi}}\right] - \frac{k}{k-\phi}\cdot\frac{\binom{k-s}{k-\phi}}{\binom{k}{\phi}}\cdot n + 1 + \log_2(k-s) \text{ if } A_s < \frac{B_s}{2}\,.$$

Here the term 1 comes from the drawings within one set (equation (B.4) in Appendix B), and the term $\log_2(k-s)$ appears because the match has to be performed on $k-s$ blocks. Note that this term could be eliminated by first looking for a match for the first block, and subsequently looking for matches for the next blocks in this set, but this would require more memory. For every value of s, first the relative size of A_s and $B_s/2$ have to be compared. Subsequently this yields a *linear relation* between memory h and security level S. This will be the case for almost all subsequent attacks. Because of expressions (5.3) and (5.4), there will be a hyperbolic relation between the rate R and the security level S.

As an example, the case $k = 5$, $\phi = 3$, $\beta = 1$, and $n = 64$ will be considered. The results are summarized in table 5.13. It can be seen that the most effective attack is the attack with the maximal value of s, namely $s = \phi$. It will not be proven that this is always true, as more efficient attacks have been developed.

Attack B: a more effective attack consists of generating a multiple collision for the first s blocks. The number of collisions is denoted with 2^c; it does not have to be a power of two, but this will simplify notation. In the next step, one has $A_s + c$ degrees of freedom available to match the remaining $k-s$ blocks. This attack has already two parameters: s and c. Moreover, the logarithm of the number of operations to generate a 2^c fold collision is a non-linear function of c, for small values of c ($1 \leq c \leq 15$, cf. Appendix B). Therefore it was decided to write a computer program to perform a numerical calculation. It was also extended to explore the efficiency of the subsequent more complicated attacks. A problem that should not be overlooked is the following: for large values of c, an attacker needs about $h\frac{s}{k} + c$ degrees of freedom to produce such a multiple collision. In most cases, there are not that many data bits that enter the first block(s). However, one can assume that an attacker can also introduce variations in the previous iteration steps. If a collision resistant function would have been

s	1	2	3
A_s/x	6/10	3/10	1/10
B_s/h	4/5	3/5	2/5

s	1	2	3
$A_s < B_s/2$ if	$h > 137.1$	$h > 106.7$	$h > 64$
$A_s \geq B_s/2$	$h\frac{2}{5} + 3$	$h\frac{3}{10} + 2.6$	$h\frac{1}{5} + 2$
$A_s < B_s/2$	$h\frac{11}{10} - 93$	$h\frac{3}{4} - 45.4$	$h\frac{9}{20} - 14$
S for $h = 128$	54.2	50.6	43.6
S for $h = 192$	118.2	98.6	72.4

Table 5.13: Values of A_s/x and B_s/h and relations between the security level S and the size of the chaining variable h for the case $k = 5$, $\phi = 3$, $\beta = 1$, and $n = 64$. The number of operations for the three different attacks is indicated for the case $h = 128$ and $h = 192$.

designed, this assumption would not have been valid. In that case any attack has to produce a collision for a single iteration step.

Attack C: under certain circumstances, attack B can be optimized by exploiting the block structure: first a 2^{c_1}-fold collision is produced for the output of the first block (possibly using variations in previous rounds), subsequently a 2^{c_2}-fold collision is produced for the output of the second block. This continues until a 2^{c_s}-fold collision is produced for the output of block s. Finally the last s blocks are matched with a birthday attack with $A_s + c_s$ degrees of freedom. The attack is optimized with respected to the parameter c_s; the choice of c_s fixes the other c_i's as follows: in order to produce a 2^{c_s}-fold collision for block s, $h/k + c_s$ trials are necessary (assuming c_s is not too small). There are only a_s bits available at the input of block s, which means that a $2^{c_{s-1}}$-fold collision will be required at the output of block $s-1$, with $c_{s-1} = h/k + c_s - a_s$. If c_s is small, $h/k + c_s$ can be replaced by the number of operations to produce a c_s-fold collision. This procedure is repeated until the first block is reached. It is assumed that there are sufficient degrees of freedom available through variations in the previous iterations.

Attack D: This attack is different from attacks B and C because it makes a more explicit use of interaction with the previous iteration step. It is based on the observation that if h is significantly smaller than the value that is being evaluated, it should be easy to produce collisions or even multiple collisions for H_{i-1}. Therefore it should also be easy to produce multiple collisions for the first s' blocks of H_{i-1}, that contain $h' = s'/h$ bits. The data bits that enter the first s' block are now fixed, and this implies that the output of these blocks will be identical. From block $s' + 1$ on, the attack continues with a type C attack.

The problem that remains to be solved is how to evaluate the number of operations to produce a multiple collision for the first s' blocks of H_{i-1}. This number

is estimated by calculating the number of operations for a collision S'; in general S' will be smaller than $h'/2$. Subsequently, the expression for the number of operations for a multiple collision is used where the size of the block is replaced by the effective block length $2S'$. The number S' can be found by comparing the efficiency of attacks A, B, C, and D for a scheme with the same configuration as the original one (this means the number of data bits has not been modified), but with h replaced by h'. If attack D is optimal, the program works in a recursive mode. It has been verified empirically that for values of $k \leq 6$ increasing the number of recursion steps above 2 does not influence the result. This can be explained by the fact that the value of h' gets smaller very quickly. If the recursion is halted, this means essentially that an attacker gets the multiple collisions for the first s' blocks for free.

Another problem is whether the function is one-way. This means that it should be hard to find the preimage for a given output. A first remark is that the input should be at least S bits long. If this is not the case for a single iteration, it should be specified that more iterations are mandatory, in order to guarantee that the overall size of the input is at least S bits. A second remark is that the function is not complete, i.e., not every output bit depends on every data bit. The number of output bits that is influenced by a single input bit is equal to

$$\frac{(k-\phi)}{k} \cdot h\,. \tag{5.10}$$

It is clear that this implies that if $\phi > 0$ producing a preimage for a given hashcode will require less than 2^h operations. A simple attack, comparable to attack A consists of looking for inputs that yield the correct output for the first s blocks and subsequently using the A_s bits that are left to match the output of the $k - s$ blocks. However, we believe that finding a preimage for a given hash value will always be harder than producing a collision, and the parameters will be selected to ensure that the number of operations to produce a collision is sufficiently large. This can be supported with an intuitive argument: for a security level of S bits it has been verified that every data bit influences at least S output bits. Again here one can remark that finding a pseudo-preimage will be much easier, as every bit of the chaining variable affects only h/k output bits.

Finally a short remark on differential attacks. The main reasons why differential attacks will not work if DES is used as the underlying block cipher is that there has not been found a good characteristic with an even number of rounds. Differential attacks on this scheme are harder because the attacker has no complete control over the plaintext, and because the keys can be selected in a special way to minimize the probability of iterative characteristics (which is not possible for MDC-2). On the other hand, the characteristic has only to hold for the subset of the output bits that has been selected. This means that in the last round the characteristic must not be completely satisfied. However, it should be noted that the success probability will decrease very fast if the characteristic is not satisfied in earlier rounds. Like in the case of MDC-2,

every data bit is used at least twice, which implies that a characteristic with a high probability is required for an attack faster than exhaustive search. Once a detailed scheme has been fixed, more work can be done on selecting the keys in such a way that differential attacks are less efficient.

5.3.4.4 A detailed study of the security level

In this section the security level of schemes with ϕ between 0 and 4 will be discussed. In order to keep the schemes practical, the value of k will be limited to 6. If $\phi = 0$ it can be shown that for practical values of the security level the schemes with small values of k are more efficient. For all other schemes, larger values of k would imply that there are too many small parts (recall that the number of parts is equal to $\binom{k}{\phi}$), which would increase the overhead of bit manipulations to an unacceptable level. Finally note that the restriction $\alpha \geq 2$ (5.5) is equivalent to $k \geq \phi + 2$.

For different values of ϕ, a linear or a piecewise linear relation between the security level S and the size of the hashcode h will be obtained. The expression for the rate can then be written as:

$$R = \frac{k - \phi}{1 - \frac{h}{kn}} . \tag{5.11}$$

If h is substituted by a linear expression in S, this yields a hyperbolic relation between R and S. In the practical calculations below, the block length n will be fixed equal to 64.

The case $\phi = 0$ In this case $\alpha = k$, which means that every data bit goes to every block. There are no problems in determining an optimal configuration of the scheme. As indicated above, at first sight none of above attacks applies, which means that the security level is equal to its upper bound $S = h/2$. Note however that for k even, attack B might be slightly more efficient. First one generates a 2^c fold collision for the first half of the output. For the case $c = 3$ and $h = 128$ this requires about 2^{58} operations. Subsequently a birthday attack with c degrees of freedom is launched against the second half of the output. This will require $2^{64-3+1+1}$ operations. This means an improvement with a factor 2. Note however that this is certainly equal to the error in all our results because of the limited accuracy of our models. The expression for the rate of this scheme reduces to

$$R = \frac{k}{1 - \frac{2S}{kn}} .$$

It can be seen that it becomes advantageous to increase k by one if the security level is given by

$$S = n \cdot \frac{k(k+1)}{2k+1} .$$

This means that $k = 4$ will be the optimal solution for a security level between 54.9 and 71.1 bits. A graphical representation of the relation between R and S is given in figure 5.7.

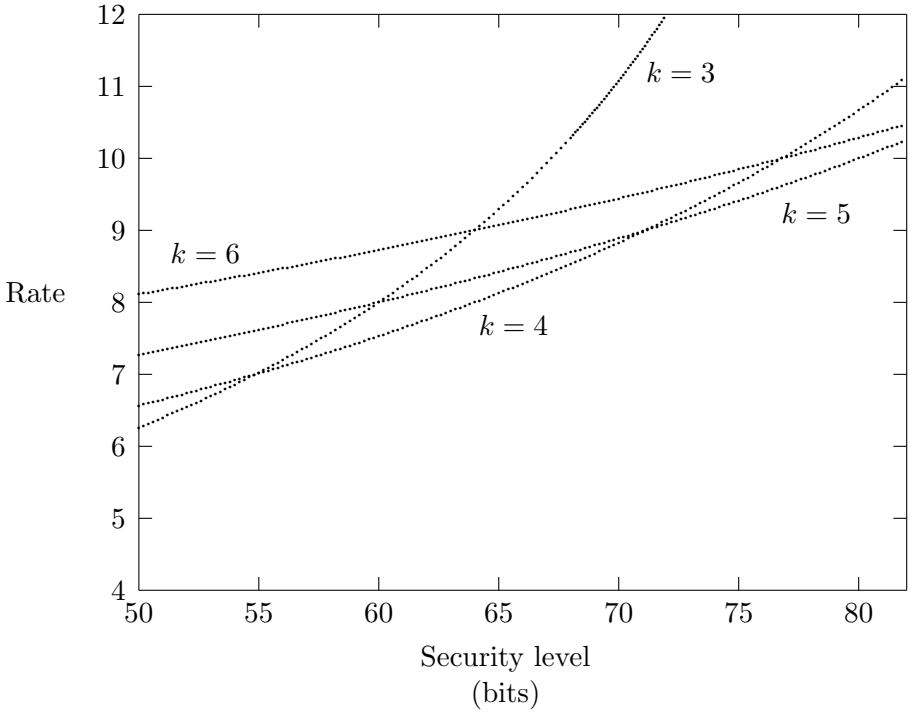

Figure 5.7: Relation between the rate R and the security level S for $\phi = 0$ and k between 3 and 6. The optimal value of k increases with S.

The case $\phi = 1$ In this case the data input is split into k parts, and every part goes to $k-1$ blocks. Computer calculation has shown that the most efficient attack is attack D with $s = s' = 1$. The number of operations can be approximated by the following expressions:

1. Number of operations to produce a collision for blocks 2 to k:

$$\frac{k-1}{k}\, h - \frac{x}{k} + 1 + \log_2(k-1) - c\,.$$

2. Number of operations to produce a 2^c-fold collision for the first block of H_{i-1}:

$$\frac{k-1}{k}\frac{h}{k} + 2(1 + \log_2(k-1)) + c\,.$$

The approximation lies in the fact that the logarithm of the number of operations to produce a 2^c-fold collision is not a linear function of c for small values of c. The total number of operations should be minimized with respect to c, which implies that both terms should be made equal. Indeed, the total number of operations can be written as $2^{a-c} + 2^{b-c}$, which is minimal for $c = (b-a)/2$. The minimal value is equal to

$$2^{\frac{a+b+2}{2}}\,.$$

This yields the following expression for the security level:

$$S = \frac{h}{2}\left[\frac{k^2-1}{k^2} + \frac{1}{k(k-1)}\right] - n\,\frac{2}{k-1} + \frac{5}{2} + \frac{3}{2}\log_2(k-1)\,.$$

For k between 3 and 5, the relation between h and S is indicated in table 5.14. For $k \geq 6$, the resulting expression is larger than $h/2$ for all values of h, which means that a simple birthday attack is more efficient. The theoretical results agree very well with the result obtained from computer calculations. Only for $k = 3$ and $S > 64$, a significant deviation can be observed for rates larger than 8. It can be explained by the fact that the program takes into account operations that are assumed to be free in our simplified model of the attack. This can be verified by limiting the recursion depth of the calculation program. The program shows that $k = 4$ is the best choice for S between 51 and 72 bits.

The case $\phi = 2$ In this case the data input is split into $k(k-1)/2$ parts, and every part goes to $k-2$ blocks. Computer calculation has shown that the most efficient attack is attack D with $s = 2$ and $s' = 1$. The number of operations can be approximated by the following expressions:

1. Number of operations to produce a collision for blocks 3 to k:

$$h\left[\frac{k-2}{k} + \frac{2}{k(k-1)(k-2)}\right] - n\,\frac{2}{(k-1)(k-2)} + 1 + \log_2(k-2) - c\,.$$

k	Security level S (bits)
3	$\frac{19}{36}h - 12.0$
4	$\frac{49}{96}h - 5.8$
5	$\frac{101}{200}h - 2.5$

Table 5.14: Relation between h and S for $\phi = 1$ and $k = 3$, 4, and 5.

2. Number of operations to produce a 2^c-fold collision for block 2:

$$\frac{h}{k} + c\,.$$

3. Number of operations to produce a 2^c-fold collision for the first block of H_{i-1}:

$$h\left[\frac{1}{k} + \frac{k-2}{k^2} + \frac{2}{k(k-1)}\right] - n\,\frac{2}{k-1} + 2(1 + \log_2(k-3)) + c\,.$$

This number of operations should be minimized with respect to c. For smaller values of h, the third term is negligible, while for larger values of h, the second term is negligible. For k between 4 and 6, the relation between h and S is indicated in table 5.15. The theoretical results agree very well with the result obtained from computer calculations. The program shows that $k = 4$ is the best choice for S smaller than 69 bits.

k	Security level S (bits)			
4	$h \geq 132$	$\frac{27}{48}h - 28.0$	$h \leq 132$	$\frac{10}{24}h - 8.7$
5	$h \geq 122$	$\frac{79}{150}h - 16.5$	$h \leq 122$	$\frac{5}{12}h - 3.0$
6	$h \geq 110$	$\frac{37}{72}h - 10.5$	$h \leq 110$	$\frac{17}{40}h - 0.7$

Table 5.15: Relation between h and S for $\phi = 2$ and $k = 4$, 5, and 6.

The case $\phi = 3$ In this case the data input is split into $\binom{k}{3}$ parts, and every part goes to $k - 3$ blocks. Computer calculation has shown that the most efficient attack is attack D with $s = 3$ and $s' = 1$. The number of operations can be approximated by the following expressions:

1. Number of operations to produce a collision for blocks 4 to k:

$$h\left[\frac{k-3}{k} + \frac{6}{k(k-1)(k-2)(k-3)}\right] - n\,\frac{6}{(k-1)(k-2)(k-3)} + 1 + \log_2(k-3) - c\,.$$

2. Number of operations to produce a 2^c-fold collision for block 3:

$$\frac{h}{k} + c\,.$$

3. Number of operations to produce a $2^{c'}$-fold collision for block 2 (with $c' = h/k + c - a_3$):

$$h\left[\frac{2}{k} + \frac{6}{k(k-1)(k-2)}\right] - n\,\frac{6}{(k-1)(k-2)} + 2(1 + \log_2(k-3)) + c\,.$$

4. Number of operations to produce a $2^{c''}$-fold collision for the first block of H_{i-1} (with $c'' = h/k + c' - a_2$). It can be shown that this number is significantly smaller than $\frac{h}{k} + c$.

This number of operations should be minimized with respect to c. The second term is always smaller than the third term. This results in the following expression for the security level:

$$S = \frac{h}{2}\left[\frac{k-1}{k} + \frac{6}{k(k-1)(k-3)}\right] - n\,\frac{3}{(k-1)(k-3)} + \frac{3}{2} + \frac{1}{2}\log_2(k-3)\,.$$

For k equal to 5 and 6, the relation between h and S is indicated in table 5.16. The theoretical results agree very well with the result obtained from computer calculations. The program shows that $k = 5$ is the best choice for S smaller than 82 bits.

k	Security level S (bits)
5	$\frac{19}{40}h - 22.0$
6	$\frac{9}{20}h - 10.5$

Table 5.16: Relation between h and S for $\phi = 2$ and $k = 4$, 5, and 6.

The case $\phi = 4$ The only case that has been studied is $k = 6$. Here it is not possible to derive simple analytic expressions that are sufficiently accurate. This is because the result obtained by the different attacks lie very closely together; depending on the value of h, the best attack is attack D with $s = 4$ and $s' = 2$ or 4. Moreover, the optimal values of c are very small, which means that the system is non-linear, and the attacks are strongly dependent on the use of recursion. An upper bound on S can be easily obtained using method A:

$$S \leq \frac{11}{30}h - 10.8\,.$$

A least squares fitting the computer evaluation yields a correlation coefficient of 0.99958 with the following expression:

$$S = 0.3756h - 14.974\,.$$

This can be approximated very well by $S = 3/8\ h - 15$.

5.3.4.5 Extensions

The study of the previous scheme assumed that h and x are continuous variables. However, in practice they will have to be integers that satisfy certain constraints:

1. x has to be an integer multiple of $\binom{k}{\phi}$. Therefore define $x' = x/\binom{k}{\phi}$.
2. $nk - h$ has to be an integer multiple of $k - \phi$.
3. h has to be an integer multiple of k. Therefore define $h' = h/k$.

Note that in order to perform the mixing stage on the output of the k blocks, one needs in fact the requirement that h is an integer multiple of k^2. However, this mixing stage is not critical in the security analysis of the scheme. The following algorithm steps through all values of x and h that satisfy the constraints, for which $h > h_0$. First the starting values are generated:

$$x'_1 = \left\lceil \frac{n - \left\lceil \frac{h_0}{k} \right\rceil}{\binom{k-1}{\phi}} \right\rceil \tag{5.12}$$

$$x_1 = \binom{k}{\phi} x'_1 \tag{5.13}$$

$$h_1 = k\left(n - \binom{k-1}{\phi} x'_1\right)\,. \tag{5.14}$$

Here $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. The next values are calculated as follows:

$$x_{i+1} = x_i + \binom{k}{\phi} \tag{5.15}$$

$$h_{i+1} = h_i + k\binom{k-1}{\phi}\,. \tag{5.16}$$

In the overview of the results it will be graphically indicated which schemes satisfy the requirements. It is of course always possible to think of schemes for which the parts differ 1 or 2 bits in size just to match the constraints. This will affect the security level compared to the ideal situation, but the decrease will certainly be only marginal. These schemes are less elegant, but as long as the asymmetry in the bit manipulations has no negative influence on the performance, this is not so important.

If the schemes are extended for $\beta > 1$, the following elements have to be considered. First, the allocation of the bits of H_{i-1} to the different blocks will have to be made in a similar way as for the data bits. However, some additional work has to be done because both allocations should be as independent as possible, i.e., the bits of H_{i-1} and X_i will have to occur in as many combinations as possible. It is clear that the rate of the scheme will increase with increasing β. The study of attacks on this scheme is more complicated, especially for type D attacks.

Another issue is how to extend this method to construct a collision resistant function. In this case the attacks to be considered would be simpler, because the interaction with the previous iteration steps can be neglected. This is not completely true however, because an attacker could exploit the fact that the function is not complete (not all output bits depend on all input bits), by producing collisions for part of the output blocks. However, if the program is adapted to evaluate this type of schemes, it becomes clear that they will never be very efficient: the best scheme with a security level of 56 bits under the constraint $k \leq 16$ has a rate of 20 (which is in case of software a little worse than MDC-4). It is a scheme with $\alpha = 3$ or $\phi = 12$. The size of the hashcode would be 272 bits, and every iteration processes 48 bits. The scheme is however very impractical because X_i has to be split into 455 blocks of 2 or 3 bits. We believe that it is certainly possible to produce more efficient collision resistant functions with a different approach, where all inputs are mixed faster from the beginning of the calculations.

The basic principles used here can also be applied to other hash functions based on block ciphers where the key is modified in every iteration step. As an example it is indicated how MDC-2 could be extended in two ways to obtain a security level larger than 54 bits. The 2 parallel DES operations will be replaced by 3 parallel DES operations ($k = 3$).

- A trivial way would be $\alpha = 3$: every data bit is used 3 times as plaintext. The size of the hashcode would be 192 bits, and the effective security level is equal to 81 bits. The rate of this scheme is equal to 3 (comprising a key change).
- A second scheme can be obtained by selecting $\alpha = 2$: the data input of 96 bits is divided into 3 32-bit parts, that are allocated in the optimal way to the 3 blocks. The rate of this scheme is equal to 2 (the same as MDC-2), but the security level is slightly larger than 60 bits.

Of course it is possible to extend this for $k > 3$, which will result in faster schemes that require more memory.

The fact that the keys are fixed opens a different possibility: if the keys are kept secret, the scheme gives a construction for a MAC for which it is hard to produce a collision even for someone who knows the secret key. This is not possible with the widespread schemes for a MAC. Applications where this property might be useful are discussed in section 2.3. Precautions should be taken to avoid that someone chooses weak keys, identical keys, or keys that are the complement of each other. This can be done by selecting a single key and by specifying a way to derive all these keys from this "master key". An example is to encrypt with the master key a string that contains the

number of the block. A more secure scheme can be obtained if two keys are specified and if triple DES encryption is used in this procedure. Note again that a given output bit will not necessarily depend on all key bits: therefore at least two iterations are required.

A disadvantage of all these new schemes is that the decreased rate has to be paid for by increasing the memory. The additional 64 to 80 bits are no problem for the chaining variables (certainly not when the computations are performed in software), but the increased size of the hashcode might cause problems. This is not the case for digital signatures, as most signature schemes sign messages between 256 and 512 bits long. Exceptions to this rule are the scheme by Schnorr [296] and DSA, the draft standard proposed by NIST [111], where the size of the argument is 160 bits. If the hash function is used for fingerprinting computer files, an increased storage can pose a more serious problem. However, it can be solved by compressing the result to a number of bits equal to twice the security level S. This can be done with a (slow) hash function with $\phi = 0$, security level S, and size of the hashcode $2S$.

5.3.4.6 Overview of results

This paragraph will give an overview of some of the results that have been obtained. First the best schemes are given for a given value of ϕ and a given security level in table 5.17. Note that it is not possible to obtain a specific security level, because the discrete character of the variables has been taken into account.

The relation between the rate R and the security level S with parameter ϕ, with $0 \leq \phi \leq k-2$ will be shown in the figures 5.8 to 5.11. The solutions that take into account the discrete character are marked with a $\diamond$. For a fixed k one can conclude that the trade-off between speed and security level becomes better for larger values of ϕ. For a fixed ϕ the situation is more complicated: the optimal k will increase with increasing security level. For a security level between 50 and 70 bits, the best choice is $k = 4$ for $\phi \leq 2$, and $k = \phi + 2$ if $\phi > 2$. The irregular behavior of the curves in case $k = 4$ and $\phi = 1$ and 2 (figure 5.9) is caused by the combination of the continuous approximation of the discrete input with the discrete character of the function that gives the number of operations for a 2^c-fold collision for $c \approx 1$ (figure B.1). If in the attack a c occurs that is slightly smaller than 1, it is not possible to use an extended attack where $c = 1$. When h increases (and hence R), c will also increase and if c becomes larger than 1, a more sophisticated attack can be applied, which results in a small decrease of the security level. Of course, in fact the input consists of an integer number of bits and this situation should not occur. Note however that the overall accuracy of the estimates is certainly not better than 1 bit, which means that this error is relatively small.

		security level $\simeq$ 54 bits				security level $\simeq$ 64 bits			
ϕ	k	R	h	S	$h(k-\phi)/\phi$	R	h	S	$h(k-\phi)/\phi$
0	4	6.92	108	54	108	8.00	128	64	128
1	4	5.82	124	58	93	6.40	136	64	102
2	4	4.74	148	55	74	6.10	172	69	86
3	5	4.00	160	53	64	4.57	180	62	72
4	6					4.27	204	62	68

Table 5.17: Overview of best results for small values of ϕ and k. The quantity $h(k-\phi)/\phi$ indicates how many output bits depend on a single input bit.

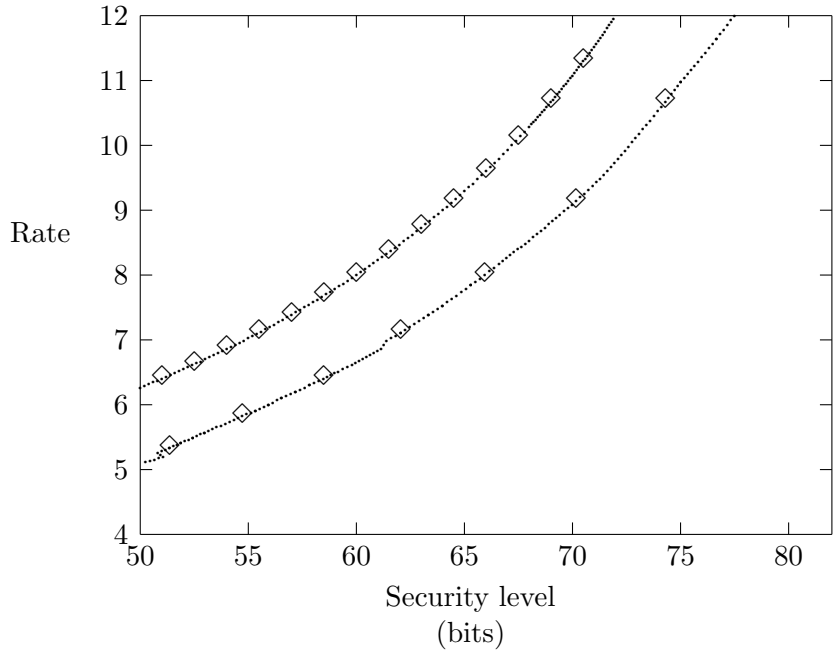

Figure 5.8: Relation between the rate R and the security level S for $k = 3$ with parameter $\phi = 0$ and 1.

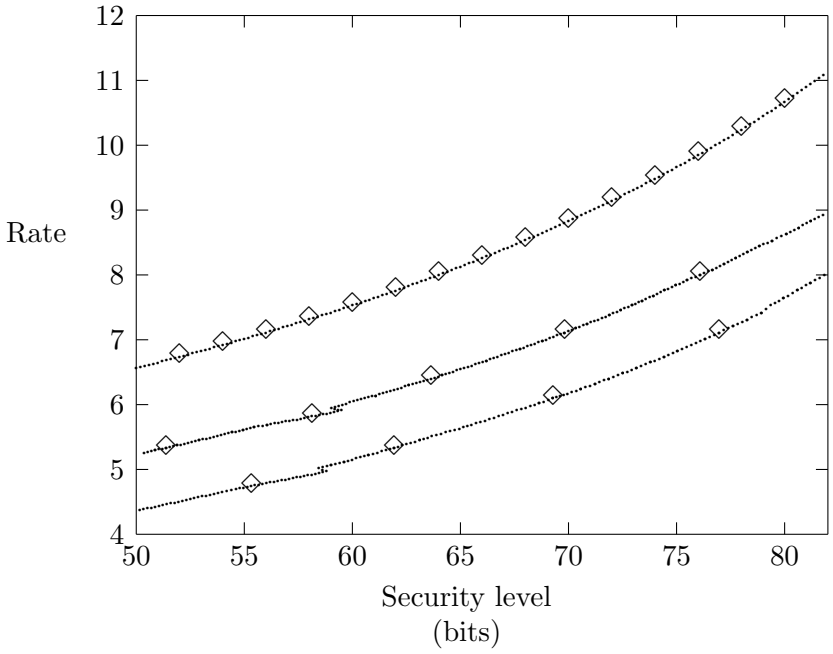

Figure 5.9: Relation between the rate R and the security level S for $k = 4$ with parameter $\phi = 0$, 1, and 2.

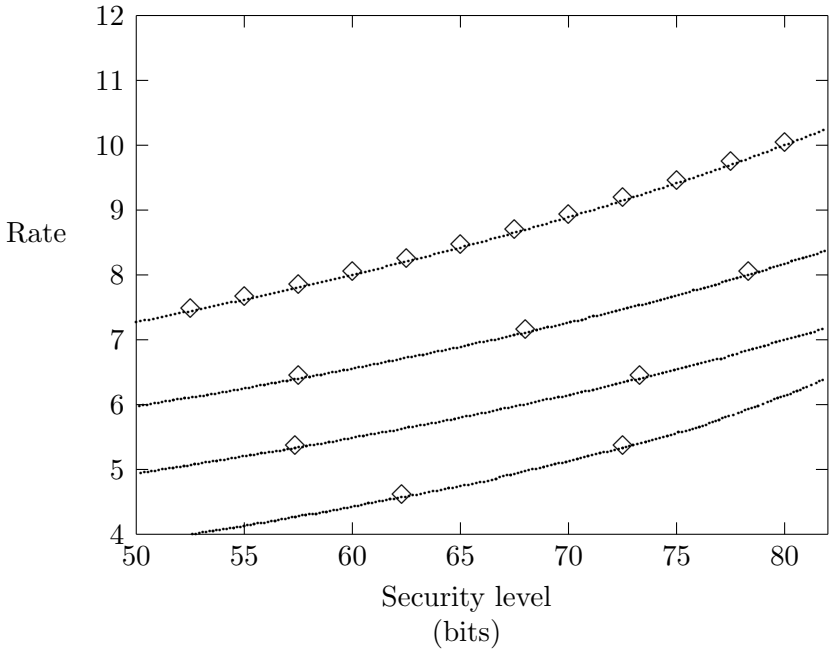

Figure 5.10: Relation between the rate R and the security level S for $k = 5$ with parameter $\phi = 0$, 1, 2, and 3.

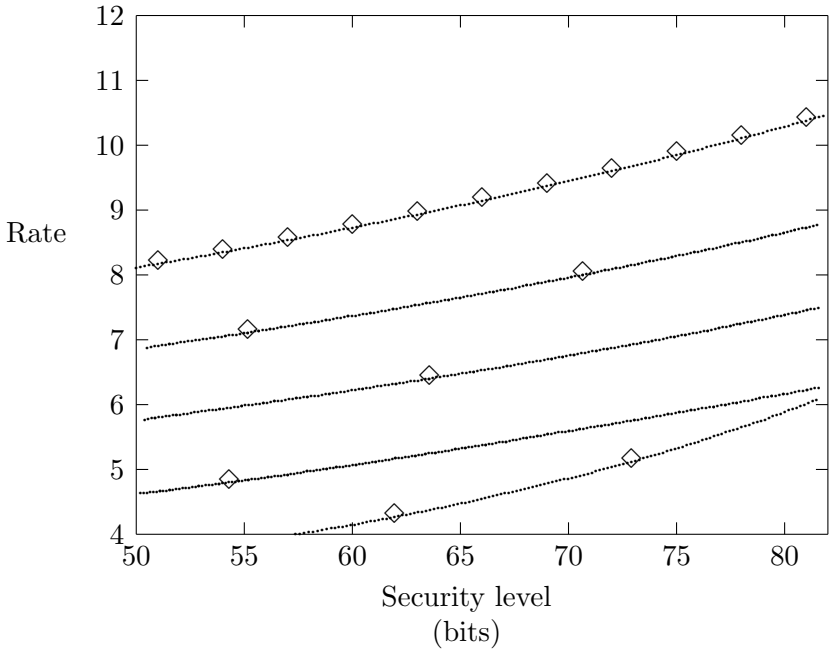

Figure 5.11: Relation between the rate R and the security level S for $k = 6$ with parameter $\phi = 0$, 1, 2, 3, and 4.

5.4 Overview of MAC proposals

In contrast with the variety of MDC proposals, very few algorithms exist. This can perhaps be explained by the fact that the first proposals for standardization that originated in the banking community are still widely accepted. The ANSI banking standard [9] specifies that the resulting MAC contains 32 bits, while the corresponding ISO standard [152] leaves open how many bits should be selected. A CBC-based MAC has also been standardized within ISO/IEC JTC1 (Information Technology) in [156].

It is clear that a result of 32 bits can be sufficient if a birthday attack (cf. section 2.5.1.3) is not feasible and if additional protection is present against random attacks (cf. section 2.5.1.1), which is certainly the case in the wholesale banking environment. In other applications, this can not be guaranteed. Therefore, certain authors recommend also for a MAC a result of 128 bits [169, 170]:

> *The use of the* 64*-bit MAC technique of FIPS and ANSI can not be considered sufficiently strong, and is not recommended if there is any possibility that the originator may attempt to defraud the message recipient, or if a Trojan horse could circumvent security controls through such a mechanism. In addition, the use of a MAC in certain command and control situations where the attacker may attempt to spoof computer-controlled equipment or processes is also not recommended.*

In a first section the CBC and CFB modes of a block cipher will be discussed, and in a second section two new schemes will be proposed.

5.4.1 CBC and CFB modes of a block cipher algorithm

A widespread method for computing a MAC are the CBC and CFB modes of a block cipher algorithm [9, 85, 109, 110, 152, 154, 156, 215]. The descriptions and standards differ because some of them select one of the two modes, suggest other padding schemes or leave open the number of output bits that is used for the MAC. If the length of the data is known on beforehand by both parties, an ambiguous padding rule can be acceptable (e.g., filling out with zeroes). The authentication of Electronic Funds Transfer in the US Federal Reserve System uses a 64-bit CBC, and additionally a secret value for IV that is changed daily and kept secret by the member banks [310]. In case of the CBC mode, the MAC consists of the last 64 bits of the ciphertext (or the n most significant bits of it) where in the CFB mode after processing the information the DES engine is operated one more time: if this step would be omitted, the MAC would depend in a linear way on the last data block. The MAC is selected from the high order end of the final output. The security relies on the well established forward error propagating properties of both modes of operation. However, in practice one faces several problems.

The attacks on the MDC schemes where the CBC mode or CFB mode are used (with a non-secret key) show that these functions are not one-way or collision resistant

for someone who knows the secret key. Therefore these schemes can not be used for certain applications (cf. section 2.3).

Three types of attacks are important for these schemes:

Exhaustive key search (cf. section 2.5.1.2): if DES is used as the underlying block cipher, exhaustive key search attacks are becoming certainly more feasible and less expensive [91, 117, 272]. It is clear that the use of triple DES with two keys can solve this problem. A different way to make exhaustive attacks infeasible is to perform an additional process [156]: after the CBC encryption, the result is deciphered with a key $K' \neq K$ and subsequently enciphered with the key K. In this way, compatibility with the simple scheme can be obtained by selecting $K' = K$. Another way to look at this suggestion is that only the last block is encrypted with triple DES using two keys.

An argument that has been suggested to keep less than 64 bits of the final result is that this would increase the "one-way" character of the MAC, i.e., the MAC would contain less information on the key. This is true because if a 64-bit result is known, one can in principle recover the key with an exhaustive attack. However, in some cases more than one key matches a single plaintext-ciphertext pair (cf. section 2.5.2.6). The probability that a single key corresponds to a given plaintext-ciphertext pair is 0.99610. The probability that 2 keys are possible is equal to 0.00389, and for 3 keys the probability is close to 0.00001. A second plaintext-MAC pair would however reveal the correct key with almost certainty. A general discussion of exhaustive attacks on a MAC has been given in section 2.5.1.2. From (2.2) it follows that if only 32 bits of the result are given, a single plaintext-MAC pair suggests approximately 2^{24} keys. It is however clear that in this case the key can be determined in most cases with two plaintext-MAC pairs.

Differential attacks (cf. section 2.5.2.7): for a DES-based MAC this new cryptanalytic technique requires currently 2^{47} chosen plaintexts to find the key. If the key is changed frequently, the attack will still find one of the keys. This is for the time being totally impractical, but it suggests that realistic attacks on single DES encryption will be developed before the end of the century. In this context it is important to note that the use of a secret IV in CBC mode does not help against differential attacks. Effective countermeasures are the addition of redundancy to every plaintext block or the use of triple encryption with two keys.

Chosen plaintext attack: this attack —suggested to us by B. den Boer— allows an existential forgery based on 3 plaintext-MAC pairs. Two of these plaintexts are known plaintexts, while the third one is a chosen plaintext. The attack can be described as follows: let H and H' be the CBC-MAC corresponding to key K and plaintext X and X' respectively. The attacker appends a block Y to X and obtains with a chosen plaintext attack the new MAC, that will be denoted with G. It is then clear that the MAC for the concatenation of X' and $Y' = Y \oplus H \oplus H'$ will also be equal to G (the input to the block cipher in the last iteration will in

both cases be equal to $H \oplus Y$). A similar attack holds for the CFB mode. This attack can be thwarted in several ways:

- The use of a post-fix free encoding, which means in practice to prepend the length to the message. This might be difficult for certain applications, where the size of the message is not known on beforehand. Note that the attack will not work in the case of fixed message size. Adding the length at the end is not completely effective: both Y and Y' could represent the length of the message.
- Apply a one-way function to the MAC after the CBC or CFB calculation. A possible function is $MAC' = E^{\oplus}(K, MAC)$.
- Apply an encryption with a different key K' to the MAC. An example of such a scheme is specified in [156] and was discussed before. If one desires to increase the key space to thwart exhaustive attacks, one can choose an independent key K' and perform a double encryption (i.e., decrypt with K' and encrypt with K). If this is not required, one can derive K' from K (e.g., by complementing every other nibble or half-byte), and perform a single encryption with K'.

If the message will also be encrypted, it is mandatory that different keys are used for authentication and for encryption. This has been illustrated for a limited number of cases in [167, 215]. A more general treatment will be given here, resolving some open problems posed in [167]. The conclusion is that this approach is very likely to introduce vulnerabilities.

Three different options can be distinguished:

1. The plaintext X is encrypted with the key K and initial value IV_1, resulting in the ciphertext Y. The MAC is computed on the ciphertext Y with the key K and initial value IV_2.

2. The MAC is computed on the plaintext X with the key K and initial value IV_2. Subsequently, the plaintext X is encrypted with the key K and initial value IV_1, resulting in the ciphertext Y.

3. The MAC is computed on the plaintext X with the key K and initial value IV_2. Subsequently, the concatenation of the plaintext X and the MAC is encrypted with the key K and initial value IV_1, resulting in the ciphertext Y.

Moreover, one has to specify the mode both for the encryption and for the MAC scheme, and whether it is allowed or not to use the same initial values. Even if only two modes are considered (namely CBC and CFB), this results in a large number of possible schemes. In case of the CFB mode, one has to specify the size of the feedback parameter. To simplify the calculations, the largest possible value for this parameter will be selected, resulting in the most efficient scheme.

The simplest issue to discuss is the influence of the initial value. If a specific relation exists between IV_1 and IV_2, the schemes of option 2 and 3 will mostly become very

weak. As will become clear from the following paragraph, the choice of the initial values is not so important in case of option 1. An overview is given in table 5.18 of the influence of a specific relation between the initial values on the MAC.

encryption	MAC	relation between IV's	MAC for option 2	MAC for option 3
CBC	CBC	$IV_1 = IV_2$	Y_t	$E(K, 0)$
CFB	CFB	$IV_1 = IV_2$	Y_t	0
CBC	CFB	$IV_1 = E(K, IV_2)$	Y_t	$E(K, 0)$
CFB	CBC	$E(K, IV_1) = IV_2$	$E(K, Y_t)$	0

Table 5.18: Relation between initial values for encryption (IV_1) and MAC calculation (IV_2) with the same key K that yield a possibly weaker scheme. The last block of the ciphertext is denoted with Y_t.

The following conclusions can be drawn from table 5.18:

- The specific relation to be imposed on the initial values is dependent on the secret key K if the mode for encryption differs from the mode of the MAC calculation.
- If option 3 is selected, the MAC is independent of both plaintext and ciphertext, and will depend on the key if the encryption is done in CBC mode. This means that under these conditions option 3 is completely insecure. For the case of the CBC mode in encryption and MAC calculation, R. Jueneman ([167], p. 38) also reaches the conclusion that option 3 is insecure, but he uses a wrong argument to support this.
- If option 2 is selected, the MAC depends on the ciphertext (and thus on the plaintext). Moreover it also depends directly on the key (the MAC can not be computed from plaintext and ciphertext only) if the encryption is done in CFB mode and the MAC calculation is done in CBC mode. It is clear that a simple chosen plaintext attack is sufficient to obtain the MAC corresponding to a given ciphertext. In the next paragraph this type of attack will be generalized to all cases.

Even if no special relation between the two initial values exists, it is possible to break the schemes. In case of option 3 and the CBC mode for both encryption and MAC, a known plaintext attack has been demonstrated by D. Coppersmith [167]. The idea is that if the attacker knows a plaintext-ciphertext pair (P, C), he can easily append the following blocks to the ciphertext:

$$MAC \oplus P, C \oplus P, C \oplus P, \ldots, C \oplus P, C\,.$$

The MAC will still be correct and the corresponding plaintext will be random ("authenticated garbage"). This attack can be extended to all other cases if more plaintext-ciphertext pairs are known.

As a conclusion one can recommend that always two different keys should be used. This can be compared to the public-key systems, where protection of both secrecy

and authenticity requires two different operations. However, a very pragmatic solution could be to derive the key for authentication from the key for secrecy in such a way that both encryption operations will become totally unrelated: in case of DES one should not use the complement, but complementing every other nibble (half-byte) would yield a good solution. If a hierarchical key structure is present, it is always possible to derive two unrelated keys from a single master key. A similar procedure could be applied to the initial values.

5.4.2 Invertible chaining for a MAC

In this section an alternative scheme for a MAC is proposed, that has the advantage that even for someone who knows the secret key, it requires $O(2^{n/2})$ operations to find a preimage of a given value. Based on the synthetic approach of section 5.3.1.4 it can be shown that this is best possible for a scheme with rate 1. From the 16 schemes with a constant key that are listed in table 5.2, 7 are trivially weak, 6 succumb to a direct attack (among those the CFB and the CBC mode), and there is 1 scheme vulnerable to a permutation attack, to a forward attack, and to a backward attack respectively.

The scheme vulnerable to a backward attack can be described as follows:

$$f = E(K, X_i \oplus H_{i-1}) \oplus X_i \,.$$

It is recalled that vulnerability to a backward attack implies that it is trivial to produce a pseudo-preimage, and a preimage (or second preimage) of a given hashcode can be computed with a meet in the middle attack (cf. section 2.5.2.1) in $O(2^{n/2})$ operations. Note that this is comparable to BMAC (with a secret key), but with the difference that this scheme is twice as efficient. If the size of the block cipher is sufficiently large, this scheme can yield a MAC that is collision resistant for someone who knows the key.

On the other hand, the scheme vulnerable to a forward attack is also of interest:

$$f = E(K, X_i \oplus H_{i-1}) \oplus X_i \oplus H_{i-1} = E^{\oplus}(K, X_i \oplus H_{i-1}) \,.$$

Here it is trivial to find a second preimage, but hard to find a preimage or pseudo-preimage of a given value in the range. It is only useful for (exotic) applications where the attacker knows the secret key and the MAC, but does not know the message corresponding to the MAC. A disadvantage of this scheme is that for a fixed X_i the function f is injective. This implies that the "loss of memory" problem that was described in section 2.4.1 will occur. An overview of the number of operations for several attacks is given in table 5.19. Note that the three attacks described for the CBC-MAC are still valid. The only difference is that in case of a chosen plaintext attack the MAC corresponding to the new message will be equal to $G \oplus H \oplus H'$. The proposed counter-measures can thwart these attacks.

An alternative solution to the construction of a one-way and/or collision resistant MAC is to encrypt the hashcode of an MDC.

mode	Backward	Forward	Direct (CBC/CFB)
pseudo-preimage	1	2^{64}	1
preimage	2^{32}	2^{64}	1
second preimage	2^{32}	1	1

Table 5.19: Number of operations to produce a pseudo-preimage, preimage and second preimage for several MAC schemes.

5.5 Conclusion

This chapter has attempted to develop a framework for the study of hash functions based on block ciphers. First it has been shown how the idea of using a hash function in cryptography emerged from the redundancy that was added to the plaintext before encryption. The treatment of the addition schemes has been generalized. Subsequently an attempt has been made to classify the large number of proposals. For the MDC schemes where the size of the hashcode is equal to the block length, a synthetic approach was suggested. For the other MDC schemes, a thorough evaluation was presented, which has resulted in the cryptanalysis of seven schemes, and a new scheme was proposed. For the MAC schemes, the interaction between encryption and MAC calculation with the same key has been studied in more detail, and two new schemes were discussed that have some interesting properties. The main open problem is an optimal construction for a CRHF based on a block cipher with "small" block length and key.

Chapter 6

Hash Functions Based on Modular Arithmetic

But who shall watch over the guard?

6.1 Introduction

The motivation to use schemes based on modular arithmetic is comparable to the motivation for schemes based on block ciphers. A variety of cryptographic schemes have been based on number theoretic problems: public-key cryptosystems (like RSA [278] and ElGamal [100]), digital signature schemes, and pseudo-random bit generators. The idea is to reduce the security —provably or intuitively— to the hardness of number theoretic problems. The availability of an implementation of modular arithmetic is a second argument to design a hash function based on these operations, especially in an environment where resources like chip area or program size are limited. A third argument is that these hash functions are easily scalable, i.e., the size of the result can be chosen freely. The disadvantage is that precautions have to be taken against attacks that exploit the mathematical structure like fixed points of modular exponentiation (trivial examples are 0 and 1), multiplicative attacks, and attacks with small numbers, for which no modular reduction occurs.

In a first section, an overview will be given of MDC proposals. Two types will be identified: schemes based on 32-bit integer operations, and schemes with large integer arithmetic, where the operations are defined on integers of size 512 bits or more. In the latter case the operands are elements of the ring corresponding to an RSA modulus, i.e., the product of two large primes, or of a Galois Field $GF(p)$ or $GF(2^n)$. For convenience we will refer to these schemes as schemes with large modulus, although this terminology applies strictly speaking only to RSA-based schemes. In a second section it will be shown how the MAC scheme by F. Cohen and Y.J. Huang based on modular arithmetic can be broken with an adaptive chosen message attack. Finally

the conclusions will be presented.

The main contributions of this chapter are the extension of the synthetic approach to the schemes based on modular exponentiation and the new attacks on the scheme by Y. Zheng, T. Matsumoto, and H. Imai and on the scheme by F. Cohen and Y.J. Huang.

6.2 Overview of MDC proposals

This section will discuss the large number of MDC schemes based on modular arithmetic. For schemes with small modulus, it will be shown that several attempts to design such a scheme were not successful. For the schemes with large modulus, one can essentially distinguish between two types of schemes: fast schemes that make use of modular squaring or exponentiation, that have however several security problems, and provably secure schemes that are rather inefficient.

6.2.1 Schemes with a small modulus

In order to improve the addition schemes (cf. section 5.2.3), R. Jueneman, S. Matyas, and C. Meyer attempted to construct a fast MDC using only single precision arithmetic (32-bit multiplications). The idea was to implement these operations on the coprocessor of a personal computer in order to obtain an MDC that is faster than a DES-based algorithm. A first scheme has a block length of only 32 bits, and was called Quadratic Congruential MDC (QCMDC) [166, 167]:

$$f = (X_i + H_{i-1})^2 \bmod N\,,$$

where N is the Mersenne prime $2^{31} - 1$, and $IV = 0$. The goal of this scheme was to protect the authenticity of information that is generated by the "user" and that will be encrypted in a later stage by a separate entity, the "sender". It was intended for environments where the secrecy protection is independent from the authenticity protection, and where the use of a separate MAC is too expensive in terms of computation and key management. The designers realized that the protection offered by this MDC is too weak in certain cases. Therefore it was recommended in [166] to let X_1 be a secret random seed (not known to the user) that is different for every message and that is sent together with the message. As it should not be known in advance by sender and receiver, it is not considered to be a secret key. However, in [167] it is stated that this protection is not sufficient if the user attacks his own messages:

- In a first attack, the user simply discards X_1, and the corresponding ciphertext, and constructs a message with the same MDC under the condition that the random seed is equal to X_2 (this block is known by the user). This can be done with a meet in the middle attack (cf. section 2.5.2.1) that requires only 18 plaintext-ciphertext pairs, if the variations are constructed by permuting blocks (indeed $9! > 2^{16}$). This attack can be thwarted by increasing the size of the MDC to 128 bits (although 80 bits are suggested in [167]).

- The second attack applies in case of encryption with an additive stream cipher or with a block cipher in OFB mode. The user can then determine the key stream sequence and perform subsequently a substitution, as discussed in section 2.3.1.2. To thwart this attack, the MDC should depend on a secret key that is not known to the user. A possibility is to choose IV to be equal to $E(K, 0)$, where K is the secret key used for the secrecy protection: in this case one clearly has a MAC where the key is derived from the encryption key. However, in order to increase the protection, it is recommended that the secret seed is also used. Moreover, the overall security level is still restricted to 32 bits.

These considerations lead to a second version, where the size of the result was extended to 128 bits by taking four passes over the message [168], while the result of an iteration is used as IV for the next one. Shortly after publication of this scheme, D. Coppersmith showed that with the generalized meet in the middle attack (cf. section 2.5.2.3) finding a collision in the second version requires about 2^{45} operations, instead of 2^{32} for the first version. This forced the author to design the third version.

In this version each message block X_i and chaining variable H_i of 128 bits are split into four 32-bit blocks that will be denoted with X_{ij} and H_{ij} respectively ($0 \leq j \leq 3$). To simplify notations, the index i will be omitted from hereon. One then defines four functions f_j as follows:

$$f_j = \Big[(H_{j \bmod 4} \oplus X_0) - (H_{(j+1) \bmod 4} \oplus X_1) + (H_{(j+2) \bmod 4} \oplus X_2) - (H_{(j+3) \bmod 4} \oplus X_3)\Big]^2 \bmod N\,.$$

When a message block is processed, the value of H_j is updated with f_j, and the new value is used immediately to update the other chaining variables. However, D. Coppersmith pointed out that the choice of the modulus results in a particular weakness: exoring the string $\mathtt{80\,00\,00\,01_x}$ has the effect of inverting the sign and the low order bit (in two's complement), which can be equivalent to adding or subtracting the modulus. This resulted in the fourth and final version, that was called QCMDCV4.

The QCMDCV4 algorithm uses four different moduli N_j, namely the four largest prime numbers smaller than $2^{31} - 1$: $2^{31} - 19$, $2^{31} - 61$, $2^{31} - 69$, and $2^{31} - 85$. To avoid attacks where the modulus is added or subtracted, a 5th block is constructed

$$X_4 = (00 \parallel X_0[31-26] \parallel X_1[31-24] \parallel X_2[31-24] \parallel X_3[31-24])\,,$$

and bit $X_j[30]$ ($0 \leq j \leq 3$) is forced to 0. Here $X[i-j]$ denotes the consecutive bits $i, i+1, \ldots, j$ of the integer X, and the most significant bit has number 31. The four functions f_j are defined as follows:

$$f_j = \Big[(H_{j \bmod 4} \oplus X_0) - (H_{(j+1) \bmod 4} \oplus X_1) + (H_{(j+2) \bmod 4} \oplus X_2) - (H_{(j+3) \bmod 4} \oplus X_3) + (-1)^j X_4\Big]^2 \bmod N_j\,.$$

In [57] D. Coppersmith showed that this scheme can be broken in about 2^{18} operations by combining a correcting block attack (cf. section 2.5.2.4) with a birthday attack (cf.

section 2.5.1.3). First the attacker selects an arbitrary message and calculates the corresponding values for the chaining variables. Now he needs a single block to obtain a chosen value of H_{t0}, H_{t1}, and H_{t2} (on almost all positions), and subsequently he obtains a collision for the remaining 35 bits with a birthday attack. This means that finding a preimage will require about 2^{35} operations. Finding a match on almost all bits of H_{t0}, H_{t1}, and H_{t2} is easy: one takes the square root of the chosen values, and subsequently solves for X_0 through X_3, starting with the least significant bits, that are independent of the higher order bits: after taking square roots the function is not complete (the output bits do not depend on every input bit). The only complication is caused by the additional block X_4. The efficiency of this attack suggests that the approach is clearly not adequate, and that modifications of this scheme will probably succumb to a similar attack.

To conclude this section, it is mentioned that recently a hash function and a MAC were proposed by C. Besnard and J. Martin [19]. Both schemes are mainly based on squaring modulo $2^{31}-1$. It is suspected that they will succumb to a divide and conquer approach.

6.2.2 Schemes with a large modulus

Most schemes in this section are based on a RSA modulus that is the product of two large primes p and q, and that satisfy some additional requirements [132]. The modulus will be denoted with N, and its size in bits is written as n, or $n = \lceil \log_2 N \rceil$. Other schemes are based on operations in $GF(2^n)$ or in $GF(p)$, where p is an n-bit prime. In most proposals n was suggested to be 512 bits, but it is clear that in the near future this will have to be increased to 768 or 1024 bits. As the size of the result is also n bits, most schemes were designed to yield a CRHF.

Before describing these schemes, it is remarked that there exists some controversy about the choice of the RSA modulus N: if N is generated by the signer, N could be equal to the modulus he uses to generate RSA signatures. In that case the signer has some additional information over the verifier, as he knows its factorization into primes. This might give him an advantage to attack the hash function, even if the security of the hash function does not rely completely on factorization. It is certainly a problem that if one is not able to prove the security of the hash function, one will also not be able to prove that knowledge of the factorization does not help an attacker. Moreover, the security of the provably secure schemes discussed in this section does rely on the fact that factoring the modulus is a hard problem. Even if a special modulus is generated for the hash function, the person who has generated the modulus may have an advantage.

As a solution it was suggested to use a universal modulus for hashing that is to be generated by a trusted authority. The main problem with this solution is the trust in this authority, and the fact that factorization efforts and other attacks will be concentrated on this single modulus. This in turn will require a larger size for this modulus. Note that a similar argument holds for the key of a certification authority, that has to certify the public keys of all users [46].

A way out of this impasse could be to calculate a hash value with the modulus of the signer and with the modulus of the intended verifier(s), and to sign the concatenation of the hashcodes. This will solve the trust problem, but it decreases the efficiency, especially if there is more than one verifier.

Note that a similar problem holds in case of schemes based on a single prime number p: here it is conceivable (but not essential) that a "weaker" prime is chosen. However, if the size of p is sufficiently large, the probability that a random prime is weak is very small, and no way is known to generate a prime for which the discrete logarithm problem is easy if one knows the trapdoor information, but finding the trapdoor itself is hard.

For this type of schemes it makes sense to define the **rate**:

Definition 6.1 *The* **rate** *of a hash function based on a modular arithmetic is the number of squarings or multiplications in the ring $\mathbb{Z}_N$ needed to process a block of n bits.*

Note that in software a squaring in $\mathbb{Z}_N$ can be implemented almost twice as fast as a multiplication; as most schemes use only squarings, this distinction will not be made.

As indicated in the introduction, a distinction will be made between schemes that are provably secure and schemes that are more practical.

6.2.2.1 Schemes that are not provably secure

A synthetic approach, similar to the treatment in section 5.3.1.4 for block ciphers, can evaluate the possible combinations with a single modular squaring (rate equal to 1) combined with 1 or 2 exors or additions modulo N. Here one has only two inputs, the argument of the modular squaring P and the feedforward FF, that each can be equal to X_i, H_{i-1}, $X_i \oplus H_{i-1}$, or a constant V. This yields 16 possible schemes, of which 7 combinations are trivially weak. The attacks that apply are indicated in table 6.1. The same symbols have been used as in table 5.2 of section 5.3.1.4, namely direct attack (D), permutation attack (P), forward attack (F), backward attack (B), and fixed point attack (FP). If the factorization of the modulus is known, the schemes are equivalent to the cases studied in section 5.3.1.4, where the secret key K is known and constant (note that in case of squaring the inverse mapping is not uniquely defined, but this has no influence on the treatment). This means that the same attacks apply: they are indicated between square brackets in table 6.1. It can be seen from table 6.1 that there are four cases for which attacking the scheme becomes harder if the factorization of the modulus is not known. Only for two schemes this means a significant improvement.

A first remark is that scheme 9 is in fact trivially weak, as the hash function is the sum modulo 2 of the message blocks and a constant that can be zero. Four of the other schemes have appeared in the literature for the exponent equal to 2, namely schemes 1, 3, 5, and 8 (the CBC-mode). The weakness of scheme 1 was discussed in [122]: one can permute blocks, insert an even number of blocks (these attacks also were discussed in section 5.3.1.4), insert zero blocks or manipulate with small blocks (i.e., blocks for which $X_i^2 < N$). The first three attacks apply to scheme 2 as well. The weakness of

$FF\backslash P$	X_i	H_{i-1}	$X_i \oplus H_{i-1}$	V
X_i	–	D^3 [D]	FP^5 [B]	–
H_{i-1}	P^1 [D]	–	FP^6 [D]	–
$X_i \oplus H_{i-1}$	P^2 [P]	D^4 [D]	F^7 [F]	D^9 [D]
V	–	–	F^8 [D]	–

Table 6.1: Attacks on the 16 different schemes based on modular exponentiation. The schemes are numbered according to the superscript.

scheme 3 (the dual of scheme 1) was discussed in [230]. Scheme 5 is similar to the $E^{\oplus}$ operation for block ciphers, and was suggested in [244]:

$$f = (X_i \oplus H_{i-1})^2 \bmod N \ \oplus X_i \,.$$

The fixed point attack was presented in [122]: if FP is a fixed point of the modular squaring [26], $H_{i-1} \oplus FP$ can be inserted between X_i and X_{i+1} without changing the hashcode. A similar attack applies to scheme 6, but here the output in case of insertion of $H_{i-1} \oplus FP$ will be equal to $H_{i-1} \oplus FP$. If the fixed point 0 is chosen however, the output will again be H_{i-1}. Note that fixed point attacks can be thwarted easily by adding the message length in the padding procedure (cf. section 2.4.1).

If one takes into account the attacks that assume knowledge of the factorization of the modulus, one can conclude from table 6.1 that scheme 5 is the most promising scheme. The fact that one can go backwards in constant time means that finding a preimage will require only $2^{n/2}$ operations with a meet in the middle attack (cf. section 2.5.2.1). This is no problem if the modulus size is 512 bits or more.

From the previous discussion it seems a little odd that the CBC mode (scheme 8) is the proposal that has received the most attention. It goes back to a 1980 scheme by D. Davies and W. Price, and a similar scheme suggested by R. Jueneman in 1982 (cf. section 6.2.1). The idea of the designers is to improve the security by adding redundancy (to X_i or to H_{i-1}), or by using exponents larger than 2. The redundancy will thwart the forward attack (or correcting block attack, section 2.5.2.4), and —in some cases— the direct attack if the factorization of the modulus is known. However, in all cases finding a pseudo-preimage requires only constant time, and hence a preimage can be found in $2^{n/2}$ operations.

A first proposal by D. Davies and W. Price [70] is very slow, namely to choose an n-bit exponent (the rate is $1.5n$):

$$f = (H_{i-1} \oplus X_i)^e \bmod N \,,$$

where e is the public exponent for an RSA cryptosystem. The RSA signature is equal to H_t concatenated with a signature on IV. The designers suggest to add redundancy ($X_{t+1} = X_1$) to thwart a forward attack or correcting block attack. However, it is noted that additional redundancy in every block is required (or including of the

message length) to avoid attacks based on the fixed points P of RSA encryption like 0 and 1. An example of such a collision is to choose the message starting with $X_1 = IV$ followed by an arbitrary number of 0 blocks. Also the proposed redundancy does not solve the problem of a direct attack in constant time if the factorization of n is known: one can easily compute the secret exponent d, and after random choice of X_1 one obtains H_1 and H_t; it is now easy to find suitable values for the remaining X_i.

In order to speed up the scheme, a second proposal was to choose $e = 2$ [73]. To avoid vulnerability to appending or inserting blocks based on a correcting last block attack, the 64 most significant bits of every message block were fixed to 0. This scheme has a rate of 1.14. With this approach, someone who knows the factorization of the modulus can find a preimage in 2^{64} operations. On the other hand, if one does not know this factorization, it should be impossible for an attacker to construct a correcting block with the imposed redundancy. However, it was pointed out by A. Jung that correcting blocks can be found based on continued fractions using the "extended Euclidean algorithm" [171]. A detailed analysis by M. Girault in [122] shows that about 2^{150} collisions can be found and that at least 170 redundancy bits have to be introduced to thwart the attack. A similar examination of fixing the least significant positions shows that in this case at least 128 fixed positions are required. A broader overview of techniques to find or approximate Lth roots can be found in [124].

The next proposal was to disperse 256 redundancy bits over the block by fixing the four most significant bits in every byte to 1 (or the hexadecimal value $\mathtt{F_x}$). This scheme has rate 2, and finding a preimage requires $2^{n/2}$ operations if the factorization of the modulus is known. In fact two proposals were made, that differ slightly: the proposal by A. Jung [171] replaced the exor by addition modulo N, and added a normalization at the end: if $H_t < \lfloor N/2 \rfloor$, $\lfloor N/2 \rfloor$ was added to H_t. These modifications were not incorporated in the informative annex D of CCITT-X.509 [46]. Applications where this hash function is recommended are the French standard ETEBAC 5 [47] for file exchange between banks and customers, and TeleTrusT [99].

A correcting block attack of D. Coppersmith on the CCITT scheme [125, 172] finds two messages X and X' such that

$$h(X') = 256 \cdot h(X)\,. \tag{6.1}$$

It will be shown that this relation can be used for an existential forgery of a signature based on three chosen messages. This forgery works if (6.1) holds between integers or if the modulus of the hash function is equal to the modulus of the multiplicative signature scheme like RSA. A simple way to thwart this attack is to add redundancy to the hashcode as specified in [155]: this redundancy guarantees that no two arguments of the signature scheme are multiples of each other.

The correcting block attack can be described as follows: given two arbitrary messages X and X' that yield hash value H_{t+1} and $H'_{t'+1}$ respectively, two blocks Y and Y' will be constructed (with the correct redundancy) such that

$$H'_{t'+1} \oplus Y' = 16 \cdot (H_{t+1} \oplus Y)\,.$$

From this equation all bits of Y and Y' can easily be determined based on the observation that multiplying by 16 corresponds to a left shift over 4 positions. If the half-bytes or nibbles of a bit string are denoted with $X[i]$, where $X[0]$ corresponds to the least significant nibble, one obtains the following equations (to simplify notations n is assumed to be a multiple of 4, and the indices are omitted from H_{t+1} and $H'_{t'+1}$):

$$\begin{aligned} Y'[0] &= H'[0], \\ Y'[2i] &= H[2i-1] \oplus H'[2i] \oplus \mathtt{F_X} \quad (1 \leq i < n/4), \\ Y[2i] &= H[2i] \oplus H'[2i+1] \oplus \mathtt{F_X} \quad (0 \leq i < n/4)\,. \end{aligned}$$

It is clear that $Y[i] = Y'[i] = \mathtt{F_X}$ for odd i. An additional condition that has to be imposed is that $H[n/4-1] \oplus \mathtt{F_X}$ is equal to 0, which happens with probability 1/16. If this is not the case, a different X has to be chosen. It is clear that now $h(X'\|Y') = 256 \cdot h(X\|Y) \bmod N$. If this equation has to be valid between integers, one has to impose that $h(X\|Y)$ is smaller than $N/256$. This event has a probability of about 1/256, which reduces the total probability to 1/4096.

The attack on the signature scheme requires two message pairs (X_0, X'_0), (X_1, X'_1) for which

$$h(X'_j) = 256 \cdot h(X_j), \qquad \text{for } j = 0, 1\,. \tag{6.2}$$

If the hash function is now combined with a multiplicative signature scheme like RSA, the following relation holds:

$$\mathrm{Sig}(h(X'_j)) = \mathrm{Sig}(256 \cdot h(X_j)) = \mathrm{Sig}(256) \cdot \mathrm{Sig}(h(X_j)), \qquad \text{for } j = 0, 1. \tag{6.3}$$

The last equality is only valid if equation (6.2) holds between integers, or if the modulus of the signature schemes is equal to the modulus of the hash function. In both cases, equation (6.3) yields 2 multiplicative relations between the 4 signatures and the unknown constant Sig(256) (one can assume that it is not possible to obtain this value directly), which implies that if 3 signatures are known, the fourth can be computed.

In the variant by A. Jung [171], the normalization makes it impossible for equation (6.2) to hold between integers, which implies that the attack will only work if the same modulus is used for signature computation and for hashing. One also has to replace the exor by addition. It is also clear that this attack is independent of the nature of the redundancy bits: it does not help to make the 4 redundancy bits dependent on the 4 other bits in the byte, as suggested in [122].

A. Jung suggests the following solution to thwart the Coppersmith attack [172]:

- The block size n' is defined as the largest multiple of 16 strictly below n, or

$$n' = 16 \cdot \left\lfloor \frac{n-1}{16} \right\rfloor .$$

  Hence between 1 and 15 most significant bits of the argument of the squaring operation will always be 0.

- Pad the message with binary ones to complete a block, and subsequently add an extra block containing the binary representation of the length of the message in bits.
- Before every squaring operation the 4 most significant bits of the n'-bit block $H_{i-1} \oplus X_i$ are forced to 1; this should make it impossible to find integer multiples, just like the previous normalization.
- After every squaring operation the n-bit result is reduced to n' bits by forcing between 1 and 15 most significant bits to 0.

He expects that these modifications thwart the Coppersmith attack and will possibly lead to a proof of security based on the collision resistance of the compression function. The result of I. Damgård (cf. section 4.3.3.2) can indeed be extended to this scheme [173] if the output of the compression function is (by definition) restricted to the 256 bits to which a constant is added in the next iteration (the fact that more bits of H_i are used in the next iteration can not decrease the security level). Due to the small difference with the broken scheme, it seems that a careful analysis is required. Moreover, this scheme has still the small problem that knowledge of the factorization of N leads to a preimage in $2^{n/2}$ operations.

An alternative improvement with rate 1.33 was put forward in [160] for $n = 512$ bits:

- 128 redundancy bits are added to X_i by fixing the 8 most significant bits of every 32-bit word to $\texttt{3F}_\texttt{x}$.
- The most significant byte of H_{i-1} is added modulo 2 to the next byte of H_{i-1}, and subsequently the most significant byte is forced to 0.
- The exor is replaced by addition modulo N.

Additionally it is suggested to limit the hashcode by taking only the most significant bytes of H_t (16 bytes or more). Because of the limited redundancy, a preimage requires about 2^{128} operations. Note that the appearance of this scheme in an ISO/IEC document does not imply that it is approved by ISO/IEC. This scheme should be carefully studied before it can be recommended.

I. Damgård suggested to construct a collision resistant function based on modular squaring (with $n = 512$) as follows [66]: introduce redundancy in both H_{i-1} and X_i at complementary positions. This has the result that the exor corresponds to a concatenation of bytes. His proposal has rate 1.36. The first three bytes of every 4-byte group (starting from the least significant byte) come from X_i, and the fourth byte is a byte from H_{i-1}. In order to avoid trivial collisions, the most significant byte is made equal to $\texttt{3F}_\texttt{x}$. This choice is a compromise between security (spread the redundancy bits as much as possible) and efficiency (lump the redundant bits together in bytes).

However, B. den Boer [82] showed that it is easy to find a collision for the basic function (i.e., a pseudo-collision). One looks for two numbers X and $X + y$ with most significant byte equal to $\texttt{3F}_\texttt{x}$ such that their squares differ only in the 3 least significant

bytes. From the equation:

$$(X+y)^2 \bmod N = X^2 + 2Xy + y^2 \bmod N\,,$$

it follows that it is sufficient to choose a y between 2^7 and 2^{11}, and to look for an X such that $2Xy$ is equal to $kN+z$, with $z \approx -y^2$ (additionally there should also be no carry from the addition of $z+y^2$ and X^2 to the fourth least significant byte of $X+y$). A suitable X can be found as follows: write kN for successive integer values of $k \geq 1$ as $X2y+z'$, with $z' \approx y^2$ (more precisely $y^2 - y < z' < y^2 + y$), until one obtains a value of X with most significant byte equal to $\texttt{3F}_\texttt{x}$. Then $2Xy = kN - z'$, or $z = -z'$. The following argument suggests that one will find a suitable value for X: if $N = q2y + r$, $0 < r < 2y$, then the candidates for X will start with q and increase at every step with q or $q+1$. Hence it follows that a sufficient number of candidates for X will be found, which suggests that a suitable X will be found with overwhelming probability.

An improved version with rate 1.36 was suggested in [160] for $n = 512$ bits:

- First a mapping g is defined that maps a 64-byte integer H_{i-1} onto the 16-byte integer H'_{i-1}. The ith byte of an integer A (starting with the most significant byte) is denoted with $A[i]$. Let $v(i)$ be the following permutation of the integers modulo 16: $v(i) = 5i + 11 \bmod 16$. Then the mapping $g(H_{i-1})$ is defined as follows:

$$\begin{aligned} H'_{i-1}[j] = {} & H_{i-1}[v(j)] \;\oplus\; H_{i-1}[v(j)+16] \;\oplus\; H_{i-1}[v(j)+32] \\ & \oplus\, H_{i-1}[v(j)+48], \qquad\qquad 0 \leq j \leq 15\,. \end{aligned}$$

- The most significant byte of X_i is forced to the value $\texttt{3F}_\texttt{x}$.
- The value to be squared is defined as follows: the first three bytes of every 4-byte group (starting from the most significant byte !) come from X_i, and the fourth byte is a byte from H'_{i-1}.
- The hashcode is equal to $g(H_t)$.

A disclaimer about the ISO/IEC approval holds for this scheme as well; further evaluation is certainly recommended.

Another proposal to construct a collision resistant function based on modular squaring was suggested with some theoretical arguments in [339, 343]. The collision resistant function consists of 2 modular squarings with $n = 500$ bits:

$$f(Y1 \| Y2) = \Big(chop'_{450} \Big((IV \| Y1)^2 \bmod N \Big) \| Y2 \Big)^2 \bmod N\,,$$

where $\text{chop}'_t(x)$ drops the t *most* significant bits of x, $Y1$ and $Y2$ are 450-bit blocks, and IV is a 50-bit initialization variable. The complete hash function has the following form:

$$H_i = f(H_{i-1} \| X_i)\,,$$

where H_{i-1} is a 500-bit block, and X_i is a 400-bit block. The rate of this scheme is equal to 2.56, and its security is based on the fact that $O(\log N)$ bits of squaring modulo N is hard if N is a Blum integer, i.e., $N = pq$ with $p \equiv q \equiv 3 \bmod 4$. From this it is wrongly concluded that finding two integers such that their squares agree at the 50 least significant positions is hard (a trivial collision for x is $x' = -x$). As only 50 bits of the first squaring are used as input to the second squaring, it follows that collisions can be found with a birthday attack in 2^{26} operations.

It can be shown that one can find a second preimage for f even if $k = n/4$ bits are selected, or $3n/4$ bits are chopped [263]. This approach renders random attacks infeasible, and yields a rate of 3.82. The algorithm is the same as the one presented in [122] to break the CBC-scheme with redundancy in the least significant positions. Indeed, choose a random $Y1$ and define $X = IV \| Y1$. Then define $X' = X + 2^k y$. It follows that

$$X'^2 = X^2 + 2^k(2Xy + 2^k y^2)\,.$$

The k least significant bits of X'^2 and X^2 will agree $\bmod N$ if (but not only if) $2Xy + 2^k y^2 < 2^{n-k}$. This condition is satisfied if both numbers satisfy the inequality, i.e., $y < 2^{n/2-k}$ and $2Xy \bmod N < 2^{n-k}$. A suitable y can be found with the extended Euclidean algorithm if $2^{n/2-k} \cdot 2^{n-k} > 2^n$ which is equivalent to $k < n/4$. For more details the reader is referred to [122].

It is clear that most attacks on the CBC or concatenation schemes can be thwarted by increasing the number of squarings and/or multiplications or by increasing the redundancy. The following schemes obtain a higher security level at the price of a reduced performance:

- The combination of two squaring operations [122], yielding a rate of 2:

  $$f = \left(H_{i-1} \oplus (X_i)^2\right)^2 \bmod N\,.$$

  Finding a fixed point implies solving the equation $X_i^2 = H_{i-1} \oplus FP$, where FP is a fixed point of the modular squaring. If the factorization of the modulus is unknown, this is only easy if the right hand side is equal to 0 or 1. It is an interesting research problem to classify the attacks against schemes with two (or more) squarings.

- Another possibility [122] is to use the CBC scheme with redundancy combined with a higher encryption exponent ($e = 3$). This implies that the rate will be larger than 2.

- A proposal by D. Davies [39] to reduce the amount of redundancy consists of concatenating the 64 least significant bits of H_{i-1} with the message block of 448 bits, combined with an exponent $2^{16} + 1 = 65537$:

  $$f = (X_i \parallel H_{i-1})^{65537} \bmod N\,.$$

  Because of the size of the chaining variable, this can only yield a OWHF. Moreover the speed is significantly reduced due to the larger exponent. The rate of this scheme is equal to 19.4.

It is clear that many more alternatives can be proposed. One can conclude that the most promising schemes seem to be those based on scheme 5. It is recommended to use an exponent ≥ 3 (to avoid attacks that exploit the local effect of small input differences), and to add redundancy (fix some input bits) to block multiplicative attacks. In this way one can obtain a scheme with an acceptable rate, that is secure even if factorization of the modulus is known.

Finally it is remarked that a completely different design for both an MDC and a MAC based on arithmetic in $GF(2^{593})$ was suggested in [4]. It uses a special 'addition' of 256 and 593 bits, where part of the output is recycled to the first input. This addition will be denoted with $+'$. The exponent is a function of the previous message blocks:

$$f = (X_i)^{g(H_{i-1})} \bmod (2^{593} - 1) +' H_{i-1} \, .$$

The size of X_i is equal to 593 bits, and the size of H_{i-1} is equal to 256 bits. The exponent is expanded with the function g to a 593-bit value with Hamming weight at most 30. I. Damgård has identified several weaknesses in this proposal. In case it is used as an MDC, he pointed out that fixed points can be found as follows: let P be a 593-bit value such that $H_{i-1} = P +' H_{i-1}$. Then, if $\gcd(g(H_{i-1}), 2^{593} - 1) = 1$, inserting

$$X_i = P^{g(H_{i-1})^{-1}}$$

leaves the chaining variable unchanged. In order to obtain a MAC, it is suggested to choose $IV = K$. It is clear that one can now append any block to a message and update the MAC, and it was pointed out by I. Damgård that a chosen message attack with $X_1 = 0$ will reveal the key. These attacks are thwarted in the second proposal, where the MAC is calculated as α^{H_t}, where α is a primitive element. However, inserting 256 0 blocks will leave the MAC unchanged. Other weaknesses might be identified if a detailed specification of the mapping g were available.

6.2.2.2 Provably secure schemes with large modulus

Only for very few hash functions one can state that their security is provably equivalent to a hard problem. To our knowledge, all these schemes but one are based on the hardness of factoring or discrete logarithm (the other one is based on the knapsack problem, cf. chapter 7). I. Damgård describes three provably secure constructions for claw-resistant permutations [64, 65]. Together with theorem 4.11 they can be used to construct a CRHF. The first scheme gives a direct reduction to factoring, while the other two, that are based on a one-way group homomorphism, yield only an indirect reduction to factoring and to the discrete logarithm problem respectively. J.K. Gibson [120] constructs a collision resistant function based on the discrete logarithm problem modulo a composite. Its security can be directly reduced to factoring. It is clear that for the schemes based on factoring, anyone who knows the factorization can trivially break the scheme. This implies that if the same modulus is used as for the signature scheme, the signer is able to find collisions or preimages. Moreover, interactions between the hashing and signing operations can make the system also insecure for outsiders, as

mentioned in [65]. The disadvantage of all four schemes is that they are very slow: hashing the message requires about the same time as applying RSA with full length exponent to the message.

The first two constructions by I. Damgård [64, 65] are based on factoring. A claw resistant permutation family with set size r_n is constructed as follows. Let $N = p_1 \cdot p_2 \cdots p_v$, where the p_i are k-bit prime numbers with $p_i \equiv 3 \bmod 4$, and v is the smallest integer such that $2^{v-1} \geq r_n$. For each N one can construct a set of claw resistant permutations with security parameter k (note that $n = vk$). For each $a \in \mathbb{Z}_N^*$ (the elements of $\mathbb{Z}_N$ relatively prime to N), one defines:

$$J(a) = (\left(\frac{a}{p_1}\right), \left(\frac{a}{p_2}\right), \cdots \left(\frac{a}{p_v}\right)).$$

The set QR_N of quadratic residues modulo N is the set of integers $\in \mathbb{Z}_N^*$ such that their image under J is equal to $(1, 1, \cdots 1)$. Let G_v denote the group of ± 1 v-tuples under pointwise multiplication modulo the subgroup generated by $(-1, -1, \ldots, -1)$. Then J induces a surjective homomorphism $\phi_N : \mathbb{Z}_N^* \longrightarrow G_v$. A set

$$A = \{a_0, a_1, \ldots, a_{r_n - 1} \mid a_i \in \mathbb{Z}_N^*\}$$

is called an injective set if $|\,\phi_N(A)\,| = |\,A\,|$. Then the claw resistant set of permutations is the set of permutations $\left\{f_i^{(N)}\right\}$ of QR_N, with

$$f_i^{(N)}(x) = (a_i x)^2 \bmod N, \qquad \text{for } x \in QR_N \text{ and } 0 \leq i \leq r_N - 1\,.$$

It can be shown that finding claws is as hard as factoring N, i.e., for any injective set, knowledge of a claw implies knowledge of the factorization of N. Note that checking whether a set is injective requires knowledge of the factorization of N, but it can be shown that revealing an injective set does not reveal information on how to factor N. The disadvantage of this construction is that increasing the set size will reduce the number of applications of f to hash a given number of bits, but at the same time the size of N be increased. This is because some factoring algorithms like the elliptic curve method have a running time that depends on the size of the smallest prime factor and not on the size of N. Hashing ν bits requires a modular squaring of a νk-bit integer.

An alternative construction, where the size of N does not depend on the set size r can be described as follows. Let N be an integer defined as above with $v = 2$, and let A be a set of r elements of QR_N. Then the claw resistant set of permutations is the set of permutations $\left\{f_i^{(N)}\right\}$ of QR_N, with

$$f_i^{(N)}(x) = a_i x^2 \bmod N, \qquad \text{for } x \in QR_N \text{ and } 0 \leq i \leq r - 1\,.$$

Because the size of A can only be polynomial in the size of N, this construction is faster by a factor $O(\log_2 k)$. However in this case one can only show that knowledge of a claw yields a square root of a number of the form $a_i \cdot a_j^{-1}$. This does not imply that one can factor N, and hence this is weaker from a theoretical viewpoint: although one

can show that possessing an algorithm for computing square roots of randomly chosen elements in QR_N leads to factoring N, an adversary might know a square root of the quotient of two elements "by chance", and not as a result of such an algorithm. Here hashing $\log_2(k)$ bits requires a modular squaring of a $2k$-bit integer.

An even slower scheme based on the discrete logarithm problem [64, 65] requires a full modular exponentiation to hash $\log_2(k)$ bits. For a k-bit prime with generator α, let A be a set of ν elements of $\mathbb{Z}_p^*$. Then the following set of permutations proves to be claw resistant:

$$f_i(x) = a_i \, \alpha^x \bmod p, \qquad \text{for } x \in \mathbb{Z}_p^* \text{ and } 0 \leq i \leq r-1 \,.$$

Here finding claws is equivalent to finding the discrete logarithm in $GF(p)$ with base α of the quotient of two elements of A. In order for the discrete logarithm problem to be hard, it is required that the prime factors of $p-1$ are not too small [251]. Here hashing $\log_2(k)$ bits requires a modular exponentiation of a k-bit integer. Here one can perform precomputations, as the base for the exponentiation is fixed.

The construction of J.K. Gibson [120] yields a collision resistant function based on the discrete logarithm modulo a composite. Let $N = pq$ be an n-bit product of two odd primes p, q, for which $p - 1 = 2up_1$ and $q - 1 = 2vq_1$, where p_1 and q_1 are odd primes. It is assumed that p, q, p_1, and q_1 are large and distinct and that u and v are both small. Let $a \in \mathbb{Z}_N^*$ and suppose its order is a multiple of p_1q_1 (it can be shown that almost all $a \in \mathbb{Z}_N^*$ satisfy this condition). Then the function f is defined as follows:

$$f : \Sigma^m \longrightarrow \Sigma^n : x \mapsto f(x) = a^x \bmod N \,.$$

A function evaluation requires about $1.5m$ modular multiplications of n-bit numbers. J.K. Gibson shows that finding collisions for f is equivalent to factoring N, and that this function is also one-way in both senses, as discussed in section 4.3.3.1. Note that additionally $p+1$ and $q+1$ should have a large prime factor in order to guarantee that factoring N is hard. The author also generalizes the result by showing that the fact that N is hard to factor implies that f is collision resistant and one-way for almost all $a \in \mathbb{Z}_N^*$.

6.3 A MAC proposal

F. Cohen has designed a MAC for the anti-virus package ASP [53]: this program computes a MAC for every file, and verifies this MAC before the file is loaded. The MAC calculation consists of two stages: first every block of the file is compressed, and subsequently the compressed values for every block are combined in a MAC. The only part of the scheme that is published, is the second stage of the compression. Several weaknesses of this compression will be demonstrated in this section (they have also been published in [252, 257]): the MAC is not uniformly distributed, the dependency between the chaining variables is limited, and part of the key can be recovered with an adaptive chosen message attack. One can determine exactly those key bits that are

necessary to make H_i with high probability independent of H_{i-1}. In the environment for which the scheme was designed, an adaptive chosen text attack is not realistic. However, in section 7.3.1 it will be shown that the function that is used in ASP is in fact different from the published version, and that this version can be attacked easily even in a restricted environment. In this section the attention will be focussed on the published MAC schemes based on modular arithmetic.

6.3.1 Description of the scheme

The first MAC designed by F. Cohen [52] is based on a combination of RSA encryption with public exponent e and a modulo reduction. The initial value is the RSA encryption of the secret key K and the round function can be described as follows:

$$f = \{1 + [X_i \bmod (H_{i-1} - 1)]\}^e \bmod N\,.$$

It was shown by Y.J. Huang and F. Cohen that this scheme has several weaknesses [148]: it is always possible to append message blocks and to compute the corresponding MAC. Moreover one can also change the message: when one finds a block X_i such that $X_i < H_{i-1} - 2$, the chaining variable H_i is independent of all previous blocks and of the key. In an improved version [148] the secret key K is used in every evaluation:

$$f = \{1 + (X_i \oplus K) \bmod (H_{i-1} - 1)\}^e \bmod N\,.$$

It will be shown that the reduced dependency is not improved, but that it only becomes more difficult to estimate when there is no dependency. This can be overcome with a chosen message attack that enables to calculate enough key bits to carry out all desired manipulations with a negligible probability of detection.

Below a complete description of the scheme of [148] is given; the first block is treated in a special way. The encryption of the plaintext block X with the RSA key (e, N) is denoted with $\mathbf{RSA}(X) = X^e \bmod N$. Note that both e and N are public [52].

$$\begin{aligned} H_1 &= \mathbf{RSA}\,(X_1 \oplus K) \\ H_i &= \mathbf{RSA}\,(1 + (X_i \oplus K) \bmod (H_{i-1} - 1))\,, \qquad 2 \le i \le t\,. \end{aligned}$$

The use of the RSA algorithm implies that certain parts of the algorithm are cryptographically strong. However, the scheme has rate $1.5n$, which means that it is very slow. It is stated in [52] that the performance can be improved by means of a preliminary compression of the plaintext. This can be looked at as a trade-off between two extremes: in the first case, there is no compression which results in a secure but slow compression algorithm; on the other hand, the message could be compressed with an MDC and then the result could be encrypted with the RSA algorithm. In the latter case, RSA does not directly improve the strength of the MDC. Our attack assumes that there is no preliminary compression such that the scheme takes maximal profit of the strength of RSA. An attack that takes into account the compression (but also

with a different MAC) will be discussed in section 7.3.1. A last remark concerns the length of the result. There is no reason to keep the full 512 bits of the result. Without decreasing the security level significantly, the result can be reduced to 128 or 256 bits by combining lower and higher parts with an exor.

6.3.2 Weakness of the modulo reduction

The coupling of the blocks by a modulo reduction results in specific weaknesses like non-uniform distribution of the intermediate variables. When x and y are uniformly distributed independent random integers between 0 and $N-1$, the probability that x mod y equals i is given by (for convenience x mod 0 is defined as x mod $N = x$):

$$\Pr\left[(x \bmod y) = i\right] = \frac{1}{N^2} \sum_{k=i+1}^{N} \left\lfloor \frac{N-1+k-i}{k} \right\rfloor .$$

It can be shown with some tedious manipulations that this is equivalent to

$$\frac{1}{N^2} \left(N - \left\lfloor \frac{N}{i+1} \right\rfloor i + \sum_{k=1}^{\left\lfloor \frac{N}{i+1} \right\rfloor - 1} \left\lfloor \frac{N-1-i}{k} \right\rfloor \right) .$$

For $\lfloor N/2 \rfloor \leq i \leq N-1$ this probability equals $\frac{N-i}{N^2}$ instead of $\frac{1}{N}$ for a uniform distribution. For $i = 0$ the sum can be approximated as follows [141]:

$$\frac{1}{N} \left[\ln(N-1) + 2\gamma + O(\frac{1}{\sqrt{N}}) \right] ,$$

where $\gamma = 0.577216\ldots$ is Euler's constant. Figure 6.1 shows this probability distribution for the case $N = 64$. It is clear from this diagram that the distribution is a linear function of i for $i \geq 32$.

Because K is a uniform random variable, the same holds for $X_i \oplus K$. The good randomness properties of the RSA cause the result H_i of the RSA operation also to be random. As a consequence,

$$\Pr\left[(X_i \oplus K) < H_{i-1} - 1\right] = \frac{1}{2} - \frac{2}{N} .$$

In that case, H_i is independent of all previous blocks.

Under the assumption that the plaintext blocks are independent, uniformly distributed random variables, one can easily prove the following proposition.

Proposition 6.1 *If the first l bits of H_{t-k} ($1 \leq k \leq t-1$) are equal to 1, the probability that H_t is independent of H_{t-k}—and thus of the data blocks X_1 to X_{t-k}—is approximately equal to $1 - 1/2^{k+l}$.*

This opens the door for tampering with messages by changing individual blocks and combining blocks of different plaintexts into one new plaintext with a low probability

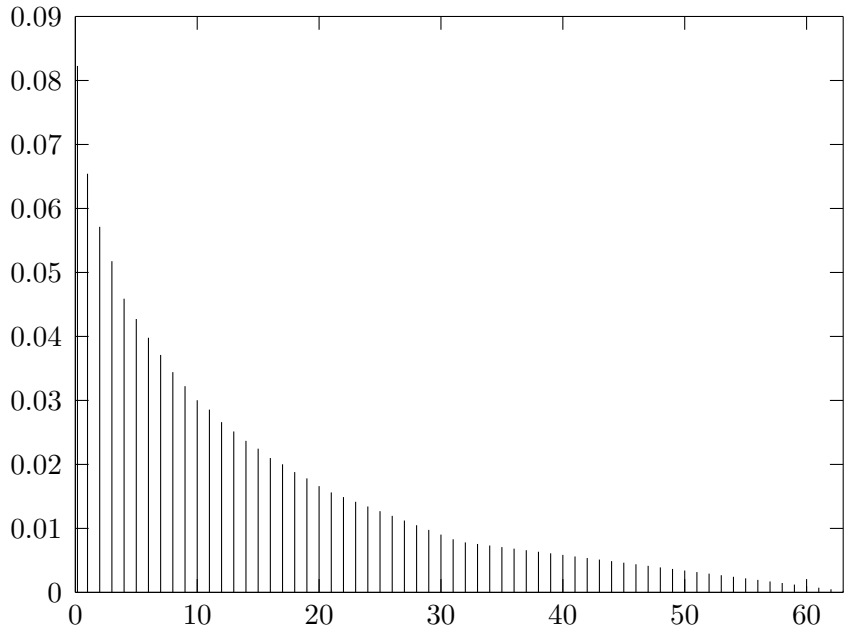

Figure 6.1: The probability distribution $\Pr[(x \bmod y) = i]$ for $N = 64$.

of detection. There especially is a very small dependence of the checksum result on the first plaintext blocks, which clearly violates the fourth condition of definition 2.3. One can wonder how an attacker can obtain the intermediate result, but this is very easy when he can compute the checksum for a shorter version of the plaintext. The error probability of an attack could be significantly lowered when he would know K or at least the first bits of K. In the following section it will be shown how the first s bits of K can be derived by means of an adaptive chosen plaintext attack.

6.3.3 Deriving the first s bits of the key K

It is assumed that an attacker can compute the checksum for messages consisting of one block X_1 and of two blocks X_1 and X_2. This implies that he can submit these messages and obtain the corresponding MAC's. In section 6.3.4 it will be shown that the attack also works for longer messages. The corresponding checksums are given by the following equations:

$$\begin{aligned} H_1 &= \mathbf{RSA}(X_1 \oplus K) \\ H_2 &= \mathbf{RSA}\left(1 + (X_2 \oplus K) \bmod (H_1 - 1)\right) . \end{aligned}$$

Because the modular exponentiation is bijective, one can extract information on the most significant bits of K by comparing H_1 and H_2. For a given X_1, one looks for an

X_2 such that H_1 equals H_2:

$$H_1 = H_2 \quad \Longleftrightarrow \quad X_1 \oplus K = 1 + [(X_2 \oplus K) \bmod (H_1 - 1)] \; .$$

If $(X_2 \oplus K) < H_1 - 1$ the modulo operation has no influence and thus

$$X_1 \oplus K = 1 + (X_2 \oplus K) \, . \tag{6.4}$$

The fact that K is unknown does not prevent an attacker from solving it for X_2 if $(X_2 \oplus K) < H_1 - 1$.

Before it is described how one can obtain X_2 some notation is introduced. An n-bit integer A will be represented as a vector with the components $(A[n], A[n-1], \ldots, A[1])$, where the most significant bits are placed first, or

$$A = \sum_{i=1}^{n} A[i] \cdot 2^{i-1} \, .$$

The vector E_i is defined as follows:

$$E_i[k] = \begin{cases} 1 & \text{for } 1 \le k \le i \\ 0 & \text{for } i+1 \le k \le n \, . \end{cases}$$

The following algorithm solves equation (6.4) for X_2.

Algorithm S — Solving for X_2.

$i = 0$
compute H_1
repeat $i = i + 1$
$X_2 = X_1 \oplus E_i$
compute H_2
until $(H_1 = H_2)$ **or** $(i \ge j)$

The expected number of trials for this algorithm is 2. It would be possible to try all n possibilities, but in order to speed up the algorithm the number of trials is limited to j, resulting in an error probability of $1/2^j$.

The attack can now be described. First it will be shown how the most significant bit of K can be determined and then it will be indicated how the attack can be extended to the higher order bits.

6.3.3.1 Deriving the most significant bit of K

The algorithm consists of two steps: in step 1 one searches for an X_1 that results in a special H_1 and in step 2 one looks for a corresponding value of X_2 that results in equality between H_1 and H_2.

Step 1

Choose X_1 and compute the corresponding value of $H_1 = \mathbf{RSA}(X_1 \oplus K)$ until

$$\begin{cases} H_1[n] &= 1 \\ H_1[i] &= 0, \quad i = n-1, n-2, \ldots, n-k+1 \\ H_1[1] &= 0 \end{cases}$$

This will require on the average 2^{k+1} RSA encryptions. If $H_1 = 0$ the attacker is very lucky, because this implies $K = X_1$. One can hence assume that $K \neq X_1$.

Step 2

Use Algorithm S to find an X_2 such that $H_1 = H_2$. Two cases have to be considered:

1. $X_1[n] \oplus K[n] = 0$ (probability $= \frac{1}{2}$). The construction of X_2 implies that $(X_2 \oplus K) < H_1 - 1$ and thus Algorithm S yields a solution after on the average 2 RSA encryptions.

2. $X_1[n] \oplus K[n] = 1$ (probability $= \frac{1}{2}$).

   (a) $X_2 \oplus K = H_1 - 1$ (probability $= \frac{n-1}{2^n}$): in this case the attacker is very lucky again because H_2 will be equal to 1 and K can easily be computed.

   (b) $X_2 \oplus K < H_1 - 1$ (probability $\simeq \frac{1}{2^k}$): Algorithm S will find a solution as in case 1.

   (c) $X_2 \oplus K > H_1 - 1$ (probability $\simeq 1 - \frac{1}{2^k}$): because $H_1[n] = 1$, the modulo operation can be replaced by a subtraction:

   $$1 + [(X_2 \oplus K) \bmod (H_1 - 1)] = 1 + (X_2 \oplus K) - H_1 + 1\,.$$

   Equality of H_1 and H_2 can be obtained if

   $$X_1 \oplus K = (X_2 \oplus K) + 2 - H_1\,.$$

   For the least significant bit, this yields the following equation:

   $$X_1[1] \oplus K[1] = X_1[1] \oplus E_i[1] \oplus K[1] \oplus H_1[1]\,.$$

   The fact that $E_i[1] = 1$ results in $H_1[1] = 1$ which contradicts the previous assumption that $H_1[1] = 0$. However, even when this would not be the case, it is very unlikely that Algorithm S would yield a solution.

It is easy to see that the above procedure allows to determine the most significant bit of K: if Algorithm S succeeds, it is decided that $K[n] = X_1[n]$, else one puts $K[n] = \overline{X_1[n]}$. There are two cases in which the algorithm fails. The fact that only j steps are applied in Algorithm S implies that it is wrongly decided with a probability of $1/2^j$ that there is no solution, but every additional RSA computation divides this error probability by 2. A more serious problem is that if $K[n] = \overline{X_1[n]}$, Algorithm S will succeed with a probability $1/2^k$. Halving this error probability requires on the average a doubling of the amount of precomputation in step 1. This leads to the conclusion that these errors will occur more frequently. The results are summarized in table 6.2.

probability	# RSA calculations	result
$\frac{1}{2}$	$2^{k+1}+2$	$K[n]$
$\frac{1}{2^{k+1}}$	$2^{k+1}+2$	$\overline{K[n]}$
$\frac{1}{2}\left(1-\frac{1}{2^k}\right)$	$2^{k+1}+j$	$K[n]$

Table 6.2: Overview of the different cases of the cryptanalytic attack.

6.3.3.2 Deriving the s most significant bits of K

The same attack can be extended to derive the first s bits of K. It is not feasible to compute *all* bits of K because the amount of computation doubles for each bit. However, an attacker does not need to know all bits of K to improve his odds significantly. When he knows the first s bits of K, he can force the first s bits of $X_i \oplus K$ to zero, which implies that $X_i \oplus K$ is smaller than H_{i-1} with a probability of $1-(1/2^s)$, when H_{i-1} is uniformly distributed. On the other hand, in case H_{i-1} is known, it is possible to find an X_i such that $X_i \oplus K < H_{i-1}$ for $2^n - 2^{n-s}$ values of H_{i-1}.

The following variation on the attack will compute the sth bit of K under the assumption that the first $s-1$ bits of K are already known.

Step 1

Choose X_1 and compute the corresponding value of $H_1 = \mathbf{RSA}(X_1 \oplus K)$ until

$$\begin{cases} H_1[i] \oplus K[i] \oplus X_1[i] &= 0, \quad i = n, n-1, \ldots, n-s+2 \\ H_1[n-s+1] &= 1 \\ H_1[i] &= 0, \quad i = n-s, n-s-1, \ldots, n-s-k+2 \\ H_1[1] &= 0\,. \end{cases}$$

This will require on the average 2^{k+s} RSA encryptions.

Step 2

As for the first bit, use Algorithm S to find an X_2 such that $H_1 = H_2$.

To derive the first s bits of K, the total number of modular exponentiations can be shown to be approximately

$$s \cdot \left(1 + \frac{j}{2}\right) + 2^{k+1} \cdot (2^s - 1)\,.$$

When $j \gg k$, the probability that these s bits are correct equals

$$\left(1 - \frac{1}{2^{k+1}}\right)^s\,.$$

6.3.4 Further extensions

In order to simplify the expressions, it was assumed that the length of the chosen plaintext was only two blocks. The attack can however be extended to longer plaintexts. It suffices to look for a plaintext that results in a very large checksum H_t. One can then add two blocks X_{t+1}, X_{t+2} to the text and with a probability H_t/N one can write

$$\begin{aligned} H_{t+1} &= \mathbf{RSA}\,(1 + (X_{t+1} \oplus K) \bmod (H_t - 1)) \\ &= \mathbf{RSA}\,(1 + (X_{t+1} \oplus K)) \\ H_{t+2} &= \mathbf{RSA}\,(1 + (X_{t+2} \oplus K) \bmod (H_{t+1} - 1))\ . \end{aligned}$$

The previous attack can now be repeated. The only difference is that the addition of 1 appears also in the first equation and thus Algorithm S is no longer necessary. On the other hand, this attack needs a plaintext with a large checksum.

6.4 Conclusion

In this chapter an overview was given of hash functions based on modular arithmetic. First weaknesses of the schemes with a small modulus were discussed. For schemes based on a large modulus, the synthetic approach of chapter 5 was used to classify many proposals. A particular scheme was identified as a promising candidate for the design of new schemes. Subsequently existing proposals and attacks were summarized. Special attention has been paid to provably secure schemes. New weaknesses were demonstrated for the scheme by Y. Zheng, T. Matsumoto, and H. Imai.

In the second part of this chapter, it was shown that the modified version of the cryptographic checksum algorithm proposed by F. Cohen and Y.J. Huang is insecure. The result of the checksum is insensitive to changes in the initial part of the plaintext and thus several manipulations are possible. Moreover, an attacker can compute the first bits of the key using an adaptive chosen text attack. Knowledge of these bits reduces significantly the chances on detecting a modification to the plaintext.

Chapter 7

Dedicated Hash Functions

If you don't know where you're going, any road will take you there. — *Lewis Carroll*

7.1 Introduction

In this chapter a number of dedicated hash functions will be discussed, i.e., algorithms that were especially designed for hashing operations. In a first section an overview will be given of MDC proposals, while in a second section the MAC proposals will be treated. Subsequently some design principles for hash functions will be discussed.

The main contribution of this chapter are three new attacks, the improvement of an existing attack, and a first evaluation of two other schemes. A second contribution is the discussion of design criteria for dedicated hash functions.

7.2 Overview of MDC proposals

The following algorithms will be described in this section: BCA, MD2, MD4 and its derivatives MD5, SHA, RIPEMD, and HAVAL, N-hash, FFT-Hash I and II, Snefru, and three hash functions based on cellular automata, namely a proposal by I. Damgård, Cellhash, and Subhash. An overview will be given of attacks on knapsack based hash functions, and 5 schemes will be discussed that are based on an additive knapsack, and 2 schemes that are based on a multiplicative knapsack.

A new attack will be presented on the BCA and on a variant of N-hash, together with a first evaluation of MD2 and SHA.

7.2.1 The Binary Condensing Algorithm (BCA)

The "Binary Condensing Algorithm" (BCA) is especially designed for the Belgian standard security system TRASEC (TRAnsmission SECurity) for EFT [319]. Its round function consists of a mixing operation and a condensation. The mixing operation

takes as input a 256-byte block that consists of 224 message bytes and 32 bytes of the chaining variable. It is based on a permutation p at byte level, two 256-byte S-boxes or substitutions S and T that operate on bytes as well and modulo 2 additions. The result of the mixing operation is condensed to 32 bytes by repeatedly exoring upper and lower halves to yield the new chaining variable. The hashcode is reduced to 64 or 128 bits again by exoring lower and upper halves.

The mixing function consists of two parallel operations that act on the same input. The input of both operations will be denoted with X, and the output with $H1$ and $H2$ respectively. The bytes of these blocks will be denoted with $X[0] \ldots X[255]$, $H1[0] \ldots H1[255]$, and $H2[0] \ldots H2[255]$. The first operation can then be described as follows:

$$H1[i] = \bigoplus_{j=0}^{255-i} S\left(\bigoplus_{k=0}^{255-j} S(X[p[k]])\right) .$$

The output of the second operation is given by

$$H2[i] = T\left(\bigoplus_{j=0}^{i} T\left(\bigoplus_{k=j}^{255} T(X[k])\right)\right) .$$

The condensation operation can be described as follows:

$$H[i] = \sum_{k=0}^{7} H1[32k+i] \oplus H2[32k+i] .$$

The final condensation to 16 bytes is computed in a similar way:

$$H_t[i] = \sum_{k=0}^{15} H1[16k+i] \oplus H2[16k+i] .$$

In the first place this function is designed to be a OWHF, and in that case a further reduction to 8 bytes is carried out. It is clear that the security of the algorithm depends strongly on the nonlinearity offered by the S-boxes, as they are the only nonlinear component. However, it can be shown that if S and T are selected randomly, the probability that they are linear is negligible [13, 131, 222].

In order to clarify this description, a simple example will be given, where the input size of the round function is restricted to 5 bytes, and the permutation is given in table 7.1. The expression for $H1[0]$ is given by

r	0	1	2	3	4	r	0	1	2	3	4
$p[r]$	4	3	1	2	0	$p^{-1}[r]$	4	2	3	1	0

Table 7.1: Permutation p and inverse permutation p^{-1} for a small example of the BCA.

$$\begin{aligned} H1[0] = S\left(S(X[4]) \oplus S(X[3]) \oplus S(X[1]) \oplus S(X[2]) \oplus S(X[0])\right) \\ \oplus S\left(S(X[4]) \oplus S(X[3]) \oplus S(X[1]) \oplus S(X[2])\right) \\ \oplus S\left(S(X[4]) \oplus S(X[3]) \oplus S(X[1])\right) \\ \oplus S\left(S(X[4]) \oplus S(X[3])\right) \\ \oplus S\left(S(X[4])\right) . \end{aligned}$$

The expression for $H1[i]$ consists of the first $5-i$ terms of $H1[0]$. In case of $H2[i]$, the expression for $H2[4]$ contains the maximal number of terms:

$$\begin{aligned} T^{-1}(H2[4]) = T\left(T(X[0]) \oplus T(X[1]) \oplus T(X[2]) \oplus T(X[3]) \oplus T(X[4])\right) \\ \oplus T\left(T(X[1]) \oplus T(X[2]) \oplus T(X[3]) \oplus T(X[4])\right) \\ \oplus T\left(T(X[2]) \oplus T(X[3]) \oplus T(X[4])\right) \\ \oplus T\left(T(X[3]) \oplus T(X[4])\right) \\ \oplus T\left(T(X[4])\right) . \end{aligned}$$

The expression for $T^{-1}(H2[i])$ consists of the first $i+1$ terms of $T^{-1}(H2[4])$.

Two weaknesses have been identified in this algorithm. The first weakness occurs if the input bytes to the round function are equal, while the second weakness depends on the properties of the permutation. If the 256 input bytes to the round function are equal to x, the following holds:

$$\begin{aligned} H1[i] \oplus H2[i] &= T(T(0)) && \text{if} && i \equiv 0 \bmod 4 \\ &= S(S(x)) \oplus T(T(T(x)) \oplus T(0)) && \text{if} && i \equiv 1 \bmod 4 \\ &= S(0) \oplus S(S(x)) \oplus T(T(T(x))) && \text{if} && i \equiv 2 \bmod 4 \\ &= S(0) \oplus T(0) && \text{if} && i \equiv 3 \bmod 4 . \end{aligned}$$

As the distance between the values that are added modulo 2 in the condensation operation is a multiple of 4 (namely 16 or 32), the internal state or the hashcode will be equal to zero, independent of the value of x. This collision for the round function also yields a collision for the hash function if all bytes of IV are equal. In that case it is also trivial to find a large number of preimages for the all zero value. This attack can however be extended to other messages: it is based on the fact that terms cancel out in the inner sums if too many values of $X[i]$ are equal. This leads to a limited number of values for the terms in the outer sum, that will also cancel out with very high probability. As a consequence, a large number of messages will be hashed to a small set. Finding a preimage in this set is much easier, and the probability to find a collision among these messages is significantly higher. In this case the attack depends on the properties of the permutation p.

The second weakness implies that if the output is reduced to 16 bytes, BCA is not a CRHF for certain choices of the permutation. The goal of the permutation is to prohibit that changes in $X[r]$ can be compensated by a modification of $X[r+1]$. However, if there exists a value r such that $p^{-1}[r+1] = p^{-1}[r] \pm 1$, a local compensation is possible. A second observation is that if there exist such an r, all terms but one in the expression

for $H1$ and $H2$ either depend on $S(X[r]) \oplus S(X[r+1])$ and $T(X[r]) \oplus T(X[r+1])$ respectively, or are independent of $X[r]$ and $X[r+1]$. The attack can now be described as follows:

Step 1

Look for an r such that $p^{-1}[r+1] = p^{-1}[r] + 1$ (the case $p^{-1}[r+1] = p^{-1}[r] - 1$ is analogous and will not be treated here). Additionally it is required that $r \equiv p^{-1}[r] \bmod 32$. In the example above, $r = 1$ satisfies the first condition.

Step 2

Look for two pairs $(X[r], X'[r])$ with $X[r] \neq X'[r]$ and $(X[r+1], X'[r+1])$ with $X[r+1] \neq X'[r+1]$ such that

$$\begin{aligned} S(X[r]) \oplus S(X'[r]) &= S(X[r+1]) \oplus S(X'[r+1]) \\ T(X[r]) \oplus T(X'[r]) &= T(X[r+1]) \oplus T(X'[r+1])\,. \end{aligned}$$

If the S-boxes S and T are random, one expects 2^{16} solutions.

Step 3

If $X[r]$ and $X[r+1]$ are replaced by $X'[r]$ and $X'[r+1]$ respectively, at most one term in the summation for $H1[i]$ will change, namely the one that contains $X[r]$ but not $X[r+1]$. This term will only occur if $0 \leq i \leq p^{-1}[r]$, and the difference is equal to

$$\Delta = S\left(S(X[r] \oplus \alpha)\right) \oplus S\left(S(X'[r] \oplus \alpha)\right) \quad \text{with} \quad \alpha = \bigoplus_{k=0}^{p^{-1}[r]-1} S(X[p[k]])\,.$$

Hence the difference pattern of $H1$ has the following form:

$$H1[i] \oplus H1'[i] = \begin{cases} \Delta & \text{for } 0 \leq i \leq p^{-1}[r] \\ 0 & \text{for } p^{-1}[r] < i \leq 255\,. \end{cases}$$

A similar observation holds for $H2$: in this case the first $r+1$ bytes of $H2$ will not change. All other bytes will be modified with the value

$$\Delta' = T\left(\beta \oplus T\left(T(X[r+1]) \oplus \gamma\right)\right) \oplus T\left(\beta \oplus T\left(T(X'[r+1]) \oplus \gamma\right)\right)\,.$$

Here $\beta = \beta_1 \oplus \beta_2$, where β_1 is independent of i and β_2 depends on i:

$$\beta = \beta_1 \oplus \beta_2 = \bigoplus_{j=0}^{r} T\left(\bigoplus_{k=j}^{255} T(X[k])\right) \oplus \bigoplus_{j=r+2}^{i} T\left(\bigoplus_{k=j}^{255} T(X[k])\right)\,,$$

and

$$\gamma = \bigoplus_{k=r+2}^{255} T(X[k])\,.$$

In order to make β_2 and thus Δ' independent of i, one can choose the $X[i]$ for $r+2 < i \leq 255$ as follows:

$$T\left(T(X[255])\right) = 0, \quad \text{and} \quad T(X[i]) = 0, \quad \text{for} \;\; r+2 < i < 255\,.$$

From this one obtains that $\beta_2 = 0$. The difference pattern of $H1$ then has the following form:

$$H2[i] \oplus H2'[i] = \begin{cases} 0 & \text{for } 0 \leq i \leq r+1 \\ \Delta' & \text{for } r+1 < i \leq 255\,. \end{cases}$$

An additional condition will now be imposed: select among the 2^{16} values solutions $X[r]$, $X[r+1]$, $X'[r]$, and $X'[r+1]$ those that yield $\Delta' = \Delta$: one expects 256 solutions.

Step 4

The difference pattern of the hashcode can now be found by adding both patterns modulo 2 and by applying the condensation operation. It can be checked that the difference pattern will be equal to 0 (or the two inputs collide) because of the condition $r \equiv p^{-1}[r] \bmod 32$. ∎

If the permutation is selected uniformly from all permutations, one can calculate the probability that the condition imposed in step 1 is satisfied. For a given r, the probability that $p^{-1}[r+1] = p^{-1}[r] + 1$ is equal to $1/256$, and this has to be divided by 32 because of the additional requirement. As there are 256 candidates for r, the probability that a suitable r can be found is given by

$$1 - \left(1 - \frac{1}{256} \cdot \frac{1}{32}\right)^{256} \approx \frac{1}{32.5} = 0.031\,.$$

If one also takes into account the case $p^{-1}[r+1] = p^{-1}[r] - 1$, one finds an overall probability of 6.1%. If one is searching for a collision and not a pseudo-collision, the first 32 bytes coming from the chaining variables can not be modified, and the probability becomes 5.3%. If additional conditions are imposed on the permutation, namely, $r \equiv 0 \bmod 32$ and $p^{-1}[r] \equiv 0 \bmod 32$, it is no longer required that $\Delta' = \Delta$.

It was shown that the BCA is not collision resistant for certain choices of the permutation, and this independently of the selection of the S-boxes. This type of differential attack could be extended in several ways, e.g., by imposing the requirement that there exists an r such that $|p^{-1}[r+1] - p^{-1}[r]|$ is 'small'. The number of elements that can not be freely chosen in the attack because of the condition on β_2 is equal to $255 - r - 1$. If r is large, a very limited number of conditions are imposed, and the attack can be converted into an attack finding a second preimage. Another extension could exploit specific weaknesses of the S-boxes. One can conclude that the security of the BCA is questionable, as the increased size of the memory does not compensate for the limited number of operations that is carried out on every input byte. It is certainly recommended to design the permutation in such a way that this type of attack becomes harder.

7.2.2 MD2

R. Rivest of RSA Data Security Inc. has designed a series of hash functions, that were named MD for "message digest" followed by a number. MD1 is a proprietary algorithm. MD2 [175] was suggested in 1990, and was recommended to replace BMAC [193]. MD3 was never published, and it seems to have been abandoned by its designer. MD4 [279, 282] and MD5 [281, 283] will be discussed in the next section. This section will give a brief description of MD2, together with a discussion of some potential attacks.

In a first stage of the algorithm, a simple 16-byte hashcode is computed and appended to the message. Subsequently a second compression algorithm is applied to the extended message. After padding the message is split in t blocks of 16 bytes. The individual bytes will be denoted with `X[i][j]`, with $0 \leq \mathtt{i} \leq t-1$ and $0 \leq \mathtt{j} \leq 15$. Both stages make use of a random 8-bit permutation, that will be denoted with `sbox[]`.

```
FOR i=0 TO 15 DO H1[i]=0; END          {initialization}
r = 0;

FOR i = 0 TO T-1 DO                    {main loop}
    FOR j = 0 TO 15 DO
        H1[j] = sbox[X[i][j] exor r];
        r = H1[j];
    END
END
```

After this stage, t is incremented by 1 and the first hashcode is appended to the message. The second stage has an internal memory of 48 bytes, or 3 16-byte blocks. At the beginning of each iteration, the first block is initialized with H_{i-1}, the second block with X_i, and the third block with $X_i \oplus H_{i-1}$. The new value of the chaining variable H_i consists of the first block of the output. Every round consists of 18 steps, and every step transforms all 48 bytes of the state. The hashcode is equal to the first 16-byte block. The pseudo-code for the second stage looks as follows:

```
FOR i=0 TO 47 DO H[i]=0; END           {initialization}

FOR i = 0 TO t-1 DO
    FOR j = 0 TO 15 DO                 {main loop}
        H[16+j] = X[i][j];
        H[32+j] = X[i][j] exor H[j];
    END

    r=0;
    FOR j = 0 TO 17 DO                  {18 rounds}
        FOR k = 0 TO 47 DO
            r= H[k] exor sbox[r];
```

```
            H[k]=r;
        END
        r = (r+j) modulo 256;
    END
END
```

The algorithm requires 54 exors and 53 table look-ups per input byte, and is clearly software oriented: the variables are updated sequentially. For the time being no analysis of the algorithm has been published.

The first hashcode is simple to analyze. Given the hashcode, it is easy to find the last 15 bytes of the message. Subsequently one can choose all other bytes except the first one, that is determined by a backward recursion. This means that if one wants to attack MD2, one will probably choose the last block, and compute the 15 corresponding bytes of the previous block. If the attack yields the first byte, it has a probability of 1/256 of being correct.

The following observations can be made on the main hashcode. The fact that the last two 16-byte blocks of the state are discarded, implies that in the last iteration one can omit the last 32 operations. The k$^{\text{th}}$ byte of the internal state after stage j ($0 \leq j \leq 17$) will be denoted with $H_j[k]$. From the hashcode one can go backwards and compute the triangle given by $H_{j+2}[15-j]$ through $H_{j+2}[15]$, for $0 \leq j \leq 15$. If one additionally chooses $H_1[15]$, one can compute from the chaining variables ($H_0[j]$, $0 \leq j \leq 15$) the remaining internal states of the first block, and the triangle containing $H_j[47-j]$ through $H_j[47]$, for $0 \leq j \leq 16$. Finding a preimage is now reduced to finding a solution for the 16 message blocks (that enter the second and third block) and the second block of the output (except for $H_{16}[31]$ that is already fixed), which means that a constraint has to be satisfied at the end of every round. The constraint in the first round can be avoided by trying all possible values for $H_1[15]$. For rounds 1 through 16, one has 16 constraints, corresponding with a success probability of 2^{-128}. Trying a single plaintext requires in this case only 31 modulo 2 additions and table look-ups. If one is searching for a pseudo-preimage, one could try to introduce symmetry (i.e., make the internal state equal for the second and third block), but this is prevented by the fact that the initial value is added to the input of the third block. From these observations it becomes clear that if the number of rounds would be smaller than 17, the hashcode would not reach all possible values. In case of a single round, the hashcode is simply a function of the chaining variables, and in case of i rounds ($2 \leq i \leq 16$), the hashcode has an entropy of $i - 1$ bytes. Hence the number of rounds is actually only one more than strictly necessary.

The most successful approach to find a collision seems to be a differential attack. One can hope (like in the case of Snefru, cf. section 7.2.6) that changes to the input will compensate each other and hence not affect the output. As the S-box is a permutation, two inputs will never yield the same output, but with a small computer program we have shown that in some cases more than one pair yields a given input and output exor. Table 7.2 gives the frequency distribution of the exor table. The value of 256 corresponds to input exor equal to 0.

# pairs	# occurrences
0	39,896
2	19,685
4	5,035
6	808
8	94
10	17
256	1

Table 7.2: Frequency distribution of the exor table for the MD2 S-box.

Modifications can compensate each other as follows: if a certain difference pattern occurs in byte k, this pattern will be transformed by the S-box if the new state of byte $k+1$ will be computed. If the previous value contains the correct difference, this change can be cancelled out if

$$\mathtt{H_j[k+1]} \oplus \mathtt{H'_j[k+1]} = \mathtt{sbox[H_j[k]]} \oplus \mathtt{sbox[H'_j[k]]} \oplus \mathtt{H_{j-1}[k+1]} \oplus \mathtt{H'_{j-1}[k+1]} \; .$$

More work has to be done in order to determine how many rounds of MD2 can be broken in this way. It would also be interesting to explore whether permutations can be found that make a differential attack harder.

7.2.3 MD4, MD5, SHA, RIPEMD, and HAVAL

MD4 is another MDC designed by R. Rivest. It was announced at the rump session of Eurocrypt'90 and was published in [279, 282]. The four other algorithms MD5, SHA, RIPEMD, and HAVAL are variants on MD4 that were proposed in a later stage.

7.2.3.1 MD4

MD4 is an iterative hash function that operates on 32-bit words. The round function takes as input a 4-word chaining variable and a 16-word message block and maps this to a new chaining variable. All operations are defined on 32-bit words. The transformation consists of 3 rounds, and each round consists of 16 steps. In every step one word of the chaining variables is modified as follows: a message word and a non-linear function of the three other chaining variables is added, and the result is rotated over a variable number of positions. This mapping is reversible, i.e., one can go backwards easily. Every message word is used exactly once in every round, but in a different order. After the 3 rounds, the previous chaining variable is added to the new chaining variable in order to make the function irreversible. The complete description specifies a padding rule and an initial value.

```
{nonlinear functions at bit level: multiplex,majority,exor}
f(j,x,y,z) = (x and y) or (not(x) and z)         ( 0 <= j <= 15)
f(j,x,y,z) = (x and y) or (x and z) or (y and z) (16 <= j <= 31)
f(j,x,y,z) = x exor y exor z                     (32 <= j <= 47)

{added constant (hexadecimal)}
K(j) = 0        ( 0 <= j <= 15)
K(j) = 5A827999 (16 <= j <= 31)
K(j) = 6ED9EBA1 (32 <= j <= 47)

{selection of message word}
r(j) =  j                   ( 0 <= j <= 15)
r(j) = (4*(j-16)) modulo 15 (16 <= j <  31)             r(31)=15
r(32..47) = 0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15

{amount for rotate left (rol)}
s( 0..15) = 3,7,11,19,3,7,11,19,3,7,11,19,3,7,11,19
s(16..31) = 3,5, 9,13,3,5, 9,13,3,5, 9,13,3,5, 9,13
s(32..47) = 3,9,11,15,3,9,11,15,3,9,11,15,3,9,11,15

{initial value (hexadecimal)}
h0 = 67452301  h1 = EFCDAB89  h2 = 98BADCFE  h3 = 10325476
```

It is assumed that the message after padding consists of t 16-word blocks that will be denoted with X[i][j], with $0 \leq$ i $\leq t-1$ and $0 \leq$ j ≤ 15. All additions are additions modulo 2^{32}. The pseudo-code for MD4 is then given below.

```
A = h0; B = h1; C = h2; D = h3;

FOR i = 0 TO t-1 DO
    FOR j = 0 TO 47 DO
        TEMP = A + f(j,B,C,D) + X[i][r(j)] + K(j);
        A = D; D = C; C = B; B = rol(TEMP,s(j));
    END
    h0 = h0 + A; h1 = h1 + B; h2 = h2 + C; h3 = h3 + D;
END
```

This algorithm is clearly software oriented: all variables are updated sequentially, and extensive use is made of standard processor instructions. The only problem for most programming languages is the rotate operation, that has to be simulated using two shifts. The order of the bytes in a word is chosen such that implementations on the slower little endian architectures[1] have the smallest overhead.

[1]In little endian processors like the 80x86 family, the least significant byte occupies the position with the lowest address. In big endian processors like the 680x0 and the RISC processors, the least significant byte occupies the position with the highest address.

Although the scheme is intended to yield a collision resistant function, the chaining variables and the message block are processed in a different way. The basic configuration allows to construct fixed points easily, but they offer no serious threat as the message length is included and IV is fixed (cf. section 7.4). If the third round is omitted, the scheme is weak: R. Merkle showed in an unpublished result that if the third round is omitted, one can easily find two messages differing in only 3 bits that hash to the same value. The attack exploits the choice of the rotations and the order of the message words. It works for 99.99% of all initial values, including the one that was chosen. B. den Boer and A. Bosselaers [81] showed how to produce a collision if the first round is omitted. The idea is to use only different message words in the 8 middle steps of both the second and third round, that both depend on message blocks 1, 2, 5, 6, 9, 10, 13, and 14. Moreover the attack exploits the fact that rotations are only over odd positions. Both attacks require only a millisecond of computation on a PC, and the second attack leaves 320 bits of freedom. It is still an open problem whether these can be used to produce a collision for all three rounds. Finally it is remarked that E. Biham [24] discussed a differential attack on MD4, that seems to be feasible for two rounds. It also relies heavily on the order of the message words.

7.2.3.2 MD5

In consequence of these attacks, R. Rivest realized that the security level of MD4 was not as large as he intended, and he proposed in 1991 a strengthened version of MD4, namely MD5 [281, 283, 295]. An additional argument was that although MD4 was not a very conservative design, it was being implemented fast into products. MD5 has an additional round, and has a multiplexer function in the first and second round. The order in which the message words are used has been changed, and 16 different rotation amounts were introduced (to avoid certain attacks and to increase the avalanche effect). Two remarkable changes are that every step now has a unique additive constant, which requires about 250 additional bytes of storage, and the core of the algorithm is modified. This yields the following description:

```
{nonlinear functions at bit level: multiplex,multiplex,exor,-}
f(j,x,y,z) = (x and y) or (not(x) and z)         ( 0 <= j <= 15)
f(j,x,y,z) = (x and z) or (y and not(z))         (16 <= j <= 31)
f(j,x,y,z) = x exor y exor z                     (32 <= j <= 47)
f(j,x,y,z) = (x or not(z)) exor y                (48 <= j <= 63)

{added constant}
K(j) = first 32 bits of sin(j+1)     ( 0 <= j <= 63)

{selection of message word}
r(j) =  j                          ( 0 <= j <= 15)
r(j) =  (1+5*(j-16)) modulo 16   (16 <= j <= 31)
r(j) =  (5+3*(j-32)) modulo 16   (32 <= j <= 47)
```

```
r(j) =  7*(j-48) modulo 16      (48 <= j <= 63)

{amount for rotate left (rol)}
s( 0..15) = 7,12,17,22,7,12,17,22,7,12,17,22,7,12,17,22
s(16..31) = 5, 9,14,20,5, 9,14,20,5, 9,14,20,5, 9,14,20
s(32..47) = 4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23
s(48..63) = 6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21

{initial value (hexadecimal)}
h0 = 67452301  h1 = EFCDAB89  h2 = 98BADCFE  h3 = 10325476
```

The pseudo-code for MD5 is then given below.

```
A = h0; B = h1; C = h2; D = h3;

FOR i = 0 TO t-1 DO
    FOR j = 0 TO 63 DO
        TEMP = A + f(j,B,C,D) + X[i][r(j)] + K(j);
        A = D; D = C; C = B; B = B + rol(TEMP,s(j));
    END
    h0 = h0 + A; h1 = h1 + B; h2 = h2 + C; h3 = h3 + D;
END
```

B. den Boer noted that an approximate relation exists between any four consecutive additive constants. Moreover, together with A. Bosselaers he developed an attack that produces pseudo-collisions, more specifically they can construct two chaining variables (that only differ in the most significant bit of every word) and a single message block that yield the same hashcode [83]. The attack takes a few minutes on a PC. This means that one of the design principles behind MD4 (and MD5), namely to design a collision resistant function is not satisfied. T. Berson [18] discussed how differential techniques could be used to break a single round of MD5, but his attack is still far off from being effective against all four rounds together.

7.2.3.3 SHA

On January 31, 1992, NIST (National Institute for Standards and Technology, USA) published in the Federal Register a *proposed* Secure Hash Standard (SHS) [112] that contains the description of the Secure Hash Algorithm (SHA). This hash function is designed to work with the Digital Signature Algorithm (DSA) proposed in the Digital Signature Standard (DSS) [111]. The algorithm is clearly inspired by MD4, but it is a strengthened version. The size of the chaining variables is increased to 5 32-bit blocks, which results in a total of 160 bits. This seems to be in line with the considerations of the feasibility of a birthday attack, as discussed in section 2.5.1.3. The number of steps per round has been increased to 20, and the number of rounds has been increased to 4, like for MD5. The increase of the number of steps per rounds implies that every

word of the chaining variable is transformed 4 times per round, just like for MD4 and MD5. The main operation has been modified and now involves a rotation of two variables. A very important change is that starting from 16, the message word is computed as the exor of four previous message words. A modification that affects only the implementation is that the bytes in a word are ordered differently.

```
{nonlinear functions at bit level}
{nonlinear functions at bit level: multiplex,exor,majority,exor}
f(j,x,y,z) = (x and y) or (not(x) and z)         ( 0 <= j <= 19)
f(j,x,y,z) = x exor y exor z                     (20 <= j <= 39)
f(j,x,y,z) = (x and y) or (x and z) or (y and z) (40 <= j <= 59)
f(j,x,y,z) = x exor y exor z                     (60 <= j <= 79)

{added constant (hexadecimal)}
K(j) = 5A827999         ( 0 <= j <= 19)
K(j) = 6ED9EBA1         (20 <= j <= 39)
K(j) = 8F1BBCDC         (40 <= j <= 59)
K(j) = CA62C1D6         (60 <= j <= 79)

{initial value (hexadecimal)}
h0 = 67452301 h1 = EFCDAB89 h2 = 98BADCFE h3 = 10325476 h4 = C3d2E1F0
```

The pseudo-code for SHA then has the following form (again all additions are modulo 2^{32}):

```
A = h0; B = h1; C = h2; D = h3; E=h4;

FOR i = 0 TO t-1 DO
    FOR j = 0 TO 79 DO
        IF (j > 15) THEN
        X[i][j] = X[i][j-3] exor X[i][j-8] exor X[i][j-14]
                  exor X[i][j-16];
        END
        TEMP = rol(A,5) + f(j,B,C,D) + E + X[i][j] + K(j);
        E = D; D = C; C = rol(B,30); B = A; A = TEMP;
    END
    h0 = h0 + A; h1 = h1 + B; h2 = h2 + C; h3 = h3 + D; h4 = h4 + E;
END
```

The design principles of the algorithm are not public, and no evaluation of SHA has been published for the time being. The algorithm might still be changed in response to comments supplied to NIST.

The fact that the message words are not simply permuted certainly increases the strength of the algorithm. It implies that it is no longer possible to change a small number of input bits to the 4 rounds: any message bit affects between 28 (words 10, 11,

and 12) and 36 (words 2 and 3) steps. At bit level, the transformation is a systematic linear $(80, 16, 23)$ code, or a code with length $n = 80$, size $k = 16$ and minimum distance 23. This means that on a given bit position, the difference modulo 2 between two set of 80 message words will have a Hamming weight of at least 23. This is not optimal, as it is shown in [145] that the minimum distance of the best linear codes with that length and dimension lies between 28 and 32. This choice offers probably the best compromise between speed and security.

7.2.3.4 RIPE-MD

In the framework of the EEC-RACE project RIPE [259, 317] a new version of MD4 was developed [277]. The rotations and the order of the message words are modified to decrease vulnerability against previous attacks. Moreover, two instances of the algorithm, that only differ in the constants, are run in parallel, but with the same input. This increases the internal memory to 256 bits. After processing a 512-bit block, both chaining variables are combined together with the initial chaining variables.

7.2.3.5 HAVAL

While SHA and RIPEMD can be considered variants on MD4, HAVAL is an extension of MD5. It was proposed by Y. Zheng, J. Pieprzyk, and J. Seberry at Auscrypt'92 [346]. The first modification in HAVAL is that the size of both message block and chaining variable is doubled to respectively 32 and 8 words. The number of rounds can be 3, 4, or 5 and each round consists of 32 steps. The simple nonlinear functions are replaced by highly nonlinear functions of 7 variables, that satisfy some specific properties like the Strict Avalanche Criterion or SAC (cf. chapter 8). Moreover a single function is used in every round, but in every step a different permutation is applied to the inputs. Again a new message order has been introduced, and every step (except for those in the first round) uses a different additive constant. Two rotations over 7 and 11 positions have been introduced. The core of the algorithm has now the following form:

```
TEMP = rol(f(j,A,B,C,D,E,F,G),25) + rol(H,11) + X[i][r(j)] + K(j);
H = G; G = F; F = E; E = D; D = C; C = B; B = A; A = TEMP;
```

At the end of the algorithm, on can apply a folding operation to reduce the size of the hashcode to 16, 20, 24, or 28 bytes. The choice in the number of rounds and in the size of the output yield 15 different versions of the algorithm. The attack on MD5 by B. den Boer and A. Bosselaers [83] is not directly applicable to HAVAL because of the additional rotation operation that is applied to the chaining variable H.

7.2.4 N-hash

N-hash is a hash function designed by S. Miyaguchi, M. Iwata, and K. Ohta [226, 229]. The size of both chaining variable and plaintext blocks is equal to 128 bits. The basic

building block of N-hash is an encryption function denoted with $R(H_{i-1}, X_i)$. This function is reversible with respect to X_i, which means that if H_{i-1} is known, X_i can be computed from $R(H_{i-1}, X_i)$. The encryption consists of 8 rounds or processing stages and is preceded by an addition to X_i of a constant and of H_{i-1} with upper and lower parts interchanged. The key input to every round is equal to $H_{i-1} \oplus V_i$, where V_i is a round dependent constant. The complete algorithm can now be described as follows:

$$f = R(H_{i-1}, X_i) \oplus H_{i-1} \oplus X_i \,.$$

One round consists of the application of 4 F functions, that are similar to the F function of FEAL [225, 228]. In fact every round is equivalent to 2 rounds of a Feistel cipher [104, 105]. The nonlinearity comes from an addition modulo 256 followed by a rotate left over 2 positions. The S-boxes show several weaknesses, e.g., an input exor of $\mathtt{80}_\mathtt{x}$ always leads to an output exor of $\mathtt{02}_\mathtt{x}$.

In [226] the question was raised whether the round function f is collision resistant. In 1989, B. den Boer [82] showed how to construct three pseudo-preimages hashing to a given value. He found constants α and β such that

$$R(H_{i-1} \oplus \alpha, X_i \oplus \beta) = R(H_{i-1}, X_i) \oplus \alpha \oplus \beta \,,$$

which implies that H_i will remain unchanged. A possible solution is to choose α equal to $a\|c\|b\|c\|a\|c\|b\|c$ and β equal to $b\|c\|a\|c\|b\|c\|a\|c$, with $a = \mathtt{8280}_\mathtt{x}$, $b = \mathtt{8080}_\mathtt{x}$, and $c = \mathtt{0000}_\mathtt{x}$. After the initial addition of H_{i-1} with upper and lower part exchanged, the data input exor of the first round is equal to $\alpha \oplus \beta = d\|d\|d\|d$ with $d = \mathtt{2000}_\mathtt{x}$. Together with an input exor of α in the key, this yields an output exor of $\alpha \oplus \beta$ after the first round. It is clear that for all subsequent rounds input and output exor will remain constant, which means that this attack is independent of the number of rounds.

E. Biham and A. Shamir exploited other weaknesses in the S-boxes to obtain an iterative characteristic of 3 rounds with probability 2^{-16} [21]. It can be described as follows. Let $\psi = \mathtt{80608000}_\mathtt{x}$ and let $\phi = \mathtt{80E08000}_\mathtt{x}$, then the following pattern of input and output exors is obtained:

$$(\psi, \psi, 0, 0) \longrightarrow (0, 0, \phi, \phi) \longrightarrow (\psi, \psi, \phi, \phi) \longrightarrow (\psi, \psi, 0, 0) \,.$$

Here the first and second transitions have probability 1/256, while the third transition has probability 1. This means that for N-hash with $3r$ rounds, a second preimage can be found with probability 2^{-16r}, and a collision can be produced in $2^{8+16(r-1)}$ operations, which means that it is faster than a birthday attack for variants of N-hash with up to 12 rounds. Note however that this attack does not apply to N-hash as the suggested number of rounds is equal to 8. Further study is necessary to find good characteristics in case the number of rounds is not divisible by three. Note that this weakness would be much more important if the function R would be used as a block cipher: in that case the characteristic can be one or two rounds shorter than the block cipher, and it would be possible to break the cipher faster than exhaustive search for the 128 key bits for any number of rounds up to about 24.

A new version of N-hash appeared in a Japanese contribution to ISO [161]. Here it is suggested to interchange X_i and H_{i-1}. However, in that case *any change* in X_i for which upper and lower part are equal (this implies that the exchange operation at the beginning of the algorithm has no effect) yields the following pattern of input and output exors:

$$(\psi_1, \psi_2, \psi_1, \psi_2) \longrightarrow (\psi_1, \psi_2, 0, 0) \longrightarrow (0, 0, \psi_1, \psi_2) \longrightarrow (\psi_1, \psi_2, \psi_1, \psi_2)\,.$$

This pattern is obtained *in all cases*, which means that if the number of rounds is divisible by three, this variant of N-Hash is totally insecure!

One can conclude that several weaknesses have been identified in N-Hash and its variants. For the time being no collisions were reported for the original proposal, but serious doubts have been raised about its security.

7.2.5 FFT-Hash I and II

FFT-Hash I and II are MDC's proposed by C. Schnorr [297, 298]. The algorithms are based on a bijective transformation that is built from a discrete Fourier transform and a polynomial recursion. The security relies on the fact that polynomial transformations of high degree over a finite field generate local randomness [234].

FFT-Hash I was presented at the rump session of Crypto'91. The input to the round function consists of 16 16-bit words that will be denoted with $X[0]$, $X[1]$,..., $X[15]$. The following transformation is applied:
Step 1: $(X[0], X[2], \ldots X[14]) = FT_8(X[0], X[2], \ldots X[14])$.
Step 2: FOR $i = 0, 1, \ldots 15$ DO $X[i] = X[i] + X[i-1]X[i-2] + X[X[i-3]] + 2^i \bmod p$.
Step 3 and 4 are identical to Step 1 and 2 respectively. Here $p = 2^{16} + 1$, and all operations are modulo p, except the indices, which are taken modulo 16. The operations FT_8 is the Fourier transform with the primitive root 2^4 of order 8:

$$FT_8 : \mathbb{Z}_p^8 \longrightarrow \mathbb{Z}_p^8 : (a_0, \ldots, a_7) \mapsto (b_0, \ldots, b_7), \text{ with } b_i = \sum_{j=0}^{7} 2^{4ij} a_j\,.$$

Finally the output words are reduced modulo p. From this transformation a compression function is obtained by selecting words $X[8], \ldots, X[15]$ as output. The hash function is now defined by specifying that the first 8 input words are used for the chaining variables and by fixing a padding procedure and an initial value. It should be noted that the bijective transform can easily be inverted, which means that finding a pseudo-preimage is trivial. It can be shown that for random inputs, the output of the hash function has a distribution that is close to uniform.

An important disadvantage of the scheme is that to store a single variable modulo p, two 16-bit words are necessary. Moreover the serial character of the recursion virtually excludes efficient hardware implementations. The following weaknesses were found in the scheme:

- if $X[i-1]$ and $X[i+1]$ are both equal to 0, a change in $X[i]$ does not propagate to the other variables,

- the indirect addressing can be exploited either if the index points to a variable with a given value or does not point to a given value,
- the FT_8 only affects the variables with an even index; moreover if a subset of 8 inputs or outputs is known, the remaining inputs and outputs can be computed.

These weaknesses lead to two independent attacks by J. Daemen, A. Bosselaers, R. Govaerts, and J. Vandewalle [62] and by T. Baritaud, H. Gilbert, and M. Girault [12]. Both attacks require only 2^{23} partial evaluations of the hash function and lead to multiple collisions, i.e., a large set of messages that hash to the same value. The first attack requires 3 message blocks, and varies the second word of the first message block, while the second attack uses 2 message blocks and varies the last word of the first message block.

These attacks resulted in a new version that was presented at Eurocrypt'92 [298]. It differs only in minor points from the previous scheme.
Step 1: FOR $i = 0, 1, \ldots 15$ DO $X[i] = X[i] + X^*[i-1]X^*[i-2] + X[i-3] + 2^i \bmod p$.
Step 2: $(X[0], X[2], \ldots X[14]) = FT_8(X[0], X[2], \ldots X[14])$.
Step 3: $(X[1], X[3], \ldots X[15]) = FT_8(X[1], X[3], \ldots X[15])$.
Step 4 is identical to Step 1. Here the * indicates that if this variable is 0, it will be replaced by 1. Three weeks later S. Vaudenay announced that he had found collisions for FFT-Hash II; he presented his attack at Crypto'92 [321]. In a further variant C. Schnorr suggests to destroy the reversibility by adding the input of the transformation to the output [298]. In view of the slow speed in both hardware and software, adding more steps seems to be unacceptable.

7.2.6 Snefru

R. Merkle suggested in 1989 a software oriented one-way hash function that he called Snefru [214]. First a reversible transformation is specified that acts on a 512-bit block, that consists of 16 32-bit words `X[0],...,X[15]`. The transformation contains at least two passes, where a single pass has the following form:

```
FOR j = 0 TO 3 DO
    FOR i = 0 TO 15 DO
       k = 1 - ((i div 2) mod 2);
       delta = sbox[k][first byte of X[i]];
       X[i-1] = X[i-1] exor delta;
       X[i+1] = X[i+1] exor delta;
   END
   rotate X over s(j) bytes;
END
```

Here the indices are taken modulo 16, and the values of `s(j)` are 2, 1, 2, and 3. Every pass uses two S-boxes that each contain 256 32-bit words, which corresponds to 2 Kbytes. After all passes have been completed, the input is added modulo 2 to the output, which makes the function irreversible.

Like in the case of FFT-Hash I, a compression function is obtained by selecting words `X[12],...,X[15]` as output. This means that a 384-bit message block will enter the round function. Here however it is claimed that this function is collision resistant, which means that it is not necessary to fix an initial value. The hash function is now defined by specifying that the first 4 input words are used for the chaining variables.

Slower and more secure variants have also been suggested, where the size of the chaining variable is equal to 256 bits. What is interesting about the algorithm is that the design philosophy and all choices made in the design are justified in [214]. In order to avoid allegations about a trapdoor in the S-boxes, they are computed with a public pseudo-random generator from random numbers that were published in 1955.

E. Biham and A. Shamir obtained the following results on Snefru [22]: finding a collision for Snefru with p passes requires $2^{12.5+16(p-2)}$ operations, and finding a second preimage takes $2^{24+32(p-2)}$ operations. Even if the S-boxes are unknown, a collision can be found in $2^{20.5}$ operations for $p = 2$ and 2^{49} operations for $p = 3$. The collision attacks remain faster than a birthday attack even if the size of the chaining variable is increased up to 224 bits. The main observation behind the attack is that if the third byte of $X[7]$ and of $X[9]$ are modified, there is a high probability (2^{-40} for $p = 2$) that these changes cancel out at certain positions. As a consequence the last 4 words are not modified. The attacks for $p > 2$ exploit the reversibility of the transformation. A similar observation was made independently by J. Williams [326], who obtained the same success probability by modifying the third byte of $X[7]$ through $X[11]$. His attack can be extended to 3 passes, for which it has a success probability of 2^{-72}, but fails for $p \geq 4$.

As a consequence of theses attacks, one should use 6 and preferably 8 passes, possibly combined with an increased size of the chaining variable. However, these measures increase the size of the S-boxes, and decrease the performance.

7.2.7 Hash functions based on cellular automata

It was suggested by S. Wolfram [329] to use a one-dimensional cellular automaton (s bits in size) for pseudo-random bit generation.

$$x_i(j) \longleftarrow x_{i-1}(j-1) \oplus [x_{i-1}(j) \vee x_{i-1}(j+1)] \ .$$

Here the indices have to be taken modulo s. The bit generator $b(x)$ starts from a random x and outputs the sequence $x_i(0)$. Let $b_{c-d}(x)$ denote the string $x_c(0)$, $x_{c+1}(0),\ldots,x_d(0)$. I. Damgård tried to use the one-way property of this generator to design a collision resistant function as follows:

$$f = b_{c-d}(X_i \parallel H_{i-1} \parallel Z) \, .$$

where Z is a random r-bit string added to make "trivial" collisions more sparse and harder to find. The value of c should not be chosen too small in order to make F complete. Proposed values are $s = 512$, $r = 256$, $c = 257$, and $d = 384$. The advantage of the scheme lies in the fact that in a parallel VLSI implementation every step would require about d clock cycles (for the given example 3 clock cycles/bit).

It was shown by O. Staffelbach and W. Meier [311] that the Wolfram pseudo-random bit generator is in fact very weak for $s < 1000$ (S. Wolfram proposed $s = 127$). Moreover J. Daemen, A. Bosselaers, R. Govaerts, and J. Vandewalle [61] found several weaknesses in the hash function proposed by I. Damgård. They proved that for the proposed parameters, for most inputs there exists an $r > 0$ such that the output of f is independent of bits x_{126-r} through x_{126}. Moreover, if the input starts with a pattern of alternating zeroes and ones, the output is independent of a number of bits that follow this pattern. These weaknesses yield multiple collisions for f and also collisions for the hash function.

J. Daemen, J. Vandewalle, and R. Govaerts subsequently proposed Cellhash, a new hash function that is based on a cellular automaton [61]. The elementary operation computes the new 257-bit chaining variable from the old chaining variable and a 257-bit message block (with the first bit always equal to 0). For a fixed message block the transformation is bijective and reversible; it can be described as follows:

$$\begin{array}{lll} \textbf{Step 1} & h_i = h_i \oplus (h_{i+1} \vee \overline{h_{i+2}})\,, & 1 \le i \le 256 \quad h_0 = \overline{h_0 \oplus (h_1 \vee \overline{h_2})}\,. \\ \textbf{Step 2} & h_i = h_{i-3} \oplus h_i \oplus h_{i+3} \oplus x_i\,, & 0 \le i \le 256. \\ \textbf{Step 3} & h_i = h_{10 \cdot i}\,, & 0 \le i \le 256. \end{array}$$

The indices should be considered modulo 257. Before the hashing operation, the message is subjected to a preprocessing to introduce redundancy, that is of paramount importance for the security. If the message after padding contains t 32-bit words that are denoted with $X[0]$ through $X[t-1]$, the expanded message contains $t' = 8 \cdot t$ 32-bit words that are given by:

$$X[j] = X[(j \bmod 8 + j \operatorname{div} 8) \bmod t], \qquad \text{with} \quad 0 \le j \le t' - 1\,.$$

Eight of these words are combined into a single 256-bit block. The chaining variables are initialized with the all zero value.

The design is clearly hardware oriented: all operations are very simple and act in parallel on all bits. The disadvantage of the redundancy scheme is that an extra register of 224 bits is required. In software the third step would be very slow. The choice of a 257-bit chaining variable guarantees that the mapping is bijective and avoids circular symmetric patterns. The designers also argue that differential attacks are very unlikely to succeed. Because of the invertibility, finding a pseudo-preimage and a pseudo-collision requires only a single operation. As a consequence, one can obtain a preimage in about 2^{128} operations with a meet in the middle attack. To avoid problems with the redundancy, 8 32-bit words in the beginning and in the middle should be kept constant. A single iteration step has only nonlinear order two, which implies that if the hash result is written as a function of the message bits, the maximal number of factors in the algebraic normal form (cf. chapter 8) is 127, while for a random Boolean function this is equal to 256. Moreover, only a small subset of terms are possible. As a consequence analysis based on formal coding has shown to be feasible if every message bit would be used only 4 times [94]. A meet in the middle attack seems however to be hard, as the inverse function is much more complex. With this technique it was also

investigated whether it is easy to find fixed points. If all 32-bit words in the plaintext are equal, this yields rather simple equations, but for the time being they have not been solved.

An improved version of Cellhash, Subhash, was proposed by the designers in [63]. The multiplier 10 is replaced by 12 in Step 3, and the redundancy scheme for the message is simplified: the cyclic structure is removed by prepending and appending zero blocks to the message. As a consequence of the modifications, the resistance to differential attacks has been improved.

7.2.8 Hash functions based on the knapsack problem

First the knapsack problem will be described. Subsequently a brief summary will be given of algorithms to solve the knapsack problem, and finally some hash functions based on the knapsack scheme and on related problems will be discussed.

7.2.8.1 The knapsack problem

The knapsack problem is the following NP-complete problem [116]: given a finite set U, for each $u \in U$ a size $s(u) \in \mathbb{N}$ and a value $v(u) \in \mathbb{N}$, and positive integers B and K, does there exist a subset $U' \subseteq U$ such that

$$\sum_{u \in U'} s(u) \leq B \quad \text{and} \quad \sum_{u \in U'} v(u) \geq K?$$

If $s(u) = v(u),\ \forall u \in U$ this problem is called the subset sum problem. In cryptography, a special case of the subset sum problem is considered, namely given a set of n b-bit integers $\{a_1, a_2, \ldots, a_n\}$, and a b'-bit integer S (with $b' \approx b + \log_2 n$), find a vector X with components x_i equal to 0 or 1 such that

$$\sum_{i=1}^{n} a_i \cdot x_i = S\,.$$

For historical reasons we will use the name knapsack to indicate this problem.

For cryptographic applications, one defines the function $f(X) = S$ for given a_i, and hopes that this function is a one-way function. The first application of this function was the 1978 Merkle-Hellman public key scheme [210]. Since then many variants have been proposed, comprising schemes with digital signature capability. However, almost all schemes have been broken [92, 32]. Most of these attacks made use of the special structure that was built into the knapsack to obtain a trapdoor one-way function. If the knapsack is used to construct a one-way function, no special structure is required. However one should note that more general attacks were published that are not based on any structure, which leads to the following question: is the subset sum only hard in worst case, while the average instance is easy, which renders knapsacks useless for cryptography. In case of a negative answer, the problem remains to determine the parameters of the knapsack such that the problem is hard. It is certainly true that many instances of the knapsack problem are easy, and that certain instances allow to

derive partial information on X. Moreover, if X is a random vector, the distribution of the result is not uniform [37]. Other weaknesses of the problem are caused by the linearity, which is very undesirable from a cryptographic standpoint. An argument in favor of the knapsack [66] is that it can be shown that deciding in general whether a given knapsack induces an injective mapping is co-NP complete; a collision is of course a witness of non-injectiveness (if the knapsack compresses its input, the knapsack is not injective, but this does not imply that finding a witness is easy).

The attractivity of the subset sum problem lies in the fact that implementations in both hardware and software would be much faster. Processing an n-bit message requires on average $n/2$ additions of b-bit integers ($O(nb)$ operations), while schemes based on the hardness of factoring or discrete logarithm require typically $1.5n$ modular multiplications ($O(n^3)$ operations).

7.2.8.2 Solving the knapsack problem

First note that in [149] it is shown that the subset sum problem becomes harder if $\mid n - b' \mid$ decreases. For small b' this mapping will be surjective and almost regular (cf. section 4.3.2.5), while for large b' the mapping becomes almost injective. If the knapsack is to be used for hashing, one will of course have that $b' < n$.

The density of a knapsack is defined as

$$d = \frac{n}{b''} \approx \frac{n}{b},$$

where $b'' = \log_2 \max_{1 \le i \le n} a_i$. Brickell and Lagarias-Odlyzko have shown that if $d \le 0.6463\ldots$, the knapsack problem can be solved by reducing it to finding a short vector in a lattice [32]. Coster-LaMacchia-Odlyzko-Schnorr [60] and Joux-Stern have extended the Lagarias-Odlyzko algorithm to solve problems with $d \le 0.9408$. Because of the reduction to lattice problems, no exact bounds are known, but in practice the running time is $O(n \cdot b^3)$.

A second algorithm by R. Schroeppel and A. Shamir solves the knapsack problem with $b \approx n$ in time $O(2^{n/2})$ and space $O(2^{n/4})$.

A third algorithm designed by Ph. Godlewski and P. Camion [126] finds a collision for a knapsack in time $O(n \log_2 n)$ under the condition that

$$n \ge 2^{2\sqrt{b'}}.$$

In the same paper, the authors describe a fourth attack that finds a preimage in $O(2^{2\sqrt{b'}})$ operations under the condition that

$$n \ge 2\sqrt{b'} \cdot 2^{\sqrt{b'}}.$$

A fifth algorithm that was proposed by P. Camion to break a multiplicative knapsack ([36], cf. section 7.2.8.4) was extended by P. Camion and J. Patarin [37] to solve the following additive knapsacks. If the number of operations (and the storage requirement) is equal to $(n/r)2^r$, a preimage can be found if

$$n \ge 2^{b'/r + \log_2 r - 1}.$$

Note that in [37] only the cases $r = 32$ and $r = 16$ are considered, but the parameters of the attack can easily be generalized. First the attack solves smaller subproblems in the subgroups modulo r-bit primes using the birthday attack; then the solutions are combined with the Chinese remainder theorem. In the case of a collision attack, n can be two times smaller.

Figure 7.1 indicates for which choices of b and n an attack is known that requires less than 2^{64} computations and storage. The 'safe' area is the region between curve 1 and curve 5.

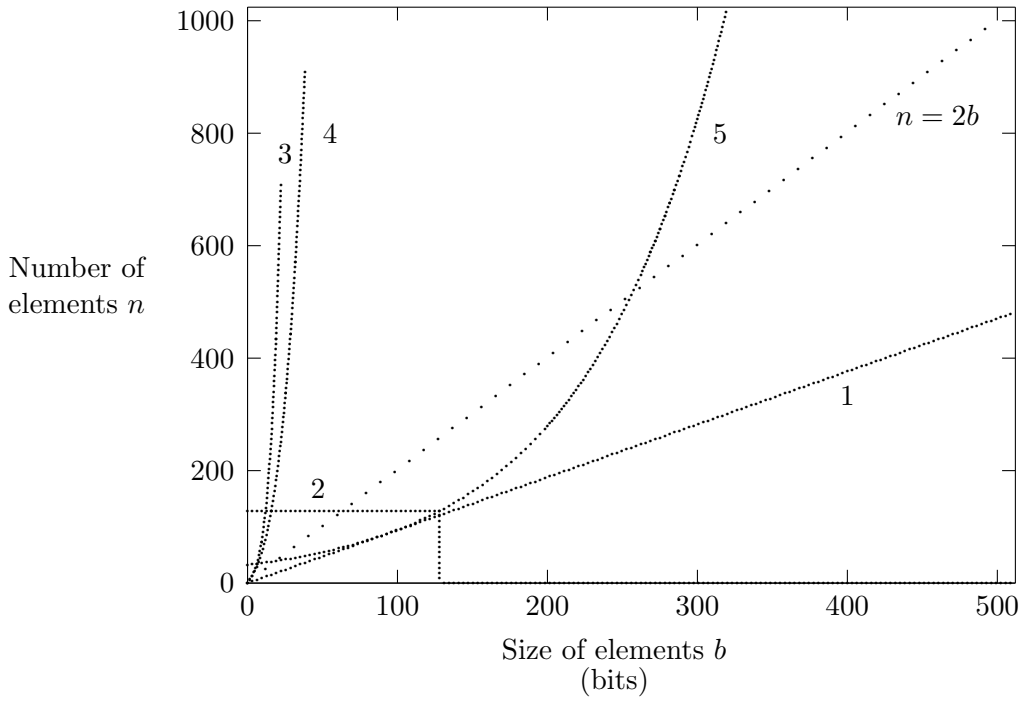

Figure 7.1: The relation between the number n of elements of the knapsack and the size b (in bits) of the elements for the 5 attacks; the numbers refer to the number of the attack.

7.2.8.3 Hash functions based on additive knapsacks

A more general knapsack was proposed by G. Gonnet. It is described in [126]. Let J be a finite set of indices, let x be a string of size $|J|$ over the alphabet X, and let T be a mapping $T : X \times J \longrightarrow [0, l[\ : (x, j) \mapsto T(x, j)$, with $l \in \mathbb{N}$. A compression function can now be defined as follows:

$$f : X^{|J|} \longrightarrow [0, s \cdot |J|\,[\ : x = (x_j)_{j \in J} \mapsto \sum_{j \in J} T(x_j, j)\,.$$

Ph. Godlewski and P. Camion [126] showed that this scheme can be broken if l is not too large by modifying an earlier attack on a scheme by J. Bosset (cf. next section). This resulted in the fourth algorithm.

I. Damgård, who was aware of the third and fourth attack by Ph. Godlewski and P. Camion, suggested that a random knapsack with $n \approx 2s$, e.g., $n = 256$ and $b = 120$ (corresponding to $s = 128$), yields a collision resistant function [66], which could be used with the construction described in section 4.3.3.2. This in turn motivated P. Camion and J. Patarin to improve again the attack on the Bosset scheme which yielded the fifth algorithm [37]. They showed that a preimage for the round function can be constructed with $O(2^{32})$ operations and storage, which is clearly feasible. Note that this only implies that one can find a pseudo-preimage and a pseudo-collision for the hash function: finding a preimage with proposition 2.2 requires still about 2^{80} operations. However, a preimage could be found directly with the Schroeppel-Shamir attack in 2^{64} operations and 2^{32} storage: in this case one has to solve a knapsack with $n = 128$ and $b = 120$. At the same time the authors of [37] pose the open problem to find a pseudo-preimage for the case $n = 512$ and $s = 256$.

J. Patarin proposed to strengthen the hash function of I. Damgård as follows [245]: every time a zero bit is encountered in the message that is to be hashed, the intermediate result is rotated to the right over one position. It is not clear whether the known attacks can be extended to this case, but it seems that a simple rotation is not sufficient to increase the security significantly.

R. Impagliazzo and M. Naor suggested to use the knapsack to construct an efficient Pseudo-random String Generator (PSG) and Universal One-Way Hash Function (UOWHF) [149]. They prove the equivalence of the security of these primitives to the one-wayness of the knapsack problem, but if the knapsack is not hard to invert on average, this is not very useful. For the PSG they suggest $n < b < 1.5n$, while for the UOWHF they recommend $n/3 < b < n$. It is clear that in the first case the extended low density attacks might cause problems, while in the second case the fifth attack by P. Camion and J. Patarin might be used to break the scheme if $b = n/3$ and $n < 768$. If b is closer to n, the security increases but the performance decreases. Indeed, if the density is approximated by n/b, the number of operations to process a single bit is proportional to

$$\frac{n}{d(1 - \frac{\log_2 n}{n}) - 1}.$$

From this equation it follows that the number of operations increases quickly if the density d approaches 1, and that for a fixed density d the number of operations per bit grows roughly linearly with n.

Ph. Godlewski and P. Camion suggested a special scheme that applies to the input a one-way function and subsequently a linear error correcting code [126]: the hashcode consists of the redundancy symbols of a MDS code [199] over $GF(p^m)$. The scheme allows to correct a limited number of errors in the image under the one-way function, but about half of the symbols can be used for detecting changes to the image and hence to the input. It has the property that at least d bits have to be modified to the

hashcode to obtain a second valid hashcode, where d is the minimum distance of the error correcting code. Because of the linearity, the attacks on the knapsack can break the scheme if either p or p^n is small.

7.2.8.4 Hash functions based on multiplicative knapsacks

The first knapsack type hash function was suggested by J. Bosset [27]. It is based on a multiplicative knapsack in $GL(2,p)$, the group of 2 by 2 invertible matrices over $GF(p)$, with $p = 10,007$. Every possible plaintext block of 6 bits is associated with a random element of the group and the result of the hash function is the product of the corresponding matrices, coded with about 80 bits.

In fact J. Bosset suggested a smaller p and hash size, namely $p = 997$ and a hash size of about 40 bits. P. Camion showed that even the larger scheme can be broken with a probabilistic factorization algorithm in $GL(2,p)$ [36]. The attack constructs a second preimage by inserting 48 correcting plaintext blocks into the second plaintext. It is based on the existence of a chain of subgroups $GL(2,p) = G_r \supseteq G_{r-1} \supseteq \ldots \supseteq G_1$ such that the indices $[G_r : G_{r-1}]$ are not too large. It was later extended to the fifth attack, which was used to break the schemes of G. Gonnet and I. Damgård (cf. previous section).

An interesting scheme by G. Zémor is also based on the hardness of finding short factorizations in certain groups [336, 337]. The general construction can be described as follows. Let $x \in X^t$ be an string of length t over the alphabet X. Let G be a group, let $\mathcal{S}$ be a set of generators with $|S| = |X|$, and let f be a bijective mapping between X and S. Then a hash function can be defined as follows:

$$h : X^m \longrightarrow G : x = (x_i)_{1 \leq i \leq t} \mapsto h(X) = f(x_1)f(x_2)\ldots f(x_t)\,.$$

The following choice is suggested: let p be a 150-bit prime number and take for G the group $SL(2,p)$ of 2×2-matrices with determinant 1 over $GF(p)$. The mapping f associates 0 with the matrix A and 1 with the matrix B, where A and B are given by:

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

The size of the hashcode is equal to $\log_2(p(p^2-1)) \approx 3\log_2 p$ bits. An implementation can be very fast, and can easily be parallelized. The security of the hash function is based on the fact that finding short factorizations in this group is hard (note that a trivial long factorization of the unity element is given by $A^p = B^p = 1$). Because of the size of p, the method by P. Camion [36] can not be applied. The properties of this hash function can be obtained by studying the Cayley graph $\mathcal{G}_p$ associated with G and S. It can be shown that this graph has a large girth (i.e., the length of the shortest cycle), namely

$$\text{girth } (\mathcal{G}_p) \geq \frac{\ln\left(\frac{p}{2}\right)}{\ln\left(\frac{1+\sqrt{5}}{2}\right)}\,.$$

This implies that two messages with the same hashcode will differ in at least 215 bits. Secondly it can be shown that this graph has a small diameter (i.e., the maximum distance between two vertices), namely $\leq 500,000 \ln p$. This shows that if the input is at least 6 Megabytes, the output ranges over the complete image. From this one would conclude that this hash function is not suitable for hashing very short messages. The large girth however implies that all messages with length ≤ 215 bits have a different image, and thus finding a preimage is still infeasible. In order to evaluate the security of the construction, one has to determine the number of operations to find a 'medium sized' factorization. Even if this construction would be insecure, other groups exist with similar properties.

7.3 Overview of MAC Proposals

In this section three schemes will be discussed extensively, namely the MAC's that are used in the anti-virus package ASP, the Message Authenticator Algorithm (MAA), and the DSA[2] algorithm (Decimal Shift and Add). Subsequently a scheme based on a stream ciphers and a scheme based on matrix algebra are described, and four proprietary algorithms are briefly mentioned.

The new contributions are the cryptanalysis of the ASP algorithms and an improved meet in the middle attack on the DSA algorithm.

7.3.1 The ASP MAC function

As discussed in section 6.3, the anti-virus package ASP [53] contains two MAC's. They are intended to generate a hashcode for every program; this hashcode is verified every time the program is loaded. In spite of the statement in [54] on p. 65, that *" The default cryptographic checksum algorithm was published in "Computers and Security", and has withstood a mild amount of attack from the broad cryptographic community,"* both algorithms that are implemented in ASP are—to the best of our knowledge—not published. Together with I. Bellefroid and K. Beyen, we have examined the algorithms and shown that they contain several weaknesses. Part of this work was also published in [16].

Both algorithms consist of an external MAC, that has the following update function:

$$f = (H_{i-1} + g(X_i)) \bmod 7295 + 2\,.$$

Here the size of the message blocks is equal to 1024 bytes, and the size of the chaining variable is only 13 bits (7295 possible values). The value for IV is obtained by applying a nonlinear transformation to the secret key K, that has only 7297 possibilities (about 13 bits), and to the name and the location of the file. The MAC is obtained by applying a similar nonlinear function to $K \oplus H_t$. It is clear that because of the size of the parameters, both an exhaustive attack for the key (cf. section 2.5.1.2) and a birthday attack (cf. section 2.5.1.3) are trivial. However, this MAC is intended for a

[2]Not to be confused with the Digital Signature Algorithm.

very restricted environment, and it is assumed that the potential attacker has no access to the secret key K or to the MAC. In this case an attacker can do no better than guess the hashcode or the key, which yields a success probability of 1/7295. This can be considered as acceptable if one only intends to thwart attacks of malicious programs like viruses. No specific weaknesses were identified in the nonlinear transformations that would improve this probability.

However, it will be shown that both algorithms can be broken even under these restricted conditions by exploiting weaknesses of g. The function g is not a memoryless mapping from 1024 bytes to 16 bits, but it is in fact an MDC with message block size of 1 byte. The chaining variable after every 1024 message bytes enters the MAC calculation, and the initial value is equal to 0. Two different versions are suggested for g.

The slowest version, that will be denoted with ASP1 has the following description:

$$g = (4 \cdot H'_{i-1}) \bmod 7295 + X[i]\,.$$

Here $X[i]$ is a message byte, that will be interpreted as an integer in two's complement, i.e., $X[i] \in [-128, 127]$. The chaining variable H'_i lies in the interval $[-128, 7421]$. The faster version will be denoted with ASP2:

$$g = (H'_{i-1} + X[i]) \oplus i\,.$$

Here the chaining variable belongs to $[0, 65535]$. It will now be shown that the following attack on ASP1 is possible: replace the first 1020 bytes of a message block, and subsequently append 4 correcting bytes such that the value $H'_{l.1024}$ and hence the input to the MAC calculation remains unchanged. A similar attack for ASP2 requires at most 60 correcting bytes.

Attack on ASP1 Let H'_{1023} be the value of the chaining variable that enters the MAC calculation before the modification, and let H'^{*}_{1019} be the value of the chaining variable after the modification. Then one simply has to solve the equation

$$H'_{1023} - (256 \cdot H'^{*}_{1019}) \bmod 7295 = 64 \cdot X[1020] + 16 \cdot X[1021] + 4 \cdot X[1022] + X[1023]\,.$$

The left hand side of the equation lies in the interval $[-7422, 7421]$, and the right hand side in $[-10880, 10795]$. It is clear that it is easy to find a solution: in most cases it is sufficient to write the left hand side in base 4. In some cases, namely if $H'_{1023} \notin [0, 7294]$, one has to work to the upper or lower boundary of the interval, and modify subsequently the coefficients. An additional weakness in this case is that the chaining variables are not uniformly distributed: the values at the border of the interval occur less frequently than expected.

Attack on ASP2 This attack is based on similar principles. The disadvantage here is that the step size is limited. This can be explained as follows: if two iterations are combined, one obtains:

$$H'_{i+1} = \big(((H'_{i-1} + X[i]) \oplus i) + X[i+1]\big) \oplus (i+1)\,.$$

If $X[i]$ and $X[i+1]$ are equal to zero, the difference in absolute value between H'_{i+1} and H'_{i-1} will always be equal to 1 if i is even. The correcting block attack will try to choose the values of $X[i]$ and $X[i+1]$ such that a larger step can be taken. It will start with an even value of i. The distance modulo 2^{16} between the two values determines whether one will go up or down towards the final value H'_{1023}. If $H'_{j+1} = H'_{1023}$, with j even and $j < 1022$, one will keep the chaining variable constant after every 2 steps by selecting $X[j+k] = 0$ and

$$X[j+k+1] = (-1)^{H'_{j+k} \bmod 2} \quad \text{for } 2 \leq k \leq 1022 - j\,.$$

A simple and conservative strategy to obtain the target value of H'_{1023} guarantees that after every 2 iterations the difference modulo 2^{16} will decrease with 128. If $X[i+1] = 0$, one obtains

$$H'_{i+1} = (H'_{i-1} + X[i]) \oplus i \oplus (i+1) = (H'_{i-1} + X[i]) \oplus \texttt{0001}_\texttt{x}\,,$$

because i is even. In the positive direction, one will choose $X[i] = 127$ if H'_{i-1} is odd, and $X[i] = 126$ if H'_{i-1} is even. In the first case the step size will be 128, and in the second case it will be 127. Note however that this can occur at most once. If the distance is smaller than 128, it is very easy to obtain the correct value in a final iteration. If the values for H'_{i-1} have to decrease, $X[i]$ will be chosen equal to -128, and a decrease of 129 per iteration will be possible (in the first step this might be only 128). This means that the total number of iterations and hence the number of correcting bytes will be at most 512.

The strategy can be improved. If a "greedy method" is used, that tries to maximize the step size after two iterations, one can show that 60 steps are sufficient. In this case the step size after two iterations will be lower bounded by 128 and upper bounded by 2174 (these numbers have to be increased by one in case of decreasing values). After at most 7 iterations it can be made equal to 2048 as follows. Assume for example that the value of H'_{974} is equal to $\texttt{4FFF}_\texttt{x}$. Choosing $X[975] = 1$ will result in a value of $H'_{975} = \texttt{5000}_\texttt{x} \oplus \texttt{03CE}_\texttt{x}$. With $X[976] = 98 = \texttt{0062}_\texttt{x}$, one obtains a value $H'_{976} = \texttt{57FF}_\texttt{x}$, which yields indeed a step size of 2048. In a similar way one can step from $\texttt{43FF}_\texttt{x}$ to $\texttt{4BFF}_\texttt{x}$, which corresponds to the same difference. When we come close, i.e., within a distance of about 1024 of the target, at most 7 steps are necessary to obtain the correct value. The maximal number of steps is hence upper bounded by $2 \cdot (7 + 16 + 7) = 60$. Depending on the value of i and of H'_i, some steps can be larger. An even better strategy will try to maximize the step size after 4 iterations. In that case the number of small steps at the beginning and at the end can be decreased. One can conclude that if the complexity of the attack is increased (i.e., more program storage and computation), the number of correcting blocks can be decreased.

An important observation is that, for a given value of H'_0, the variables H'_i can be approximated as originating from a random walk. This means that the expected distance between H'_{1024} and H'_0 is small, which implies that the distance that has to be corrected with the correcting bytes is not uniformly distributed, but tends to be small as well. As a consequence, 40 correcting bytes will be sufficient for almost all cases.

Finally note that a virus does not know whether ASP1 or ASP2 is used. However, due to the simplicity of both checksums one can find many modifications that will be undetected with both checksums. From these attacks it follows that these algorithms should be replaced by a fast MDC, like MD4 or one of its variants. A MAC is not necessary as the hashcode is not readable or writable by an attacker.

7.3.2 Message Authenticator Algorithm (MAA)

The MAA was published in 1983 by D. Davies and D. Clayden in response to a request of the UK Bankers Automated Clearing Services (BACS) [72, 74]. It was the intention that the implementation of the algorithm takes advantage of the fast 32-bit multiplication available on a mainframe computer, which is not so well suited to implement DES-like operations. In 1987 it became a part of the ISO 8731 banking standard [154]. The algorithm consists of three parts. The *prelude* is a key expansion from 64 bits to 192 bits: it generates 8 bytes for the initial value of the main loop, 8 bytes parameters of the main loop, and 8 bytes that will be used as data in the *coda*. It also eliminates weaker key bytes equal to $\mathtt{00_x}$ or $\mathtt{FF_x}$. Note that it has only to be executed in case of installation of a new key. The main loop consists of two parallel branches with logical operations, additions, and multiplications modulo $2^{32}-1$ or modulo $2^{32}-2$. The chaining variable has a length of 8 bytes and the message is processed in blocks of 4 bytes. In the coda, part of the key is introduced in the main loop, and with a modulo 2 addition the result is reduced to a 32-bit hashcode. Although the algorithm is certainly suitable for a mainframe computer, implementation in present day personal computers and workstations can also be very efficient.

In the description of the main loop, V and W are derived from the key, the chaining variables are represented with $(H1_i, H2_i)$, and the 32-bit message block is denoted with X_i. The following description is then obtained:

$$\begin{array}{ll} \textbf{Step 1} & V = \mathrm{rol}\,(V,1) \qquad K_i = V \oplus W\,. \\ \textbf{Step 2} & H1_i = \mathtt{MUL1}(H1_{i-1} \oplus X_i, S1(K_i + H2_{i-1})) \\ & H2_i = \mathtt{MUL2}(H2_{i-1} \oplus X_i, S2(K_i + H1_{i-1}))\,. \end{array}$$

Here `MUL1` is a multiplication modulo $2^{32}-1$, `MUL2` is a (modified) multiplication modulo $2^{32}-2$, and $S1$ and $S2$ are functions that fix 4 bits of the 32-bit argument to 1 and 4 bits to 0.

For a fixed key and message block, the mapping is injective: as discussed in section 2.4.1, this implies that the number of attained states decreases if variations are introduced in the beginning of a long message. Due to the fact that the size of the internal memory is equal to twice the size of the result, this effect is not very important. The designers suggest to limit the size of the messages to about 4 Megabytes, and in [154] a special mode is suggested for longer messages. Two small security problems of the algorithm have been discussed by the designers. First, factors of the modulus for multiplication might appear in the intermediate result and remain there through the rest of the computation, but this is avoided by the choice of the moduli. Secondly, if one of the arguments of the multiplication is equal to 0, it can be kept equal to

0 by choosing the subsequent message blocks equal to 0. For the second multiplier, this problem is avoided by the mappings $S1$ and $S2$. For the first multiplier, this can only happen if $X_i = H1_{i-1}$ or $X_i = H2_{i-1}$. If both chaining variables would be equal by chance, one could construct a large number of messages with hashcode equal to 0. However, as both chaining variables depend on the secret key, that attacker has no idea about their value, and hence this does not seem to help him. The only consequence of these weaknesses is that someone who knows the secret key does not need a birthday attack (which would take 2^{16} operations) to find two colliding messages. He can exploit this weakness to generate easily such a set [318].

A statistical evaluation was performed in collaboration with K. Van Espen and J. Van Mieghem [318]; it showed that the only weakness in the building blocks is the most significant bit of `MUL2`. However, this does not seem to affect the security of the algorithm. A differential attack on the algorithm might reveal some information on the chaining variables or on the secret key. The simplest input difference seems to be 2^{31}, as this avoids carries. However, extracting information is hard, as in the two last rounds 2 32-bit key blocks are used as message blocks, which means that all information is mixed with an unknown key.

7.3.3 Decimal Shift and Add algorithm (DSA)

This proposal was designed in 1980 by Sievi of the German Zentralstelle für das Chiffrierwesen, and it is used as a message authenticator for banking applications in Germany [74]. The fact that it is based on decimal representation and arithmetic makes it suitable for calculation by hand or with a pocket calculator with 10-digit precision, although present day processors still have a limited number of instructions for decimal arithmetic. The algorithm was considered as a candidate for international standardization, but was not withheld.

The key length is 20 decimal digits, which corresponds to 66.5 bits, and the data is processed in blocks of 10 digits. The operations are cyclic shifts and additions modulo 10^{10}. The data is processed in two parallel chains that each depend on 10 key digits:

$$H1_i = f_i(K_1, (X_i + H1_{i-1}) \bmod 10^{10}) \quad H2_i = f_i(K_2, (X_i + H1_{i-1}) \bmod 10^{10}) \,,$$

where the chaining variables are initialized with the corresponding key (the scheme would probably be more secure if the first chain would be initialized with the second key and vice versa), and 10 zero blocks are appended to the message. The function f can be described as follows:

$$f_i(K, Y) = (\mathrm{ror}(Y, K[i \bmod 10]) + Y) \bmod 10^{10} \,.$$

Here $\mathrm{ror}(x, a)$ is function that performs a right rotation over a positions of its argument, and $K[i]$ denotes the ith digit of K (the most significant bit has number 9). At the end the two chains are combined:

$$\begin{aligned} H'_t &= (H1_t + H2_t) \bmod 10^{10} \\ H_t &= H'_t[9{-}2] + H'_t[7{-}0] \bmod 10^8 \,, \end{aligned}$$

where $H[i-j]$ denotes the $i-j+1$-digit number consisting of digits $i, i+1, \ldots, j$ of H.

The security is certainly improved by the use of two independent parallel chains and by the reduction at the end. It is clear that the appending of the zero blocks makes it harder to observe the influence of modifications to the last message block. Some weaknesses have been discovered in [144] based on the non-uniformity of the carry vectors, and a meet in the middle attack was described that requires a storage of about 4 Terabytes. The storage requirement can be reduced by considering the following improvement.

- For a known message-MAC pair, compute $H1_t$ for all $10^{10} \approx 2^{33}$ values of K_1. Sort this array of about 37 Gigabytes.
- For a given H_t, compute the ± 100 corresponding values of H'_t; this is easy, as was shown in [144].
- Compute now $H2_t$ for all 10^{10} values of K_2, and check for all values of H'_t whether there is a value in the table such that $H'_t = (H1_t + H2_t) \bmod 10^{10}$. It is expected that about 10^{12} matches will be found; the correct key can be found after checking this key on two additional message-MAC pairs (cf. section 2.5.1.2).

Hence this attack requires 2^{40} MAC evaluations (most work has to be spent on eliminating false matches) and a storage of 37 Gigabytes. It is clear that this scheme does not offer an acceptable security level. The complexity of breaking this scheme might be reduced with differential attacks using a 1-block message: if α is added to the most significant digit of X_0, this will result in an addition of α mod 10 to the most significant digit of $H1_1$ and $H2_1$, and an addition of α to the position determined by $K_1[0]$ and $K_2[0]$ respectively. It can be expected that the 10 subsequent iterations in both chains (caused by the addition of the 0 blocks), will create a pattern in H_t that reveals some information about the key.

7.3.4 Based on a stream cipher

At Auscrypt'92, X. Lai, R. Rueppel, and J. Woollven [185] have proposed an efficient MAC based on a stream cipher. The basic idea is to use a running key sequence generated from the secret key to demultiplex a message sequence into two subsequences, which then are fed into two accumulating feedback shift registers. The MAC consists of the final state of both shift registers. The initial state of both registers can either be used as part of the secret key or can be made public. An interesting aspect of this proposal is that it is non-iterative (cf. section 2.4.1). It can be shown that the MAC has a uniform distribution if the shift registers satisfy certain conditions. For the security of the scheme, it is required that the running key sequence never repeats, which would also be required if it were to be used for secrecy protection.

One weakness of this scheme as currently proposed is the vulnerability of the end of the message: if one appends s bits to the message, one can update the corresponding MAC with a success probability of $1/2^s$: it is sufficient to guess the next s bits of the

key stream for the new message and to modify the MAC accordingly. If all appended bits are equal, one only has to predict the Hamming weight of the key stream, which is rather easy even for large s. A similar property holds if one wants to modify the last s bits of the message. These attacks can be thwarted by appending the message length and a sufficiently long 0/1 balanced string to the end of the message.

7.3.5 Matrix algebra

A scheme based on matrix algebra [138] uses as key a random $t \times t$ matrix A with t the number of b-bit blocks of the plaintext X:

$$MAC = X^t AX = \sum_{i<j} a_{ij} \cdot x_i \cdot x_j \,.$$

As the size of the input is restricted, it is in fact a keyed one-way function and not a MAC. After observing t^2 linear independent equations of this form, an opponent can solve the set for A. This makes the scheme totally insecure against a chosen message attack. If X' belongs to $\ker(A)$ then adding X' to X does not affect the result, which implies that someone who knows the secret key can easily construct messages with the same hashcode.

The dual of this scheme uses as key a $1 \times t$ random vector A and the information is stored in a $t \times t$ matrix:

$$MAC = A^t XA = \sum_{i<j} a_i \cdot a_j \cdot x_{ij} \,.$$

Computing the t entries of the key vector requires solving a quadratic set of equations ($O(t^3)$ operations), and every matrix X' with $A \in \ker(X')$ can be added to the information without affecting the hashcode.

7.3.6 Proprietary algorithms

A large number of proprietary MAC's have been proposed for specific applications. Very little information is available about these algorithms. Taking into account all the problems that have been identified in published algorithms, one can only trust them if they have been reviewed extensively by independent experts.

The Telepass algorithm [121, 135] is in fact not a hash function, but a keyed one-way function. It is used for generating MAC's in smart cards. A 64-bit result is computed from a 96-bit key, 32 bits of data, and an input E of 64 bits. It is a proprietary algorithm that requires only 200 bytes of microcode and is limited to the resources available in a smart card.

The Data Seal is a proprietary MAC [191] that was designed to protect electronic bank transfers. Another proprietary authenticator is provided by S.W.I.F.T. to the member banks to exchange financial messages [74]. No details of these algorithms are public.

In second generation mobile and cordless telephone systems, like GSM and DECT, a challenge response procedure is carried out to authenticate a portable radio termination. This procedure is based on one or two keyed one-way functions, which in case of DECT is called the DECT Standard Authentication Algorithm (DSAA) [323]. Again no details of this algorithm have been revealed.

7.4 Design principles

For the time being we are far from a theory to design efficient and secure hash functions. After having discussed eight dedicated MDC's and five MAC's, comprising some variants, it becomes clear that the best one can hope for is to extract a number of design principles from these proposals.

One of the first requirements for a hash function is that its description should be concise and clear. This will make the hash function more attractive to evaluate and to implement. Additionally one would like to have a motivation for all design decisions, and a comparison with alternatives that were rejected.

In a first part, some criteria that affect the security will be discussed, while in the second part some implementation aspects will be treated. Finally it will be explained why trapdoors can also be a problem for hash functions.

This section was partially inspired by the presentations and discussion on the Oberwolfach meeting on "Cryptographic Hash Functions," March 24-27, 1992 organised by T. Beth.

7.4.1 Security relevant criteria

The following security relevant properties will be discussed: should the round function be collision resistant and one-way or not, should the round function be partially bijective and one-way, how to protect begin and end of the hashing operation, the use of redundancy in the message, and finally a treatment of some specific problems.

Collision resistant and one-way The main issue faced by the designer is whether he will base his dedicated hash function on a function that is both collision resistant and one-way or not. Arguments in favor of a collision resistant and one-way function are that analyzing the security can be reduced to analyzing a single iteration as shown in section 4.3.3.2 for collision resistance and in section 2.4.2 for the one-way property. The advantage of this construction is also that the choice of a specific IV is not so important: the scheme should be secure with any IV. Indeed, if a perfectly secure compression function would exist, finding a pseudo-preimage or a pseudo-collision would be not easier than finding a preimage or a collision. Another advantage of this approach is that the size of the chaining variables can be increased at the cost of a decreased performance. Finally the tree construction described in section 2.4.4 can be used.

On the other hand, from analyzing the hashing process, it follows that the roles of H_i and X_i are essentially different. This means that it seems natural to impose different conditions on both inputs of the round function. E.g., if finding a pseudo-preimage is relatively easy, one can use proposition 2.2 to find a preimage faster than 2^n operations, but this does not mean that finding a preimage becomes feasible. One can hope that loosening the requirements will improve the performance of the hash function.

If the option is taken to design a collision resistant function, one should be consistent and design the round function such that it is symmetric with respect to message and chaining variable. Designs along these lines are BCA, MD2, FFT-hash, Snefru, the Damgård scheme based on a cellular automaton and the knapsack schemes. Designs that treat chaining variables and message blocks differently are N-hash, Cellhash, Subhash, MD4 and its variants, and all MAC's. On the other hand, the only round functions for which no collisions were found are those of MD2, Snefru with more than 6 passes, MD4, SHA, and RIPE-MD (MD5 is excluded). It is rather surprising, as the MD4 variants are designs of the second type.

Bijectivity and one-wayness A second related issue is the question whether the round function or part of the round function should be bijective and one-way. On the one hand, bijectivity for part of the round function seems to be a desirable property: this avoid attacks where the entropy of the internal variables or chaining variables decreases (cf. the long message attack discussed in section 2.4.1), and one can show that at certain places no collision can occur. On the other hand, one would like to have a one-way mapping: if an attacker can go backwards easily, he will be able to use meet in the middle techniques. In practice however it seems hard to combine both properties: proving that a scheme is bijective is in most cases done by proving that one can go backwards. Examples based on number theoretic problems do exist (e.g., RSA), but are are rather slow. In order to get around this, many schemes combine a bijection with a feedforward of the data input or the chaining variable input or both. Of course this destroys the bijectivity.

A distinction has to be made between schemes that treat chaining variables and message blocks symmetrically and schemes that do not have this property. In the first case BCA, FFT-hash, and Snefru are based on a bijective mapping, where the output size is reduced by exoring lower and upper halves (for BCA) and by chopping part of the bits (for FFT-hash and Snefru). In case of Snefru the round function is made one-way by exoring the input to the output. The security of the Damgård scheme based on cellular automata and of the knapsack schemes is based on the one-wayness of the round function. With respect to this criterion, MD2 behaves a bit differently: the mapping is one-way, but part of the internal state can be recovered. The optimal configuration in this case seems to be a number of bijective mappings, interleaved with feedforwards in order to avoid a meet in the middle attack. The security can also be increased by not simply selecting the output bits but applying a simple compression. It is a well known open problem to find a very efficient one-way permutation.

If message blocks and chaining variables are treated in a different way, the schemes that contain a mapping that is bijective if one of the inputs is fixed can be treated in the same way as single length hash functions based on a block cipher (cf. section 5.3.1.4). This means that all properties like fixed points can be derived from the discussion there. Cellhash and Subhash have a round function that is bijective for a fixed message and for a fixed chaining variable, but the mapping can be inverted easily if one of the inputs is fixed. It is equivalent to a CFB mode, but it derives its strength from the specific redundancy scheme in the message. MD4 and its variants are based on a function that is bijective for a fixed message block, while the one-way property is introduced by a feedforward of the previous chaining variable, hence it corresponds to scheme 5 in table 5.4. The first variant of N-hash is bijective for a fixed chaining variable, and the other one is bijective for a fixed message block. In both variants the chaining variable and the message are added modulo 2 to the output of the function. The first variant of N-hash corresponds to scheme 3 in table 5.4, and the second variant corresponds to scheme 7. From the discussion in section 5.3.1.4 it follows that fixed points can be found easily for MD4 and its variants and for N-hash and its variant. The impact of these fixed points is however limited. The conclusions on differential attacks can however not be transferred in a straightforward way: in the case of DES, a differential attack for a constant plaintext with a varying key is probably infeasible, but this is not necessarily the case for these functions. This implies that no configuration is a priori more secure against differential attacks.

For the MAC's the situation is different: MAA and DSA are one-way mappings, even for a fixed and known key, while the ASP MAC's are clearly invertible (apart from the nonlinear transformation at the end, that can however be listed in a table).

Beginning and end It is not the intention of this paragraph to repeat the extensive discussion on the initial value IV and on the padding scheme. It is recalled that it was recommended to choose a single IV or a limited set of IV's, and to use an unambiguous padding rule that comprises the addition of the length of the message. For some schemes it can be useful to impose a maximal message length.

Many schemes are vulnerable to attacks where one goes backwards or to attacks that concentrate on the end of the message like correcting block attacks. Their security can be increased without reducing the performance by adding at the end a limited number of blocks that are a function of all message blocks. Examples are the first hashcode that is used in MD2 and the Quisquater-Girault scheme of section 5.3.2.3. Other solutions are the addition of zero blocks (DSA and Subhash) or the use of cyclic redundancy in the message (Cellhash).

It was also discussed in section 2.4.1 how adding a first block that is dependent on IV can have as effect that finding a pseudo-preimage is not easier than finding a preimage: this is especially interesting in case of schemes where going backwards is very easy, and still the use of different IV's is necessary.

In case of a MAC, the security can certainly be increased if not only the round function, but also IV and the last blocks depend on the secret key K. This makes it

harder to predict the value of H_i and to obtain information on K or on H_i by selecting the last message blocks in a specific way.

Another principle that seems to increase the security is to work with two independent chains with a limited interaction in every round: in this way the security margin caused by the additional internal memory can 'wipe out' certain weaknesses in the round function.

Redundancy in the message A good design principle seems to be to use every message bit as many times as possible, and in such a way that it is hard to compensate changes in one message block. From this it follows that the same bit should be used at different locations and in different functions. This calls for a structure like MD4 where the message is used as 'key' to 'encrypt' in a reversible way the chaining variables.

Instead of duplicating every message bit, as suggested in section 5.3.1.2, the idea of using an error correcting code seems to be very promising: in this way one can guarantee that—independently of the selection of the message—the actual input to the hash function will differ in at least d locations, with d the minimum distance of the code. There are several arguments to avoid making more than a single 'pass' over the message: the performance decreases, it might be that additional storage is necessary, if the round function is invertible a generalized meet in the middle attack may be applied (cf. section 2.5.2.3), and finally in implementations one has to verify that it is indeed the same message that enters the hashing process.

Specific attacks A variety of attacks have to be considered when designing a dedicated hash function (cf. section 2.5). The first evaluation that has to be performed on an algorithm is certainly a statistical evaluation: it should be checked whether any irregularities can be found in the distribution of the output, and whether input differences affect the complete output. Special attention can be paid here to weaknesses occurring in most and least significant bits, to the relation between parity of inputs and outputs, and to other linearities.

Differential cryptanalysis has shown to be one of the most powerful techniques, especially against schemes based on S-boxes. In case of hash functions, one will look for two inputs for which the output difference is zero, or for which the output difference is equal to the input difference (if a feedforward exists of the input to the output).

Fixed points are certainly not usable in practice, but in absence of better criteria, they can be used to discriminate between several schemes.

7.4.2 Efficiency

In general a choice is made between software and hardware implementations. In software implementations, one will try to update variables sequentially, i.e., use the output of a calculation immediately in the next step. An important constraint is the limitation of the memory access, which exists at two levels. In the first place, one will try to keep as many variables as possible in the registers. Secondly, one will try to optimize the use of the cache memory. These considerations become more and more important as the

access time to the memory seems to decrease more slowly than the cycle time of the processors. This suggests that faster hash functions will rather make use of logic and arithmetic operations available on a standard processor, than of S-boxes. However, the advantage of S-boxes is that they yield a strong nonlinear relation between input and output. Other less important aspects are word size, byte ordering, and problems with carries. Finally it should be remarked that one should not try to optimize the design towards a single processor: designing and reviewing a dedicated hash function will take several years, and by that time the processor will probably be outdated. The MD4 family is clearly optimized to be fast in software: these algorithms run at several Megabit/sec on present day computers, which makes them about one order of magnitude faster than other schemes that are not yet broken.

For hardware oriented hash functions, one will try to make use of parallelism. Nonlinearity in hardware can be generated efficiently by S-boxes. The diffusion can be increased by bit permutations, that simply correspond to wire crossings. Ideally, such a dedicated hash function should also take a very limited area: this will decrease the cost of the IC, and will make it possible in the near future to integrate the hash function as a basic component in other IC's. A design that consumes too much area will very likely go the same way as most ASIC (Application Specific IC) designs for RSA during the eighties: they were never built for economical reasons. The only dedicated hardware hash function seem to be Cellhash and its variant Subhash. The expected speed is about 300 Megabit/sec with a 10 MHz clock, but the area will probably be much larger than the area for a DES implementation.

In order to limit the evaluation effort, one might ask the question whether a single hash function can be designed that would offer a good compromise between hardware and software implementation criteria. In that case, a few rules should be observed. A limited degree of parallelism is necessary in order to make hardware implementations efficient, and this will be advantageous as well on computer architectures with several processors. In order to make software implementations fast, permutations at bit level should be designed in a structured way (byte or word structure), or it should be possible to combine them together with the S-boxes, as the permutation P of DES. Moreover S-boxes seem to be a mandatory component, as they are acceptable in both hardware and software. These S-boxes should be optimal rather than large and random, and have a size of at most a few Kilobytes. Unacceptable for hardware are algorithms that use a large number of constants that do not seem to increase the security, like the 64 constants of MD5. The internal memory of the algorithm should not be too large (at most 80 bytes). The other operations should be simple arithmetic (addition) and logic operations. In this way, one can achieve a speed of 1 Mbit/sec in software and between 50 and 100 Mbit/sec in hardware (both estimates are for current technology), which is about twice as fast as the current DES implementations.

To conclude this section, table 7.3 gives an overview of the speed of some hash functions in software. All timings were performed on a 16 MHz IBM PS/2 Model 80 with a 80386 processor. The implementations were written by A. Bosselaers. Most of them use additional memory to improve the speed. The C-code was compiled with

a 32-bit compiler in protected mode. In order to allow for a comparison with the schemes of chapter 5 and 6, the speed of a software implementation of DES [8, 108], PES [181], and FEAL [225] are given, as well as the timings for a modular squaring and exponentiation with a short exponent. In this case a 512-bit modulus was chosen, and no use was made of the Chinese remainder theorem to speed up the computations. From these figures it can be derived that MDC-2 will run at about 100 Kbit/sec. Note that in some cases the C-compiler produces code that is almost optimal. The implementations of ASP1, ASP2, and QCMDCV4 (indicated with a *) are written in 16-bit code, and could be further optimized. Some algorithms like Snefru and SHA would perform relatively better on a RISC processor, where the complete internal state can be stored in the registers. On this type of processor, SHA is only about 15% slower than MD5. No results for the different versions of HAVAL have been included, as their performance will vary between the performance of MD4 and MD5.

type	hash function	C language (Kbit/sec)	Assembly language (Kbit/sec)
dedicated MAC	ASP1		2000*
	ASP2		3730*
	MAA		2750
dedicated MDC	BCA	764	764
	MD2	78	78
	MD4	2669	6273
	MD5	1849	4401
	SHA	710	1370
	RIPEMD	1334	3104
	N-hash	266	477
	FFT-hash I	212	304
	Snefru-2	970	970
	Snefru-4	520	520
	Snefru-6	358	358
	Snefru-8	270	270
	QCMDCV4		860*
block cipher	DES (+ key schedule)	130	200
	DES (fixed key)	512	660
	PES (+ key schedule)		520
	FEAL-16 (fixed key)	333	623
modular arithmetic	squaring	50	273
	exponentiation $(2^{16}+1)$	1.8	14

Table 7.3: Performance of several hash functions on an IBM PS/2 (16 MHz 80386).

7.4.3 Trapdoors in hash functions

It is very common to check proposed encryption functions for trapdoors. These are weaknesses that are built in the system on purpose, and that allow the person who knows of these weaknesses to break the algorithm much faster. This risk is certainly larger in case of proprietary algorithms, where the algorithm can only be reviewed by a limited number of persons. Although one would not expect that this issue comes up in the design of hash functions, it is clear that also here this problem arises. An obvious weakness would be the choice of the IV: one could select it in such a way that a particular plaintext block yields a fixed point.

The most obvious place to look for trapdoors are S-boxes. In order to avoid any allegations, designers tend to generate these S-boxes in a random way: the S-boxes from Snefru are derived from random numbers that were published in 1955, and the S-box from MD2 is derived from the digits of the number π. However, this solution is not completely effective: breaking the hash function might be more easy if a certain property is present. If this event has probability 10^{-9}, the designer can easily come up with 10^9 'straightforward' ways to generate the S-box from the public string, but still the 'random' permutation has the required property. In some cases, the security of the scheme might be increased if certain properties are built into the S-boxes. It is then even harder to show that no other criteria were applied.

The issue of trapdoors in the modulus of hash functions based on modular arithmetic has already been discussed in the previous chapter. Another place where it is very easy to insert a trapdoor is in a random knapsack. Here we do not think about a trapdoor that would yield a public key scheme, as most of these schemes have been broken, nor about deriving partial information about the preimage (this can be done if e.g., only one weight is odd). In this trapdoor, the person who selects the weights a_i can select the last weight as being the difference between 2 disjoint subsets of the previous weights. Hence he can take two messages of his choice (he can also reorder the weights) that hash to the same value. Although this is quite obvious, no one seems to have remarked this in relation with hash functions based on a knapsack. Here again a solution might be to generate the knapsack from a public random string.

7.5 Conclusion

In this chapter eight MDC's and five MAC's and their variants were described. Three new attacks have been described, namely on the BCA, on a variant of N-hash and on the MAC's of the ASP package. An improved meet in the middle attack on DSA was presented. Also some properties and structures of MD2 and of SHA were discussed. Other schemes were briefly described together with possible attacks. Special attention was paid to attacks on knapsack based hash functions.

Finally some design principles for hash functions were discussed, where several viewpoints have been considered: the security evaluation, the efficiency of software and hardware implementation, and the possible existence of trapdoors.

Chapter 8

Cryptographic Properties of Boolean Functions

Basic research is when I'm doing what I don't know what I'm doing. — *Werner von Braun*

8.1 Introduction

As long as no provably secure and efficient hash functions are available, dedicated hash functions will play an important role. The design and evaluation of these functions, but also of other cryptographic functions like block ciphers and stream ciphers, requires the definition of design criteria: in the absence of a 'theory' to design these functions, most criteria have been introduced by the designer to avoid a certain type of attack. More general criteria, like one-wayness of the round function, have been discussed in the previous chapter. However, a different way to look at such a function or at one of its building blocks is as a Boolean function that maps an n-bit string to an m-bit string. In the largest part of this chapter we will limit ourselves to the case $m = 1$. For convenience, we reserve the term 'Boolean function' for this case, and we will use the term 'S-box' for the more general case (rather than look-up table or LUT). A further generalization is a mapping from $GF(q^n)$ to $GF(q^m)$ [190]. In case $m = 1$, this class has been called 'Post functions'.

During the last fifteen years, cryptographic research in the area of Boolean functions has concentrated on the following topics:

1. Define criteria for Boolean functions.

2. Count Boolean functions satisfying certain criteria or combined criteria; this comprises proving that certain criteria can not be met at the same time. In some cases one is satisfied with the asymptotic behavior of the number of functions.

3. Construct Boolean functions that satisfy certain criteria. One has suggested explicit and recursive procedures. If both fail, one can try to perform an exhaustive search or an optimization procedure like simulated annealing.

4. Study the relation between the properties of the Boolean function and the security of the cryptographic function.

The theory of Boolean functions has shown to be useful in many applications. C. Shannon showed in 1938 that these functions can be used to study switching circuits [302]. In this context the goal was to obtain a mathematical description and to produce an optimal realization. In his seminal 1949 paper on cryptography, he suggested the use of an iterated substitution-permutation structure for a block cipher. The substitution would be realized by S-boxes. Later a different iterated structure with S-boxes was proposed, the Feistel cipher [104, 105]. Boolean functions also play an important role in the design of stream ciphers [287]. It was shown that Boolean functions can be used to construct error-correcting codes with interesting properties, namely the Reed-Muller codes that were published independently by D. Muller and I. Reed in 1954, but that have also been attributed to N. Mitani [199], p. 403. The study of theses codes revealed many interesting properties of Boolean functions. Another application for Boolean functions is the design of ranging codes [316].

This chapter will be restricted to cryptographic properties of Boolean functions. Only at the end some generalizations to S-boxes will be made. It will start with describing the basic definitions, and discussing several transformations on Boolean functions. Subsequently some cryptographic criteria for Boolean functions will be defined, and some general characterizations of functions that satisfy these criteria will be given. In section 8.4 the functions will be studied that satisfy several of these criteria. The largest part of this section concentrates on quadratic functions, but also some new results on bent functions will be given. An important new result is described in section 8.4.1.5: it is shown that there exists a trade-off between the static and the dynamic properties of a Boolean function. Section 8.5 describes several construction methods for Boolean functions, and section 8.6 discusses briefly the more general case of S-boxes. Section 8.7 presents the conclusions.

The main contribution of this chapter is the definition of a new criterion, the propagation criterion of degree k and order m, that generalizes the previous strict avalanche criterion (SAC) and perfect nonlinearity. A second new result is the characterization of properties of quadratic functions. Properties of the functions that satisfy the higher order SAC are studied, and all functions that satisfy SAC of maximal order are constructed. Moreover this chapter proposes a new construction for bent functions and an efficient way to count the number of bent functions of 6 variables (which was previously unknown). Part of the work in this chapter (namely parts of section 8.3.4 and section 8.4.2) was performed in collaboration with W. Van Leekwijck and L. Van Linden. A proof has been included only for propositions and theorems that are new; sometimes a trivial proof of a new result was omitted.

8.2 Definitions

8.2.1 Basic definitions

A Boolean function f is a function whose domain is the vector space $\mathbb{Z}_2^n$ of binary n-tuples $\underline{x} = [x_1, x_2, \ldots, x_n]$ and which takes the values 0 and 1. The set of all these functions will be denoted with $\mathcal{F}_n$. In some cases it will be more convenient to work with functions that take the values $\{-1, 1\}$. The function $\hat{f}$ is defined as $\hat{f}(\underline{x}) = 1 - 2 \cdot f(\underline{x})$. In the following a function will be a Boolean function unless stated explicitly.

The Hamming weight hwt of an element of $\mathbb{Z}_2^n$ is the number of components equal to 1. The Hamming weight of a function f is the number of function values equal to 1. The Hamming distance $d(f, g)$ between two functions f and g is the number of function values in which they differ:

$$d(f, g) = \mathrm{hwt}(f \oplus g) = \sum_{\underline{x}} f(\underline{x}) \oplus g(\underline{x}) = 2^{n-1} - \frac{1}{2} \sum_{\underline{x}} \hat{f}(\underline{x}) \cdot \hat{g}(\underline{x}) \,.$$

Here $\sum_{\underline{x}}$ denotes the summation over all $\underline{x} \in \mathbb{Z}_2^n$. The correlation between two functions f and g, denoted with $c(f, g)$, is closely related to the Hamming distance:

$$c(f, g) = 1 - d(f, g)/2^{n-1} \,.$$

A function is said to be linear if there exists a vector $\underline{b} \in \mathbb{Z}_2^n$ such that it can be written as $L_{\underline{b}}(\underline{x}) = \underline{x} \cdot \underline{b}$ or $\hat{L}_{\underline{b}}(\underline{x}) = (-1)^{\underline{x} \cdot \underline{b}}$. Here $\underline{x} \cdot \underline{b}$ denotes the dot product of $\underline{x}$ and $\underline{b}$, defined as $\underline{x} \cdot \underline{b} = x_1 b_1 \oplus x_2 b_2 \oplus \ldots \oplus x_n b_n$. The set of affine functions $A_{\underline{b}, b_0}(\underline{x})$ is the union of the set of the linear functions and their complement: $A_{\underline{b}, b_0}(\underline{x}) = L_{\underline{b}}(\underline{x}) \oplus b_0$, with $b_0 \in \mathbb{Z}_2$.

8.2.2 Transformations on Boolean functions

The most natural way to describe a Boolean function is to enumerate all values for its 2^n possible arguments. This representation is called the *truth table*. However, in some cases it is interesting to look at other representations of the same function or at functions that have been derived from the function. The most common representations that have been used in order to optimize the implementation are the conjunctive and disjunctive normal form [76, 294]. The representations that will be considered here are the algebraic normal form, the Walsh-Hadamard transform and the autocorrelation function.

8.2.2.1 The algebraic normal transform

The algebraic normal transform was introduced by Zhegalkin in 1927 [98]. A Boolean function is written as a sum modulo 2 of the 2^n possible products of the n inputs:

$$f(\underline{x}) = a_0 \oplus \bigoplus_{1 \le i \le n} a_i x_i \;\oplus \bigoplus_{1 \le i < j \le n} a_{ij} x_i x_j \;\oplus \ldots \oplus a_{12\ldots n} x_1 x_2 \cdots x_n \,.$$

This form is called the algebraic normal form or ringsum expansion of a Boolean function f, denoted with $\mathcal{F}$ and the corresponding transformation is called the algebraic normal transform. When the natural order of the variables is used, namely 1, x_1, x_2, x_1x_2, x_3, x_1x_3, ...,$x_1x_2 \ldots x_n$ the following definition can be stated [164]:

Definition 8.1 *Let f be a Boolean function of n variables. Then the* **algebraic normal transform** *is a linear transformation (with respect to addition modulo 2) defined as*

$$[\mathcal{F}] = A_n \cdot [f]$$

$$\textit{with} \quad A_n = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes A_{n-1} \,, \qquad A_0 = [1] \,.$$

The transform is an involution: $A_n^2 = I_n$.

Here $\otimes$ denotes the Kronecker product between matrices.

An important measure for the complexity of a Boolean function is the nonlinear order. If the order of a product of i terms is defined as i, one obtains the following definition.

Definition 8.2 *The* **nonlinear order** *of a Boolean function (notation:* $\mathrm{ord}(f)$*) is defined as the maximum of the order of its product terms that have a nonzero coefficient in the algebraic normal form.*

The constant Boolean functions have nonlinear order 0, the affine Boolean functions have nonlinear order < 2, and the quadratic functions are the functions with nonlinear order < 3.

8.2.2.2 The Walsh-Hadamard transform

The Walsh-Hadamard transform is an orthogonal transform like the discrete Fourier transform. S. Golomb was apparently the first to consider the Walsh-Hadamard transform of Boolean functions [330].

Definition 8.3 *Let f be a real-valued function with domain the vector space $\mathbb{Z}_2^n$. The* **Walsh-Hadamard transform** *of f is the real-valued function F over the vector space $\mathbb{Z}_2^n$ defined as*

$$F(\underline{w}) = \sum_{\underline{x}} f(\underline{x}) \cdot (-1)^{\underline{x} \cdot \underline{w}} \,.$$

The function f can be recovered by the **inverse Walsh-Hadamard transform:**

$$f(\underline{x}) = \frac{1}{2^n} \sum_{\underline{w}} F(\underline{w}) \cdot (-1)^{\underline{x} \cdot \underline{w}} \,.$$

It follows from this definition that the Walsh-Hadamard transform writes a Boolean function f as a sum of linear functions. The relationship between the Walsh-Hadamard transform of f and $\hat{f}$ is given by [114]

$$\hat{F}(\underline{w}) = -2F(\underline{w}) + 2^n\,\delta(\underline{w}) \quad \text{and} \quad F(\underline{w}) = -\frac{1}{2}\hat{F}(\underline{w}) + 2^{n-1}\,\delta(\underline{w})\,,$$

where $\delta(\underline{w})$ denotes the Kronecker delta ($\delta(\underline{0}) = 1, \delta(\underline{k}) = 0 \quad \forall \underline{k} \neq \underline{0}$). The Walsh-Hadamard transform $\hat{F}(\underline{w})$ of the affine function $A_{\underline{b},b_0}$ is given by $(-1)^{b_0}\,2^n\delta(\underline{w} \oplus \underline{b})$.

As the Walsh-Hadamard transform is linear, an alternative definition based on a matrix product is possible. The function values of f and F are written in the column matrices $[f]$ and $[F]$ respectively

$$[F] = H_n \cdot [f]\ ,$$

where H_n is the Walsh-Hadamard matrix of order n that can be recursively defined as

$$H_n = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes H_{n-1}\ , \qquad H_0 = [1]\ .$$

It is easily seen that $H_n^2 = 2^n \cdot I_n$.

The distance between two functions $\hat{f}$ and $\hat{g}$ can also be computed in the Walsh-Hadamard domain.

Proposition 8.1 *Let $\hat{f}$ and $\hat{g}$ be Boolean functions with Walsh-Hadamard transform $\hat{F}$ and $\hat{G}$ respectively. The Hamming distance between $\hat{f}$ and $\hat{g}$ is then given by*

$$d(f,g) = 2^{n-1} - 2^{-n-1}\sum_{\underline{w}} \hat{F}(\underline{w})\hat{G}(\underline{w})\,.$$

Parseval's theorem can now be restated a follows:

Theorem 8.1 (Parseval) *Let $\hat{f}$ be a real-valued function with domain the vector space $\mathbb{Z}_2^n$ and with Walsh-Hadamard transform $\hat{F}$. Then*

$$\sum_{\underline{w}} \hat{F}^2(\underline{w}) = 2^n \sum_{\underline{x}} \hat{f}(\underline{x})^2\,.$$

For Boolean functions this reduces to the following corollary:

Corollary 8.1

$$\sum_{\underline{w}} \hat{F}^2(\underline{w}) = 2^{2n}\,.$$

A real-valued function is Boolean iff $|\hat{f}(\underline{x})| = 1, \forall \underline{x}$. If this condition is translated to the Walsh-Hadamard domain, one obtains the following proposition of R. Titsworth.

Proposition 8.2 *Let $\hat{f}$ be a real-valued function with domain the vector space $\mathbb{Z}_2^n$ and with Walsh-Hadamard transform $\hat{F}$. Then $\hat{f}$ is* **Boolean** *iff*

$$|\hat{f}(\underline{x})| = 1 \quad \Longleftrightarrow \quad \sum_{\underline{w}} \hat{F}(\underline{w})\hat{F}(\underline{w} \oplus \underline{s}) = 2^{2n}\delta(\underline{s})\,.$$

The following proposition describes the effect of transformations on the truth table or in the Walsh-Hadamard domain. It is clear that if the function value of f is complemented, or equivalently, if $\hat{f}$ is multiplied by -1, the Walsh-Hadamard transform $\hat{F}$ will also be multiplied by -1, and vice versa. A more general transformation is the addition of an affine function to f. A second type of transformations is the set of affine transformations on the truth table or in the Walsh-Hadamard domain. The group of all affine transformations, denoted with $AGL(n)$, is the group with elements $A\underline{x} \oplus \underline{a}$, where A is a regular matrix $\in \mathbb{Z}_2^{n\times n}$ and $\underline{a}$ is a vector $\in \mathbb{Z}_2^n$. The following theorem considers the effect of the combination of an affine transformation to the input and the addition of an affine function to the function value. It generalizes results of [199], p. 417 and [255].

Proposition 8.3 *Let $\hat{f}$ and $\hat{g}$ be real-valued functions with domain the vector space $\mathbb{Z}_2^n$ and with Walsh-Hadamard transform $\hat{F}$ and $\hat{G}$ respectively.*

$$\begin{aligned}
&\textit{If} \quad \hat{g}(\underline{x}) = (-1)^{b_0}\ (-1)^{\underline{b}\cdot\underline{x}}\ \hat{f}(A\underline{x} \oplus \underline{a}) \\
&\textit{then} \quad \hat{G}(\underline{w}) = (-1)^{b_0}\ (-1)^{\underline{b}\,\cdot\, A^{-1}\underline{a}}\ (-1)^{\underline{w}\,\cdot\, A^{-1}\underline{a}}\ \hat{F}((A^{-1})^t\underline{w} \oplus (A^{-1})^t\underline{b}). \\
&\textit{If} \quad \hat{G}(\underline{w}) = (-1)^{b_0}\ (-1)^{\underline{b}\cdot\underline{w}}\ \hat{F}(A\underline{w} \oplus \underline{a}) \\
&\textit{then} \quad \hat{g}(\underline{x}) = (-1)^{b_0}\ (-1)^{\underline{b}\,\cdot\, A^{-1}\underline{a}}\ (-1)^{\underline{x}\,\cdot\, A^{-1}\underline{a}}\ \hat{f}((A^{-1})^t\underline{x} \oplus (A^{-1})^t\underline{b})\,.
\end{aligned}$$

Proof: Only the first part will be proven: the second part follows from the similarity between the Walsh-Hadamard transform and its inverse. From the definition of the Walsh-Hadamard transform it follows that

$$\begin{aligned}
\hat{G}(\underline{w}) &= \sum_{\underline{x}} \hat{g}(\underline{x}) \cdot (-1)^{\underline{x}\cdot\underline{w}} \\
&= \sum_{\underline{x}} (-1)^{b_0}\ (-1)^{\underline{b}\cdot\underline{x}}\ \hat{f}(A\underline{x} \oplus \underline{a}) \cdot (-1)^{\underline{x}\cdot\underline{w}}\,.
\end{aligned}$$

With the substitutions $\underline{w}' = \underline{w} \oplus \underline{b}$ and $\underline{x}' = A\underline{x} \oplus \underline{a}$, one obtains:

$$\hat{G}(\underline{w}' \oplus \underline{b}) = (-1)^{b_0}\ (-1)^{\underline{w}'\,\cdot\, A^{-1}\underline{a}}\ \sum_{\underline{x}'} (-1)^{\underline{w}'\,\cdot\, A^{-1}\underline{x}'}\ \hat{f}(\underline{x}')\,.$$

This can be rewritten as

$$\hat{G}(\underline{w}' \oplus \underline{b}) = (-1)^{b_0}\ (-1)^{(A^{-1})^t\underline{w}'\,\cdot\, \underline{a}}\ \sum_{\underline{x}'} (-1)^{(A^{-1})^t\underline{w}'\,\cdot\, \underline{x}'}\ \hat{f}(\underline{x}')\,.$$

Subsequently one sets $w'' = (A^{-1})^t \underline{w}'$, which results in

$$\hat{G}(A^t \underline{w}'' \oplus \underline{b}) = (-1)^{b_0} \ (-1)^{\underline{w}'' \cdot \underline{a}} \ \hat{F}(\underline{w}'') \, .$$

Finally one replaces $A^t \underline{w}'' \oplus b$ by $\underline{w}$, yielding

$$\hat{G}(\underline{w}) = (-1)^{b_0} \ (-1)^{\underline{b} \cdot A^{-1} \underline{a}} \ (-1)^{\underline{w} \cdot A^{-1} \underline{a}} \ \hat{F}((A^{-1})^t \underline{w} \oplus (A^{-1})^t \underline{b}) \, ,$$

which completes the proof. ∎

If in this proposition A is the unity matrix and $\underline{b}$ the all zero vector, one obtains the following special case.

Corollary 8.2 *A dyadic shift and a complementation in the Walsh-Hadamard domain generate all 2^{n+1} possible modifications of terms in the algebraic normal form with degree smaller than 2, or*

$$\textit{if } \ \hat{G}(\underline{w}) = (-1)^{b_0} \ \hat{F}(\underline{w} \oplus \underline{b}) \ \textit{then } \hat{g}(\underline{x}) = (-1)^{b_0} \ (-1)^{\underline{b} \cdot \underline{x}} \ \hat{f}(\underline{x}) \, .$$

It is clear from the definition of the Walsh-Hadamard transform that $\hat{F}(\underline{0})$ is equal to the sum of all function values. The following generalization was developed when studying the Hamming weight of the cosets of the first order Reed-Muller code [199], p. 415:

Proposition 8.4 *Let $\hat{f}$ be a Boolean function of n variables with Walsh-Hadamard transform $\hat{F}$. Then*

$$d(f, \underline{b} \cdot \underline{x} \oplus b_0) = \frac{1}{2} \left(2^n - (-1)^{b_0} \ \hat{F}(\underline{b}) \right) \, .$$

The last proposition of this section describes the Walsh-Hadamard transform of a Boolean function f of $n+1$ variables that is the concatenation of two functions g_1 and g_2 of n variables [114]. The algebraic normal form of f can be written as

$$f(\underline{x}_{n+1}) = (1 \oplus x_{n+1}) g_0(\underline{x}_n) \oplus x_{n+1} g_1(\underline{x}_n) = g_0(\underline{x}_n) \oplus x_{n+1}(g_0(\underline{x}_n) \oplus g_1(\underline{x}_n)) \, .$$

Here the index $_n$ of the vector $\underline{x}_n$ denotes the dimension, and $\underline{x}_{n+1} = \underline{x}_n \| x_{n+1}$. If $x_{n+1} = 0$, f is equal to g_0, and if $x_{n+1} = 1$, f is equal to g_1.

Proposition 8.5 *Let $\hat{g}_0$ and $\hat{g}_1$ be Boolean functions of n variables with Walsh-Hadamard transform $\hat{G}_0$ and $\hat{G}_1$ respectively. Then the Walsh-Hadamard transform $\hat{F}$ of the concatenation of $\hat{g}_0$ and $\hat{g}_1$ is given by*

$$\hat{F}(\underline{w}_{n+1}) = \hat{G}_0(\underline{w}_n) + (-1)^{w_{n+1}} \ \hat{G}_1(\underline{w}_n) \, .$$

Concerning the terminology 'Walsh-Hadamard transform', it is important to note that in fact four different transformations were defined. The Hadamard transform [14] is the transformation that has been described here, and the Walsh transform is the same transformation but with reversed order for the bits of $\underline{w}$. As both transformations are often confounded, we suggest the name Walsh-Hadamard transform.

8.2.2.3 The autocorrelation function

The autocorrelation function is a well known concept in signal processing. It is also a very useful tool to study cryptographic properties.

Definition 8.4 *Let $\hat{f}$ be a real-valued function with domain the vector space $\mathbb{Z}_2^n$. The* **autocorrelation function** *of $\hat{f}$ is the real-valued function over the vector space $\mathbb{Z}_2^n$ defined as*

$$\hat{r}_{\hat{f}}(\underline{s}) = \sum_{\underline{x}} \hat{f}(\underline{x}) \cdot \hat{f}(\underline{x} \oplus \underline{s}) \,.$$

If it is allowed by the context, the index $_{\hat{f}}$ will be omitted. If $\hat{f}$ is Boolean, the autocorrelation function of f can be defined in a similar way:

$$r_f(\underline{s}) = \sum_{\underline{x}} f(\underline{x}) \oplus f(\underline{x} \oplus \underline{s}) \,.$$

From this definition it follows that $\hat{r}_{\hat{f}} = 2^n - r_f$. Note that $\hat{r}(\underline{0}) = 2^n$ and $r(\underline{0}) = 0$. The value of the autocorrelation function in $\underline{s}$ is proportional to the correlation between $\hat{f}(\underline{x})$ and $\hat{f}(\underline{x} \oplus \underline{s})$. The autocorrelation function of the affine function $A_{\underline{b},b_0}$ is given by $2^n(-1)^{\underline{s}\cdot\underline{b}}$.

A more general concept is the crosscorrelation between two functions:

Definition 8.5 *Let $\hat{f}$ and $\hat{g}$ be real-valued functions with domain the vector space $\mathbb{Z}_2^n$. The* **crosscorrelation function** *of $\hat{f}$ and $\hat{g}$ is the real-valued function over the vector space $\mathbb{Z}_2^n$ defined as*

$$\hat{c}_{\hat{f},\hat{g}}(\underline{s}) = \sum_{\underline{x}} \hat{f}(\underline{x}) \cdot \hat{g}(\underline{x} \oplus \underline{s}) \,.$$

It is clear that $\hat{c}_{\hat{f},\hat{g}} = \hat{c}_{\hat{g},\hat{f}}$. If $\hat{f}$ and $\hat{g}$ are Boolean functions, one can define the crosscorrelation function $c_{f,g}$ in a similar way. It follows from this definition that the autocorrelation function is the crosscorrelation of a function with itself or $\hat{r}_{\hat{f}} = \hat{c}_{\hat{f},\hat{f}}$.

One of the most important theorems for all Fourier-type transformations is the convolution theorem. It states that the transform of the convolution is the product of the transforms. In case of the Walsh-Hadamard transform, one obtains the following formulation.

Theorem 8.2 (convolution) *Let $\hat{f}$ and $\hat{g}$ be real-valued functions with domain the vector space $\mathbb{Z}_2^n$ and with Walsh-Hadamard transform $\hat{F}$ and $\hat{G}$ respectively. Then the Walsh-Hadamard transform of the crosscorrelation of $\hat{f}$ and $\hat{g}$ is equal to the pointwise product of $\hat{F}$ and $\hat{G}$, or*

$$\sum_{\underline{x}} \hat{c}_{\hat{f},\hat{g}}(\underline{x})(-1)^{\underline{x}\cdot\underline{w}} = \hat{F}(\underline{w}) \cdot \hat{G}(\underline{w}), \quad \forall \underline{w} \in \mathbb{Z}_2^n \,.$$

If $\hat{f} = \hat{g}$, one obtains the Walsh-Hadamard variant of the well known *Wiener-Khintchine theorem*, that in case of the Fourier transform states that the Fourier

transform of the autocorrelation function is equal to the energy spectrum (the square modulus of the Fourier transform). The Walsh-Hadamard energy spectrum of any real-valued function f is defined as $\hat{F}^2$. The Walsh-Hadamard transform of $\hat{r}$ will be denoted with $\hat{R}$.

Theorem 8.3 (Wiener-Khintchine) *Let $\hat{f}$ be a real-valued function with domain the vector space $\mathbb{Z}_2^n$ and with Walsh-Hadamard transform $\hat{F}$. Then the Walsh-Hadamard transform $\hat{R}$ of the autocorrelation function of $\hat{f}$ is equal to the Walsh-Hadamard energy spectrum of $\hat{f}$:*

$$\hat{R}(\underline{w}) = \hat{F}^2(\underline{w}), \quad \forall \underline{w} \in \mathbb{Z}_2^n \,.$$

This theorem is very useful to obtain a spectral characterization of certain cryptographic properties.

The effect on the autocorrelation function of transformations on f can be derived from the combination of the Wiener-Khintchine theorem with proposition 8.3.

Proposition 8.6 *Let $\hat{f}$ and $\hat{g}$ be real-valued functions with domain the vector space $\mathbb{Z}_2^n$ and with autocorrelation function $\hat{r}_{\hat{f}}$ and $\hat{r}_{\hat{g}}$ respectively.*

$$\textit{If } \hat{g}(\underline{x}) = (-1)^{b_0}\ (-1)^{\underline{b}\cdot\underline{x}}\ \hat{f}(A\underline{x} \oplus \underline{a}) \textit{ then } \quad \hat{r}_{\hat{g}}(\underline{s}) = (-1)^{\underline{s}\,\cdot\,(A^{-1})^t\underline{b}}\ \hat{r}_{\hat{f}}(A\underline{s}) \,.$$

This implies that a dyadic shift of the function values and complementing the function do not affect the autocorrelation function, while the addition of a linear function does not modify the absolute value of the autocorrelation function.

This section is concluded with a description of the autocorrelation function of a Boolean function f of $n+1$ variables that is the concatenation of two functions g_1 and g_2 of n variables.

Proposition 8.7 *Let $\hat{g}_0$ and $\hat{g}_1$ be Boolean functions of n variables with autocorrelation functions $\hat{r}_{\hat{g}_0}$ and $\hat{r}_{\hat{g}_1}$ respectively, and with crosscorrelation $\hat{c}_{\hat{g}_0,\hat{g}_1}$.*
Then the autocorrelation function $\hat{r}_{\hat{f}}(\underline{s}_{n+1})$ of the concatenation of $\hat{g}_0$ and $\hat{g}_1$ is given by

$$\begin{array}{ll} \hat{r}_{\hat{g}_0}(\underline{s}_n) + \hat{r}_{\hat{g}_1}(\underline{s}_n) & \textit{if } s_{n+1} = 0 \\ 2 \cdot \hat{c}_{\hat{g}_0,\hat{g}_1}(\underline{s}_n) & \textit{if } s_{n+1} = 1 \,. \end{array}$$

Proof: From proposition 8.5 it follows that the Walsh-Hadamard transform of $\hat{f}$ is given by

$$\hat{F}(\underline{w}_{n+1}) = \hat{G}_0(\underline{w}_n) + (-1)^{w_{n+1}}\ \hat{G}_1(\underline{w}_n) \,,$$

and hence the energy spectrum is equal to

$$\hat{F}(\underline{w}_{n+1})^2 = \hat{G}_0(\underline{w}_n)^2 + \hat{G}_1(\underline{w}_n)^2 + 2(-1)^{w_{n+1}}\ \hat{G}_0(\underline{w}_n) \cdot \hat{G}_1(\underline{w}_n) \,.$$

The Wiener-Khintchine theorem states that the autocorrelation function of $\hat{f}$ can be obtained by taking the inverse Walsh-Hadamard transform, or

$$\hat{r}_{\hat{f}}(\underline{s}_{n+1}) = \frac{1}{2^{n+1}} \left[\sum_{\underline{w}_{n+1}} (-1)^{\underline{s}_{n+1} \cdot \underline{w}_{n+1}} \left(\hat{G}_0(\underline{w}_n)^2 + \hat{G}_1(\underline{w}_n)^2 + 2(-1)^{w_{n+1}} \hat{G}_0(\underline{w}_n) \cdot \hat{G}_1(\underline{w}_n) \right) \right].$$

If $s_{n+1} = 0$, the first two terms clearly yield $\hat{r}_{\hat{g}_0}(\underline{s}) + \hat{r}_{\hat{g}_1}(\underline{s}_n)$, and the second term vanishes. In case $s_{n+1} = 1$, the first two terms sum to 0, and the third term yields the crosscorrelation of $\hat{g}_0$ and $\hat{g}_1$ (by the convolution theorem). ∎

8.3 Criteria for Boolean functions and their properties

A design criterion D_n is a valuation, i.e., a mapping from $\mathcal{F}_n$ to a set W, and a function f is considered to be 'good' if the value $D_n(f)$ belongs to some well defined subset W' of W. In this section the following criteria will be defined: completeness, nonlinearity, balancedness and correlation immunity, and the propagation criterion of degree k.

In cryptography, it is important that properties still hold after applying simple or weak transformations to the function. For transformations applied to the argument of the function, one has the following definition [208].

Definition 8.6 *Let $\Omega(n)$ denote the group of all invertible transformations of $GF(2^n)$, and let D_n be a design criterion. The* **symmetry group of** *D_n, denoted with $I(D_n)$, is the largest subgroup of $\Omega(n)$ which leaves D_n invariant, or*

$$I(D_n) = \{\alpha \in \Omega(n) \mid D_n(\alpha(f(\underline{x}))) = D_n(f(\underline{x})),\ \forall \underline{x} \in \mathbb{Z}_2^n,\ \forall f \in \mathcal{F}_n\}\ .$$

We will encounter three types of criteria. For the first type, the symmetry group is equal to the group of all permutations and complementations of variables, while for the second type it is equal to $GL(n)$, the subgroup of all linear transformations, and for third type it is equal to $AGL(n)$, the subgroup of all affine transformations.

One can also consider modifications to the algebraic normal form of the function. An important case is the addition of an affine function: some criteria are not affected by such an addition, while others clearly are. Another modification that might be applied is a small change to the truth table. A design criterion D_n is called **robust** if small changes to the truth table do not lead to large changes of D_n. This concept is related to stability as defined in [97].

8.3.1 Completeness

One of the oldest criteria suggested for a Boolean function is that it should be complete or nondegenerate [176]. This means that the output should depend on all input bits. It is clear that this is a very weak criterion, as it does not specify how strong the dependency should be. If a function is not complete, the truth table will show specific

symmetries. However, it will be much easier to identify this in the algebraic normal form: if a function does not depend on input i, the variable x_i will simply not appear. In the autocorrelation function this can be detected easily: if $\underline{e_i}$ is the ith unit vector, i.e., the vector with a one in position i and zeroes elsewhere, $\hat{r}(\underline{e_i})$ will be equal to 2^n. With the aid of the Wiener-Khintchine theorem, this can be restated in terms of the Walsh-Hadamard transform.

8.3.2 Nonlinearity

Nonlinearity is a desirable property in cryptography, as linear systems are known to be very weak. In this section four definitions for nonlinearity will be considered, namely the nonlinear order, the distance to affine functions, the distance to rth order functions, and the distance to linear structures.

A natural choice for a nonlinearity criterion seems to be $\mathrm{ord}(f)$. The symmetry group of this criterion is $AGL(n)$ [208], and of course the nonlinear order is not affected if terms of lower order are added. However, the criterion is not robust: a small modification to the truth table can cause a big difference. The function $A_{\underline{b},b_0}(\underline{x}) + x_1 x_2 \cdots x_n$ satisfies $\mathrm{ord}(f) = n$, but its truth table differs only in a single location from an affine function.

The following proposition establishes a relation between the nonlinear order of a Boolean function and its Hamming weight. It is a corollary of a deep theorem by R. McEliece ([199], p. 447):

Proposition 8.8 *Let f be a Boolean function of n variables with* $\mathrm{ord}(f) = r$ *($r > 0$). Then* $\mathrm{hwt}(f)$ *is a multiple of*

$$2^{\lceil \frac{n}{r} \rceil - 1}\,.$$

The truth table, Walsh-Hadamard spectrum, and autocorrelation function of affine functions are recognized easily. For functions of maximal nonlinear order, one knows that their Hamming weight is odd, and one can also prove certain properties of the autocorrelation function and the Walsh-Hadamard spectrum.

Proposition 8.9 *Let f be a Boolean function of n variables with $n > 2$.*
If $\mathrm{ord}(f) = n$*, the autocorrelation function $\hat{r}_f$ can have no zeroes.*

Proof: It is sufficient to show that a zero in the autocorrelation function implies that $\mathrm{hwt}(f)$ is even, which yields a contradiction. Let $\underline{s}$ be the value for which $\hat{r}(\underline{s}) = 0$ or $r(\underline{s}) = 2^{n-1}$. Then

$$\begin{aligned}
\mathrm{hwt}(f) &= \sum_{\underline{x}} f(\underline{x}) \mod 2 \\
&= \sum_{\underline{x}} f(\underline{x} \oplus \underline{s}) \mod 2 \\
&= \frac{1}{2} \sum_{\underline{x}} f(\underline{x}) \oplus f(\underline{x} \oplus \underline{s}) \mod 2 \\
&= \frac{1}{2} r(\underline{s}) \bmod 2 = 2^{n-2} \bmod 2\,.
\end{aligned}$$

If $n > 2$, the theorem follows. ∎

The following proposition is due to R. Titsworth. It follows directly from proposition 8.4.

Proposition 8.10 *Let f be a Boolean function of n variables with $n > 1$.*
If $\mathrm{ord}(f) = n$, the Walsh-Hadamard transform $\hat{F}$ can have no zeroes.

A more general relation between the nonlinear order of a Boolean function and its Walsh-Hadamard spectrum was obtained by Ph. Langevin [186].

Proposition 8.11 *Let f be a Boolean function of n variables. If 2^s is a divisor of $\hat{F}(\underline{w})$, $\forall \underline{w}$, then $\mathrm{ord}(f) \leq n - s + 1$, and if additionally 2^{s+1} (with $s \geq 2$) is no divisor of any $\hat{F}(\underline{w})$, then $\mathrm{ord}(f) \leq n - s$.*

A more robust measure of nonlinearity is the distance of a Boolean function to the set of affine functions [208].

Definition 8.7 *Let f be a Boolean function of n variables. The* **distance to the set of affine functions** *of f is defined as*

$$\min_{\underline{b},b_0} d(f, A_{\underline{b},b_0}) .$$

It can be shown that this distance is upper bounded by

$$2^{n-1} - 2^{\lfloor n/2 \rfloor - 1} .$$

This upper bound can only be met if n is even, and the functions that achieve this bound are the bent functions (cf. section 8.3.4 and 8.4.2). The symmetry group of this criterion is $AGL(n)$ [208], and it is not affected if an affine function is added to f.

The distance to affine functions can be obtained easily from the Walsh-Hadamard spectrum. The alternative formula for the distance

$$d(f, g) = 2^{n-1} - 2^{-n-1} \sum_{\underline{w}} \hat{F}(\underline{w}) \hat{G}(\underline{w})$$

shows that the minimum distance to affine functions equals

$$2^{n-1} - \frac{1}{2} \max_{\underline{w}} |\, \hat{F}(\underline{w}) \,| \ .$$

The maximal value of the spectrum is thus a measure for the minimum distance to affine functions. The fact that the Walsh-Hadamard transform can be used to obtain the best affine approximation of a Boolean function was observed by R. Rueppel [286].

A robust generalization with the same symmetry group is the distance to functions with nonlinear order $\leq k$ [208]. Determining the best rth order approximation for $r \geq 2$ is not easy for large n, as it is equivalent to decoding the rth order Reed-Muller codes. A practical algorithm to find such an approximation has been described in [206].

This method assumes that the distance in a number of subsets of the truth table is not too large.

Another robust definition of nonlinearity is the distance to linear structures [208]. Its symmetry group is also $AGL(n)$ [208]. A linear structure for a function is a value of $\underline{s} \neq \underline{0}$ such that $f(\underline{x}) \oplus f(\underline{x} \oplus \underline{s})$ is constant or such that $|\hat{r}(\underline{s})| = 2^n$. The distance to linear structures can then be defined as the shortest distance to the set of functions with a linear structure. It can be calculated as follows:

$$2^{n-2} - \frac{1}{4} \max_{\underline{s} \neq \underline{0}} |\hat{r}(\underline{s})| = 2^{n-2} - \frac{1}{2} \max_{\underline{s} \neq \underline{0}} |r(\underline{s}) - 2^{n-1}| \ .$$

It was shown in [48] that linear structures of S-boxes can lead to weaknesses in Feistel-type substitution ciphers. If f has a linear structure, there is a linear transformation that maps f onto a function that is linear in some of its input bits [241].

8.3.3 Balancedness and correlation immunity

For cryptographic applications, it is often very important that the truth table has as many 0 as 1 entries. In that case, the uncertainty over the value of f or the entropy $H(f)$ is maximal (the definition of the entropy and related concepts was given in section 3.2.2).

Definition 8.8 *Let f be a Boolean function of n variables.*
Then f is **balanced** *iff* $\mathrm{hwt}\, f = 2^{n-1}$.

A necessary and sufficient condition for a function to be balanced is that $\hat{F}(\underline{0}) = 0$, which follows directly from the definition of the Walsh-Hadamard transform. A necessary condition is that $\mathrm{ord}(f) < n$ (cf. proposition 8.10).

A related concept, suggested by T. Siegenthaler [307], is correlation immunity. It originated from analyzing the security of certain stream ciphers [308].

Definition 8.9 *Let f be a Boolean function of n variables. Then f is m***th order correlation immune***, $CI(m)$ ($1 \leq m \leq n$), iff $\hat{f}(\underline{x})$ is statistically independent of any subset of m input variables.*

G. Xiao and J. Massey [330] showed that this condition is equivalent to the condition that $f(\underline{x})$ is statistically independent of any linear combination of at most m input variables. A function satisfies $CI(m)$ iff knowledge of m input variables gives no *additional* information on the output, or iff the mutual information between any m input variables $i_1, i_2, \ldots i_m$ and the output is equal to 0:

$$H(f(\underline{x})) - H(f(\underline{x}) \mid x_{i_1}, x_{i_2}, \ldots, x_{i_m}) = 0 \,.$$

Note that if f is constant, knowledge of any number of input bits gives no additional information about the function value, and hence the function satisfies $CI(n)$. In [38] it was shown that the truth table of a correlation immune function corresponds to an orthogonal array.

T. Siegenthaler derived conditions on the algebraic normal form. These conditions were strengthened by G. Xiao and J. Massey [330]:

Proposition 8.12 *Let f be a Boolean function of n variables. If f satisfies $CI(m)$, then* $\mathrm{ord}(f) \leq n - m$, *and all coefficients of order $n - m$ are equal to $2^{-n}\hat{F}(\underline{0})$.*

The property that a function is balanced and satisfies $CI(m)$ will be abbreviated with $CIB(m)$, while the nonbalanced functions that satisfy $CI(m)$ will be said to satisfy $CIN(m)$. In the special case that f satisfies $CIB(m)$, one obtains the following result:

Corollary 8.3 *Let f be a Boolean function of n variables that satisfies $CIB(m)$:*

$$\begin{aligned} &\text{if } 0 < m < n-1 \quad \text{then } \mathrm{ord}(f) \leq n - m - 1, \\ &\text{if } m = n-1 \qquad\quad \text{then } f = \bigoplus_{i=1}^{n} x_i \oplus \epsilon \text{ with } \epsilon = 0 \text{ or } 1\,. \end{aligned}$$

A characterization in the Walsh-Hadamard domain was given by G. Xiao and J. Massey [330], and their proof was later simplified by L. Brynielsson [287].

Proposition 8.13 *Let f be a Boolean function of n variables. Then f satisfies $CI(m)$ iff*

$$\hat{F}(\underline{w}) = 0 \quad 1 \leq \mathrm{hwt}(\underline{w}) \leq m\,.$$

From the distance formula based on the Walsh-Hadamard spectrum it follows that a function f satisfies $CI(m)$ iff its correlation to linear functions of the form $(-1)^{\underline{b}\cdot\underline{x}}$, with $1 \leq \mathrm{hwt} \leq \underline{b}$ is equal to zero. From this standpoint, Parseval's theorem can be restated as follows:

$$\sum_{\underline{b}} c^2(f, L_{\underline{b}}) = 1\,,$$

or the sum of the squares of the correlation to all affine functions is equal to 1. This implies that if the correlation to a subset of affine functions is smaller, the correlation to other affine functions will increase. For the largest values of m, one can note that $CI(n-1)$ is only satisfied by a linear function, namely the sum of all variables (and its complement), and $CI(n)$ (correlation immunity of maximal order) is only satisfied by a constant function. Note that balancedness implies that the correlation to a constant function is zero.

A slightly different interpretation of correlation immunity is the following: a function f satisfies $CIB(m)$ iff it is balanced, and every function obtained from f by fixing $1 \leq m' \leq m$ input bits is still balanced (independently of the value that is given to these bits). For functions that satisfy $CIN(m)$, i.e., functions that are not balanced, one obtains the following interpretation. The *unbalance* of the function is defined as $\hat{F}(\underline{0})$ (for a balanced function the unbalance would be equal to 0). In this case a function satisfies $CIN(m)$ iff every function obtained from f by fixing $1 \leq m' \leq m$ input bits has the minimum possible unbalance, independently of the value that is given to these bits. The unbalance can be reduced with a factor of at most 2 if a single input bit is fixed. Hence f satisfies $CI(m)$ iff fixing m' input bits of f, with $1 \leq m' \leq m$ results in an unbalance of $\hat{F}(\underline{0})/2^{m'}$, independently of the choice of these input bits. For $m = 1$, the functions obtained from $\hat{f}$ by setting x_n to 0 and 1 will be denoted

with $\hat{g}_0$ and $\hat{g}_1$ respectively. Proposition 8.5 yields an expression for the unbalance of $\hat{g}_0$ and $\hat{g}_1$:

$$\hat{G}_0(\underline{0}_{n-1}) = \frac{1}{2}\left(\hat{F}(\underline{0}_{n-1}0) + \hat{F}(\underline{0}_{n-1}1)\right)$$
$$\hat{G}_1(\underline{0}_{n-1}) = \frac{1}{2}\left(\hat{F}(\underline{0}_{n-1}0) - \hat{F}(\underline{0}_{n-1}1)\right) .$$

It is clear that the unbalance will be halved in both cases iff $\hat{F}(\underline{0}_{n-1}1) = 0$. This argument can be repeated when variable x_i is fixed, yielding $\hat{F}(\underline{e_i}) = 0$, $\forall i$ with $1 \leq i \leq n$, which corresponds the case $m = 1$ in proposition 8.13. A generalization can be found in [115, 320].

From proposition 8.3 it follows that the symmetry group of the correlation immunity criterion is the group of all permutations and complementations of variables, and that the criterion is not affected by complementing the function either. On the other hand, the addition of a linear function corresponds to a dyadic shift in the Walsh-Hadamard domain (cf. corollary 8.2), and hence it can modify the value of m. The criterion is not robust, but a small modification to the truth table will only result in a small increase in the correlation. If a single entry is modified in the truth table, every value of the Walsh-Hadamard spectrum $\hat{F}$ is modified with the value ± 2. For example, the all zero function satisfies $CI(n)$, and the function $x_1 x_2 \cdots x_n$ (the product of all variables) has the following Walsh-Hadamard spectrum:

$$\hat{F}(\underline{0}) = 2^n - 2 \text{ and } \quad \hat{F}(\underline{w}) = -2 \cdot (-1)^{\mathrm{hwt}\,(\underline{w})} \ \forall\, \underline{w} \neq \underline{0}\,.$$

Hence the correlation to all linear functions remains small. The distance of the function to a balanced function has as symmetry group all linear transformations, or $GL(n)$. If an affine function is added, the behavior is similar to the behavior for correlation immunity.

A criterion which has a larger symmetry group, namely $AGL(n)$, is the number of zeroes of $\hat{F}$, denoted with $N_{\hat{F}}$. It is not affected either by the addition of an affine function. However, if one takes into account Parseval's theorem, and considers the distance to affine functions, one can conclude that for many applications it might be better to have a flat Walsh-Hadamard energy spectrum. The only functions with this property are the bent functions, and they will be discussed in section 8.3.4 and 8.4.2.

Note finally that the concept of correlation immunity was generalized by R. Forré in [115], by considering the entropy profile, i.e., the values of $H(f(\underline{x}))$ and $H(f(\underline{x}) \mid x_{i_1}, x_{i_2}, \ldots, x_{i_m})$ for $m = 1$ through $n-1$, and for all $\binom{n}{m}$ choices of i_1, i_2, $\ldots$ i_m. For a good function, the entropy profile has to be as large as possible and should be smooth.

8.3.4 Propagation criteria

The propagation criteria study the dynamic behavior of a Boolean function, i.e., what happens if the input to the function is modified.

Definition 8.10 *Let f be a Boolean function of n variables. Then f satisfies the* **propagation criterion of degree** *k, $PC(k)$ ($1 \leq k \leq n$), if $\hat{f}(\underline{x})$ changes with a probability of $1/2$ whenever i ($1 \leq i \leq k$) bits of $\underline{x}$ are complemented.*

This criterion generalizes two previous criteria. The strict avalanche criterion or SAC that was introduced by A. Webster and S. Tavares in [324] is equivalent to $PC(1)$, and perfect nonlinearity, that was proposed by W. Meier and O. Staffelbach in [208] is equivalent to $PC(n)$. Later the same definition was given independently in [3]. If a function that satisfies $PC(1)$ is approximated by a function of less variables, it will agree for at most 75% of the values [11, 114]. The approximation by a function of less variables seems to be an interesting open problem, that is related to lower order approximations. More work certainly has to be done on this subject.

The relation between the probability that the output changes and the autocorrelation function can be obtained directly from the definition of the autocorrelation function:

Proposition 8.14 *Let f be a Boolean function of n variables.*

$$\Pr\left(\hat{f}(\underline{x}) \neq \hat{f}(\underline{x} \oplus \underline{s})\right) = \frac{1}{2} - \frac{\hat{r}(\underline{s})}{2^{n+1}}\,.$$

Summing these probabilities for all $\underline{s} \neq \underline{0}$ yields:

$$\sum_{\underline{s} \neq \underline{0}} \Pr\left(\hat{f}(\underline{x}) \neq \hat{f}(\underline{x} \oplus \underline{s})\right) = 2^{n-1} - \frac{\hat{F}^2(\underline{0})}{2^{n+1}}\,.$$

If f is balanced then $\hat{F}(\underline{0}) = 0$ and the average of the probabilities is $2^{n-1}/(2^n - 1) > \frac{1}{2}$.

It is now straightforward to restate $PC(k)$ in terms of the autocorrelation function, which yields the dual of proposition 8.13.

Proposition 8.15 *Let f be a Boolean function of n variables. Then f satisfies $PC(k)$ iff*

$$\hat{r}(\underline{s}) = 0 \;\; \textit{for} \;\; 1 \leq \mathrm{hwt}(\underline{s}) \leq k\,.$$

The expression of $PC(1)$ under the form of a sum of a function with a shifted version of itself has been used in several papers (e.g., [114, 208]), but no connection was made to the concept of an autocorrelation function. On the one hand, this concept clearly shows the similarity to correlation immunity, and on the other hand it results in the possibility to use the Wiener-Khintchine theorem to calculate $\hat{r}$ efficiently and to prove several theorems easily. An information theoretic interpretation of $PC(k)$ is that the mutual information between $f(\underline{x})$ and $f(\underline{x} \oplus \underline{s})$ is zero, or

$$H(f(\underline{x} \oplus \underline{s})) - H(f(\underline{x} \oplus \underline{s}) \mid f(\underline{x})) = 0 \;\text{ for }\; 1 \leq \mathrm{hwt}(\underline{s}) \leq k\,.$$

The relation between $PC(k)$ and the nonlinear order is more complex than in the case of $CI(m)$. From proposition 8.9 it follows that if $\hat{r}$ has only a single zero, the

function can not have maximal linear order. The functions that satisfy $PC(n)$ are well known as the perfect nonlinear or bent functions. They were discovered by O. Rothaus in the sixties, but his results were only published in 1976 [285]. Bent functions have been used in spread spectrum techniques [243]. In 1989 they were rediscovered by W. Meier and O. Staffelbach when they were studying perfect nonlinear functions. Here the properties of bent functions will be briefly summarized. Some new results will be given in section 8.4.2. Bent functions only exist for even values of n. The nonlinear order of bent functions is bounded from above by $\frac{n}{2}$ for $n > 2$. Their autocorrelation function is an impulse, and hence their energy spectrum is flat with value 2^n. This implies that bent functions are never balanced and the difference between the number of ones and the number of zeroes equals $\pm 2^{\frac{n}{2}}$. In [208] it is shown that bent functions have maximum distance 2^{n-2} to all linear structures and maximum distance $2^{n-1} - 2^{\frac{n}{2}-1}$ to all affine functions, and thus they have minimum correlation $\pm 2^{-\frac{n}{2}}$ to all affine functions. For odd values of n, there exist functions that satisfy $PC(n-1)$, and their nonlinear order is upper bounded by $(n+1)/2$. Note that if Post functions from $\mathbb{Z}_q^n$ to $\mathbb{Z}_q$ are considered, a perfect nonlinear function is always bent (i.e., has a flat spectrum), but a bent function is only a perfect nonlinear function if q is prime [237].

Based on the Wiener-Khintchine theorem, the $PC(k)$ criterion can be restated as follows:

$$\sum_{\underline{w}} (-1)^{\underline{s}\cdot\underline{w}} \hat{F}^2(\underline{w}) = 0 \text{ for } 1 \leq \text{hwt}(\underline{s}) \leq k\,.$$

This generalizes the characterization by R. Forré in [114] for the case $k = 1$, and yields a simple proof.

In the same paper R. Forré proves that the following condition is sufficient for $PC(1)$:

$$\hat{F}^2(\underline{w}) = \hat{F}^2(\underline{w} \oplus \underline{v})\,,$$

where $\underline{v}$ denotes the all 1 vector $[11\cdots1]$. It is however not a necessary condition: with the aid of proposition 8.3 this can be reduced to

$$\hat{r}(\underline{s}) = (-1)^{\underline{s}\cdot\underline{v}}\, \hat{r}(\underline{s})\,,$$

which is equivalent to

$$\hat{r}(\underline{s}) = 0 \text{ if hwt}(\underline{s}) \text{ is odd!}$$

This condition is clearly much stronger than $PC(1)$.

A different way to look at $PC(k)$ is based on the directional derivative.

Definition 8.11 *Let f be a Boolean function of n variables. The* **directional derivative** *of f in direction $\underline{s}$ is defined as:*

$$d_{f,\underline{s}}(\underline{x}) = f(\underline{x}) \oplus f(\underline{x} \oplus \underline{s})\,.$$

The relation between the directional derivative and the autocorrelation function is given by:

$$r(\underline{s}) = \sum_{\underline{x}} d_{f,\underline{s}}(\underline{x})\,.$$

One now obtains an alternative definition: f satisfies $PC(k)$ iff the directional derivative $\hat{d}_{f,\underline{s}}(\underline{x})$ of f is balanced $\forall\ \underline{s} : 1 \leq \mathrm{hwt}(\underline{s}) \leq k$. This concept will be more important if further generalizations of $PC(k)$ are studied.

From proposition 8.6 it follows that the symmetry group of $PC(k)$ is the group of all permutations and complementations of variables (note that for $PC(1)$ this was already proven in [114]). Moreover, it also implies that the criterion is not affected by the addition of an affine function. It was already shown in [208] that perfect nonlinearity or $PC(n)$ has as symmetry group the affine group $AGL(n)$. None of these criteria is robust, but a small modification to the truth table will only result in a small modification to the autocorrelation function. Indeed, if a single entry of the truth table is modified, the autocorrelation function will change for every nonzero argument with ± 4. Another criterion that has as symmetry group $AGL(n)$ would be the number of zeroes of $\hat{r}(\underline{s})$, denoted with $N_{\hat{r}}$.

Note that the concept of $PC(k)$ could be generalized in the same way as $CI(m)$, by defining the autocorrelation profile, i.e., the values of $\hat{r}(\underline{s})$ ordered according to increasing Hamming weight of $\underline{s}$. For a good function, the autocorrelation profile has to come as close as possible to the impulse function in the origin, which is the autocorrelation function of a bent function.

It was shown that $CI(m)$ can also be explained in terms of properties of functions that are obtained when input bits of f are fixed. The motivation for this approach is that in certain types of attacks the cryptanalyst fixes part of the input, and tries to attack the remaining function. A natural question that arises is how $PC(k)$ behaves with respect to this criterion, i.e., which properties are satisfied by functions that are obtained from f by fixing input bits? It will be shown that if one imposes that these functions also satisfy $PC(k)$, a stronger criterion is obtained, that will be denoted with $PC(k)$ of higher order. The first higher order propagation criterion was suggested by R. Forré [114] for $PC(1)$. In this section new and generalized definitions will be proposed, that have been described and studied in [255, 320]. Both generalizations are straightforward, but in the first case an additional restriction is imposed: it is required that $k + m \leq n$ or if m bits are kept constant, at most $n - m$ bits can be changed.

Definition 8.12 *Let f be a Boolean function of n variables. Then f satisfies the* **propagation criterion of degree k and order m** *($PC(k)$ of order m) iff any function obtained from f by keeping m input bits constant satisfies $PC(k)$.*

A more general definition is obtained if the restriction $k + m \leq n$ is removed. This means that a certain value is given to m bits and subsequently k bits are changed. However, the set of bits that are given a certain value and the set of those that are changed can have common elements. This leads to an information theoretic definition, which is comparable to correlation immunity.

Definition 8.13 *Let f be a Boolean function of n variables. Then f satisfies the* **extended propagation criterion of degree k and order m** *($EPC(k)$ of order m) iff knowledge of m bits of $\underline{x}$ gives no information on $f(\underline{x}) \oplus f(\underline{x} \oplus \underline{s})$, $\forall\ \underline{s}$ with $1 \leq \mathrm{hwt}(\underline{s}) \leq k$.*

The definition can be restated in terms of balancedness and correlation immunity of the directional derivative.

Proposition 8.16 *Let f be a Boolean function of n variables.*

1. *f satisfies $EPC(k)$ of order 0 iff the directional derivative $\hat{d}_{f,\underline{s}}(\underline{x})$ of f is balanced $\forall\ \underline{s} : 1 \leq \mathrm{hwt}(\underline{s}) \leq k$.*
2. *f satisfies $EPC(k)$ of order $m > 0$ iff the directional derivative $\hat{d}_{f,\underline{s}}(\underline{x})$ of f satisfies $CIB(m)$, $\forall\ \underline{s} : 1 \leq \mathrm{hwt}(\underline{s}) \leq k$.*

Both definitions only make sense if one can show that if the criterion holds for order m, it also holds for $0 \leq m' \leq m$. This was done for $PC(1)$ of order m in [194], and the proof can be adapted for the more general criteria. It can be shown that the higher order criteria have the same symmetry group and behavior under the addition of an affine function as the original criteria.

The relation between PC and EPC is given in the following proposition:

Proposition 8.17 *Let f be a Boolean function of n variables.*

1. *If f satisfies $EPC(k)$ of order m (with $k \leq n - m$) then f satisfies $PC(k)$ of order m.*
2. *If f satisfies $PC(k)$ of order 0 or 1 then f satisfies $EPC(k)$ of order 0 or 1.*
3. *If f satisfies $PC(1)$ of order m then f satisfies $EPC(1)$ of order m.*

Proof: The first part follows directly from the definition of $PC(k)$ and $EPC(k)$, and the same holds for the second part if the order is equal to 0. The proof of the second part can be completed as follows. Let f be a function that satisfies $PC(k)$ of order 1. Now f satisfies $EPC(k)$ of order 1 iff

$$\sum_{\underline{x},\, x_i = b_0} \hat{f}(\underline{x}) \cdot \hat{f}(\underline{x} \oplus \underline{a}) = 0\,,$$

$\forall i,\ 1 \leq i \leq n$, $\forall b_0 \in \mathbb{Z}_2$, and $\forall \underline{a}$ with $1 \leq \mathrm{hwt}(\underline{a}) \leq k$. If $a_i = 0$, this condition is equivalent to $PC(k)$ of order 1. If $a_i = 1$, the expression can be reduced to

$$\frac{1}{2} \sum_{\underline{x}} \hat{f}(\underline{x}) \cdot \hat{f}(\underline{x} \oplus \underline{a}) = 0\,,$$

and this is equivalent to $PC(k)$ of order 0.
The proof of the third part is based on the same principles. Let f be a function that satisfies $PC(1)$ of order m. Now f satisfies $EPC(1)$ of order m iff

$$\sum_{\underline{x},\, x_{j_1} = b_{j_1} \,\ldots\, x_{j_m} = b_{j_m}} \hat{f}(\underline{x}) \cdot \hat{f}(\underline{x} \oplus \underline{e_i}) = 0\,,$$

$\forall i$, $1 \leq i \leq n$, $\forall$ choices of m positions $j_1, \ldots j_m$ out of n, and $\forall b_{j_1} \ldots b_{j_m} \in \mathbb{Z}_2$. If $j_k \neq i$, $\forall k$, with $1 \leq k \leq m$, this condition is exactly the condition for $PC(k)$ of order m. If there exists a k such that $i = j_k$, the expression can be reduced to

$$\frac{1}{2} \sum_{\underline{x},\, x_{j_1}=b_{j_1} \ldots x_{j_{k-1}}=b_{j_{k-1}} x_{j_{k+1}}=b_{j_{k+1}} \ldots x_{j_m}=b_{j_m}} \hat{f}(\underline{x}) \cdot \hat{f}(\underline{x} \oplus \underline{e_i}) = 0\,.$$

This equality is satisfied as it is equivalent to the condition that f satisfies $PC(1)$ of order $m-1$. ∎

In section 8.4.1.4 it will be shown that this theorem is tight: there exist functions that satisfy $PC(2)$ of order 2, but do not satisfy $EPC(2)$ of order 2.

It follows directly from the definition of $PC(1)$ that the maximal order is equal to $n-2$. A new result for $PC(1)$ of higher order is an upper bound on the nonlinear order.

Theorem 8.4 *Let f be a Boolean function of n variables, with $n > 2$.*

1. *If f satisfies $PC(1)$ of order m $(0 \leq m < n-2)$, then $2 \leq \mathrm{ord}(f) \leq n-m-1$.*
2. *If f satisfies $PC(1)$ of order $n-2$, then $\mathrm{ord}(f) = 2$.*

Proof: The lower bound on $\mathrm{ord}(f)$ follows from the fact that a linear function does not satisfy $PC(1)$ of order 0. In order to prove the upper bound of the first part, it will be shown that if $0 \leq m < n-2$, f can have no terms of order $\geq n-m$. Assume that f has a term of order $n-k$, with $0 \leq k \leq m$. By setting all k variables that do not occur in this term to zero, a function f' of $n-k$ variables is obtained with $\mathrm{ord}(f') = n-k$. From proposition 8.9 it follows that if $n-k > 2$, the autocorrelation function of f' can have no zeroes, and hence f' can not satisfy $PC(k)$.
The upper bound of the second part follows from the fact that a function that satisfies $PC(n-2)$ also satisfies $PC(n-3)$. ∎

The resulting upper bound is indicated in table 8.1. In section 8.4.1.4 a character-

m	0	1	...	$n-4$	$n-3$	$n-2$
nonlinear order $\leq$	$n-1$	$n-2$	...	3	2	2

Table 8.1: Upper bound on nonlinear order of functions satisfying $PC(1)$ of order m.

ization will be given of second order functions satisfying $PC(1)$ of higher order.

This section is concluded with the spectral characterization of functions satisfying $PC(k)$ and $EPC(k)$ of order m, which generalizes the case $PC(1)$ of order m as given in [114].

Proposition 8.18 *Let $\hat{f}$ be a Boolean function of n variables with Walsh-Hadamard transform $\hat{F}$.*

Then $\hat{f}$ satisfies $PC(k)$ of order m iff

$$\sum_{\underline{w}} \hat{F}(\underline{w}) \cdot \hat{F}(\underline{w} \oplus \underline{a}) \cdot (-1)^{\underline{w} \cdot \underline{s}} = 0 \,,$$

$\forall \underline{a}$ with $1 \leq \mathrm{hwt}(\underline{a}) \leq m$, and $\forall \underline{s}$ with $1 \leq \mathrm{hwt}(\underline{s}) \leq k$, with $a_i s_i = 0$, $\forall i$ with $1 \leq i \leq n$. Then $\hat{f}$ satisfies $EPC(k)$ of order m iff

$$\sum_{\underline{w}} \hat{F}(\underline{w}) \cdot \hat{F}(\underline{w} \oplus \underline{a}) \cdot (-1)^{\underline{w} \cdot \underline{s}} = 0 \,,$$

$\forall \underline{a}$ with $1 \leq \mathrm{hwt}(\underline{a}) \leq m$, and $\forall \underline{s}$ with $1 \leq \mathrm{hwt}(\underline{s}) \leq k$.

Proof: The proof for $k = 1$ and $m = 1$ was given in [114]. Here we will prove the result only for $k = 2$, $m = 2$, since the general case is very tedious. Assume that input i and j of $\hat{f}$ are made equal to zero, resulting in the function $\hat{f}'$. By applying proposition 8.5 twice, one finds the following expression for $\hat{F}'$:

$$\hat{F}'(\underline{w}') = \frac{1}{4} \left[\hat{F}(\underline{w}') \oplus \hat{F}(\underline{w}' \oplus \underline{e_i}) \oplus \hat{F}(\underline{w}' \oplus \underline{e_j}) \oplus \hat{F}(\underline{w}' \oplus \underline{e_{ij}}) \right] ,$$

where $\underline{e_{ij}}$ denotes the vector with a one in positions i and j and zeroes elsewhere. Now f' has to satisfy $EPC(2)$, which can be expressed with the Wiener-Khintchine theorem:

$$\sum_{\underline{w}'} (-1)^{s_k w'_k \oplus s_l w'_l} \, \hat{F}'(\underline{w}')^2 = 0 \,,$$

$\forall k, l$ with $1 \leq k \leq l \leq n$. In case of $PC(2)$, one imposes the condition that k and l are different from both i and j. By substituting the expression for $\hat{F}'(\underline{w}')$, and regrouping the terms such that the summation over $\underline{w}'$ can be rewritten as a summation over $\underline{w}$, one obtains four terms:

$$\begin{aligned}
&\frac{1}{16} \sum_{\underline{w}} (-1)^{s_k w_k \oplus s_l w_l} \, \hat{F}^2(\underline{w}) \\
&+ \frac{1}{8} \sum_{\underline{w}} (-1)^{s_k w_k \oplus s_l w_l} \, \hat{F}(\underline{w}) \hat{F}(\underline{w} \oplus \underline{e_i}) + \frac{1}{8} \sum_{\underline{w}} (-1)^{s_k w_k \oplus s_l w_l} \, \hat{F}(\underline{w}) \hat{F}(\underline{w} \oplus \underline{e_j}) \\
&+ \frac{1}{8} \sum_{\underline{w}} (-1)^{s_k w_k \oplus s_l w_l} \, \hat{F}(\underline{w}) \hat{F}(\underline{w} \oplus \underline{e_{ij}}) \,.
\end{aligned}$$

The first three terms are zero as f satisfies also $PC(k)$ of order 0 and 1. Hence the fourth term must also vanish. The proof is then completed by observing that the assignment of a different value to input bits i and j does not lead to new conditions on $\hat{F}$. ∎

8.4 Functions satisfying certain criteria

This section will describe functions that satisfy the criteria defined in the previous section. The emphasis will be on quadratic functions, since several new results have been obtained for these functions. It is very hard to extend these results to functions with higher nonlinear order. Some work on third order functions has been done in relation to Reed-Muller codes, but many problems are still unsolved. Subsequently two new constructions for bent functions will be described, one of which has shown to be equivalent to a previous construction, an efficient method will be proposed to count the number of bent functions of 6 variables, and some remarks will be made on the extension of the concept of bent functions for odd n.

8.4.1 Quadratic functions

First a canonical form for a quadratic function will be described. This canonical form will then be used to derive the results on $PC(k)$ and on $CI(m)$. Subsequently the quadratic functions that satisfy higher order propagation criteria are studied, and the section is concluded with an observation on combined properties of the autocorrelation function and the Walsh-Hadamard spectrum.

8.4.1.1 A canonical form

For the study of quadratic functions, it is useful to write the second order coefficients of the algebraic normal form in a binary upper triangular matrix A with zero diagonal. Hence

$$f(\underline{x}) = \underline{x}A\underline{x}^t \oplus \underline{b} \cdot \underline{x} \oplus b_0 \,.$$

One now defines the matrix B as follows: $B = A + A^t$. B is a symmetric matrix with zero diagonal, and is called a **symplectic matrix**. The relation between the quadratic function f and the matrix B is that the associated symplectic form of the function can be written as

$$\underline{x}B\underline{y}^t = f(\underline{x} \oplus \underline{y}) \oplus f(\underline{x}) \oplus f(\underline{y}) \oplus f(\underline{0}) \,.$$

The number of second order coefficients equals $\frac{n\cdot(n-1)}{2}$ and will be denoted with $\Delta(n)$. The quadratic functions can be reduced with an equivalence transform to a canonical form based on Dickson's theorem [199], p. 438.

Theorem 8.5 (Dickson's theorem) *If B is a symplectic $n \times n$ matrix of rank $2h$, then there exists an invertible binary matrix R such that RBR^T has zeroes everywhere except on the two diagonals immediately above and below the main diagonal, and there has $1010\ldots100\ldots0$ with h ones.*
Every quadratic Boolean function can by an affine transformation of variables be reduced to $\bigoplus_{i=1}^{h} x_{2i-1}x_{2i} \oplus \epsilon$, with ϵ an affine function of x_{2h+1} through x_n.

The rank of the $2^{\Delta(n)}$ symplectic matrices is given by the following theorem [199], p. 436:

Lemma 8.1 *The number of symplectic $n \times n$ matrices over $\mathbb{Z}_2$ of rank $2h$ equals*

$$M(n, 2h) = \begin{bmatrix} n \\ 2h \end{bmatrix} \cdot M(2h, 2h)\,.$$

Here $\begin{bmatrix} n \\ k \end{bmatrix}$ denotes the binary Gaussian binomial coefficient, defined for all nonnegative integers k by

$$\begin{bmatrix} n \\ 0 \end{bmatrix} = 1, \quad \begin{bmatrix} n \\ k \end{bmatrix} = \begin{bmatrix} n-1 \\ k-1 \end{bmatrix} + 2^k \begin{bmatrix} n-1 \\ k \end{bmatrix}$$

and $M(2h, 2h) = (2^{2h-1} - 1)2^{2h-2} \cdots (2^3 - 1)2^2$.

8.4.1.2 Quadratic functions satisfying $PC(k)$

In this section the number of quadratic Boolean functions of n variables that satisfy $PC(k)$ will be studied. From section 8.3.4 it follows that this criterion is not affected by the addition of affine terms, and hence abstraction will be made of linear and constant terms. The number of quadratic Boolean functions of n variables that satisfy $PC(k)$ and that do not satisfy $PC(k+1)$ divided by 2^{n+1} will be denoted with $|\,PC(k)^2_n\,|$. Note that it follows from the definition of $PC(k)$ that the number of quadratic functions satisfying $PC(k)$ is equal to $2^{n+1} \cdot \sum_{l=k}^{n} |\,PC(l)^2_n\,|$. The values $|\,PC(k)^2_n\,|$ for small n are given in table 8.2. In this section, the numbers in this table will be explained and partially generalized for arbitrary n.

k	$\lvert PC(k)^2_2 \rvert$	$\lvert PC(k)^2_3 \rvert$	$\lvert PC(k)^2_4 \rvert$	$\lvert PC(k)^2_5 \rvert$	$\lvert PC(k)^2_6 \rvert$	$\lvert PC(k)^2_7 \rvert$	$\lvert PC(k)^2_8 \rvert$
1	0	3	13	320	8661	467432	45272585
2	1	1	0	280	4480	530180	57911392
3	–	0	0	140	420	486920	32568200
4	–	–	28	28	0	291648	3888640
5	–	–	–	0	0	97216	0
6	–	–	–	–	13888	13888	0
7	–	–	–	–	–	0	0
8	–	–	–	–	–	–	112881664

Table 8.2: The number of quadratic functions of n variables satisfying $PC(k)$ and not $PC(k+1)$ divided by 2^{n+1} ($|\,PC(k)^2_n\,|$) for $n = 2$ to 8.

For n even the quadratic bent functions or functions satisfying $PC(n)$ correspond to the symplectic matrices of full rank, and their number $|\,PC(n)^2_n\,| = M(n, n)$ [199], p. 442. This result was found independently by the author and in [97]. If n is odd, no functions exist that satisfy $PC(n)$, but we obtained a construction for all $|\,PC(n-1)^2_n\,| = M(n-1, n-1)$ functions satisfying $PC(n-1)$. It will be described in section 8.4.1.4. The generalization of these results for arbitrary k is nontrivial, because $PC(k)$ is not invariant under $AGL(n)$ (cf. section 8.3.4), and hence one can

not make use of Dickson's theorem to obtain directly the number of quadratic functions satisfying $PC(k)$. Therefore we first study a number that is invariant under affine transformations, namely the number of zeroes of the autocorrelation function (denoted by $N_{\hat{r}}$). Subsequently these results can be applied to $PC(k)$.

Theorem 8.6 *The autocorrelation function of a quadratic function takes the values* 0 *and* $\pm 2^n$. *The number of zeroes is given by* $N_{\hat{r}} = 2^n - 2^{n-2h}$ *for* $1 \leq h \leq \lfloor \frac{n}{2} \rfloor$.
The number of functions with this number of zeroes equals $M(n, 2h)$. *There are* $\left[{n \atop 2h} \right]$ *possible patterns for these zeroes and to every pattern correspond exactly* $M(2h, 2h)$ *functions.*
The coordinates where $\hat{r}(\underline{s}) \neq 0$ *form a* $(n-2h)$*-dimensional subspace of* $\mathbb{Z}_2^n$.

Proof: For the canonical quadratic function:

$$f(\underline{x}) = \bigoplus_{i=1}^{h} x_{2i-1} x_{2i} \,,$$

the autocorrelation function can be written as follows:

$$r(\underline{s}) = \sum_{\underline{x}} \left(\bigoplus_{i=1}^{h} s_{2i-1} s_{2i} \;\oplus\; \bigoplus_{i=1}^{h} (x_{2i-1} s_i \oplus x_{2i} s_{2i-1}) \right) .$$

Note that the affine function ϵ can be omitted because affine terms have no influence on the autocorrelation function. If $r(\underline{s}) = 2^{n-1}$, corresponding to $\hat{r}(\underline{s}) = 0$, the first part of the sum has no influence on the result. It is easily seen that $r(\underline{s})$ will be equal to 2^{n-1} if there exists at least one $s_i \neq 0$, with $1 \leq i \leq 2h$. Hence the number of nonzeroes of $\hat{r}(\underline{s})$ equals 2^{n-2h}, corresponding to the vectors $\underline{s}$ with $s_i = 0$, for $1 \leq i \leq 2h$. It is clear that these vectors form a subspace of $\mathbb{Z}_2^n$ of dimension $n - 2h$. The number of distinct subspaces of dimension $n - 2h$ corresponds to the number of $[n, n-2h]$ codes and equals $\left[{n \atop 2h} \right]$ [199], p. 444. ∎

The last part of theorem 8.6 makes it possible in principle to compute the number of quadratic functions satisfying $PC(k)$.

Corollary 8.4 *The number of quadratic functions of* n *variables satisfying* $PC(k)$ *is given by*

$$\sum_{h=1}^{\lfloor \frac{n}{2} \rfloor} L(n, n-2h, k) M(2h, 2h) \,,$$

where $L(n, r, k)$ *denotes the number of linear* $[n, r, d]$ *codes with minimum distance* $d > k$.

Proof: This follows from the observation that a function will satisfy $PC(k)$ if the nonzeroes of r, that form a subspace of dimension $n - 2h$, occur at positions with

Hamming weight $> k$. This is equivalent to the statement that the nonzeroes should form a linear code with length n, dimension $n - 2h$ and minimum distance $d > k$. ∎

Because of the Singleton bound ($d \leq n - r + 1$) [199] p. 33, the lower limit of this sum can be increased to $\lfloor \frac{k+1}{2} \rfloor$. However, the computation of $L(n, r, k)$ even for small values of n is a difficult problem. Even the maximal d for given n and r (notation $d_{max}(n, r)$) remains an open problem except for $r \leq 5$ and $d \leq 3$. In [145] a table of known bounds $d_{max}(n, r)$ is listed for $n \leq 127$. Table 8.3 gives, for n between 2 and 8, the number of linear codes with a given dimension and minimum distance. From the last column it can be verified that the number of distinct (n, r, d) codes is equal to $\begin{bmatrix} n \\ r \end{bmatrix}$.

In order to clarify how corollary 8.4 together with table 8.3 can be used to explain the numbers in table 8.2, a complete example is given for $n = 6$. In this case, $1 \leq h \leq 3$, and the autocorrelation function has 1, 4, or 16 nonzeroes and the number of corresponding functions is 13888 (bent functions), 18228, and 651. For 4 nonzeroes, $r = 2$ and from table 8.3 one finds $d_{max}(6, 2) = 4$. Hence the bent functions are the only functions satisfying $PC(4)$, $PC(5)$ and $PC(6)$. The number of $[6, 2, 4]$, $[6, 2, 3]$, $[6, 2, 2]$, and $[6, 2, 1]$ codes is 15, 160, 305, and 171 respectively. With every code correspond 28 functions, resulting in 420, 4480, 8540, and 4788 functions for every class. For 16 nonzeroes, every code corresponds to exactly one function. The number of $[6, 4, 2]$ and $[6, 4, 1]$ codes is given by 121 and 530. The number of 6-bit functions satisfying $PC(3)$ equals $13888+420 = 14308$, the number of functions satisfying $PC(2)$ equals $14308 + 4480 = 18788$, and the number of functions satisfying $PC(1)$ equals $18788 + 8540 + 121 = 27449$.

Corollary 8.4 can be combined with the results in [145] to determine an expression for the number of quadratic functions satisfying $PC(k)$ for large values of k.

Corollary 8.5 *The number of quadratic functions of n variables satisfying $PC(k)$ is given by*

$$\begin{array}{lll} M(n,n), & \lfloor \frac{2n}{3} \rfloor \leq k \leq n & n \text{ even,} \\ M(n-1,n-1) \cdot \sum_{i=k+1}^{n} \binom{n}{i}, & \lfloor \frac{4n}{7} \rfloor \leq k \leq n-1 & n \text{ odd and } n \not\equiv 2 \bmod 7, \\ & \lfloor \frac{4n}{7} \rfloor - 1 \leq k \leq n-1 & n \text{ odd and } n \equiv 2 \bmod 7. \end{array}$$

Proof: For n even this follows from $d_{max}(n, 2) = \lfloor \frac{2n}{3} \rfloor$ [145]. For n odd this follows from the fact that the number of $[n, 1, d]$ codes equals $\binom{n}{d}$ and that $d_{max}(n, 3) = \lfloor \frac{4n}{7} \rfloor - 1$ if $n \equiv 2 \bmod 7$ and $d_{max}(n, 3) = \lfloor \frac{4n}{7} \rfloor$ else. ∎

8.4.1.3 Quadratic functions satisfying $CI(m)$

In this section the number of quadratic Boolean functions of n variables that are correlation immune and balanced will be studied. The situation here is more complex than in the case of $PC(k)$, as these criteria are affected by the addition of linear terms (cf. section 8.3.3). For the two constant functions 0 and 1, it is clear that they satisfy $CIN(n)$. The linear functions $\underline{b} \cdot \underline{x} \oplus b_0$ satisfy $CIB(\text{hwt}(\underline{b}) - 1)$.

r \ d	0	1	2	tot
0	1	0	0	1
1	0	2	1	3
2	0	1	0	1

r \ d	0	1	2	3	tot
0	1	0	0	0	1
1	0	3	3	1	7
2	0	6	1	0	7
3	0	1	0	0	1

r \ d	0	1	2	3	4	tot
0	1	0	0	0	0	1
1	0	4	6	4	1	15
2	0	22	13	0	0	35
3	0	14	1	0	0	15
4	0	1	0	0	0	1

r \ d	0	1	2	3	4	5	tot
0	1	0	0	0	0	0	1
1	0	5	10	10	5	1	31
2	0	65	75	15	0	0	155
3	0	115	40	0	0	0	155
4	0	30	1	0	0	0	31
5	0	1	0	0	0	0	1

r \ d	0	1	2	3	4	5	6	tot
0	1	0	0	0	0	0	0	1
1	0	6	15	20	15	6	1	63
2	0	171	305	160	15	0	0	651
3	0	725	640	30	0	0	0	1395
4	0	530	121	0	0	0	0	651
5	0	62	1	0	0	0	0	63
6	0	1	0	0	0	0	0	1

r \ d	0	1	2	3	4	5	6	7	tot
0	1	0	0	0	0	0	0	0	1
1	0	7	21	35	35	21	7	1	127
2	0	420	1022	945	280	0	0	0	2667
3	0	3941	6265	1575	30	0	0	0	11811
4	0	7000	4781	30	0	0	0	0	11811
5	0	2303	364	0	0	0	0	0	2667
6	0	126	1	0	0	0	0	0	127
7	0	1	0	0	0	0	0	0	1

r \ d	0	1	2	3	4	5	6	7	8	tot
0	1	0	0	0	0	0	0	0	0	1
1	0	8	28	56	70	56	28	8	1	255
2	0	988	3038	4144	2345	280	0	0	0	10795
3	0	19628	46522	28560	2445	0	0	0	0	97155
4	0	77926	109991	12840	30	0	0	0	0	200787
5	0	63114	34041	0	0	0	0	0	0	97155
6	0	9702	1093	0	0	0	0	0	0	10795
7	0	254	1	0	0	0	0	0	0	255
8	0	1	0	0	0	0	0	0	0	1

Table 8.3: The number of linear codes with a given dimension r and minimum distance d for n between 2 and 8.

The number of quadratic Boolean functions of n variables that satisfy $CIB(m)$ and not $CIB(m+1)$ will be denoted with $|CIB(m)_n^2|$, and for the number of nonbalanced functions with the same property the symbol $|CIN(m)_n^2|$ will be used. These numbers comprise the affine functions, and will always be even, as complementing the function does not affect $CI(m)$. Note that it follows from the definition of correlation immunity that the total number of functions that satisfy $CIB(m)$ is given by $\sum_{l=m}^{n} |CIB(l)_n^2|$. For nonbalanced functions a similar expression holds. The values of $|CIB(m)_n^2|$ and $|CIN(m)_n^2|$ for small n are given in table 8.4. In this section, some of the numbers in this table will be explained and generalized for arbitrary n.

m	$\|CIB(m)_2^2\|$	$\|CIB(m)_3^2\|$	$\|CIB(m)_4^2\|$	$\|CIB(m)_5^2\|$	$\|CIB(m)_6^2\|$	$\|CIB(m)_7^2\|$
0	4	62	648	32346	1506524	271866518
1	2	6	212	3620	306686	26871250
2	0	2	8	540	13760	1722686
3	–	0	2	10	1150	40950
4	–	–	0	2	12	2170
5	–	–	–	0	2	14
6	–	–	–	–	0	2
7	–	–	–	–	–	0
tot	6	70	870	36518	1828134	300503590

m	$\|CIN(m)_2^2\|$	$\|CIN(m)_3^2\|$	$\|CIN(m)_4^2\|$	$\|CIN(m)_5^2\|$	$\|CIN(m)_6^2\|$	$\|CIN(m)_7^2\|$
0	8	48	1072	27400	2253912	230261024
1	0	8	104	1496	110856	6069616
2	2	0	0	120	1280	34440
3	–	2	0	0	120	2240
4	–	–	2	0	0	0
5	–	–	–	2	0	0
6	–	–	–	–	2	0
7	–	–	–	–	–	2
tot	10	58	1178	29018	2366170	236367322

Table 8.4: The number of balanced and nonbalanced quadratic functions of n variables that are mth order correlation immune and not $m+1$th order correlation immune ($|CIB(m)_n^2|$ and $|CIN(m)_n^2|$) for $n = 2$ to 7.

The number of balanced functions is already known, as the weight distribution of functions with ord ≤ 2 has been obtained from the study of Reed-Muller codes [199], p. 443.

Theorem 8.7 *Let A_i be the number of functions with* ord ≤ 2 *and Hamming weight i. Then $A_i = 0$ unless $i = 2^{n-1}$ or $i = 2^{n-1} \pm 2^{n-1-h}$ for some h, $0 \leq h \leq \lfloor \frac{n}{2} \rfloor$. Also*

$A_0 = A_{2^n} = 1$ *and*

$$A_{2^{n-1} \pm 2^{n-1-h}} = 2^{h(h+1)} \cdot \frac{(2^n - 1)(2^{n-1} - 1) \cdots (2^{n-2h+1} - 1)}{(2^{2h} - 1)(2^{2h-2} - 1) \cdots (2^2 - 1)} \quad \textit{for } 1 \leq h \leq \lfloor \frac{n}{2} \rfloor .$$

$A_{2^{n-1}}$ *can be evaluated because all* A_i *sum to* $2^{1+n+\Delta(n)}$.

For large values of m the functions that satisfy $CIN(m)$ are known from proposition 8.12. With every function correspond two functions, namely the function and its complement.

$m = n$**:** $|CIB(n)_n^2| = 0$ and $|CIN(n)_n^2| = 2$. The only function is the all zero function.

$m = n - 1$**:** $|CIB(n-1)_n^2| = 2$ and $|CIN(n-1)_n^2| = 0$. The only function is the sum of all variables.

$m = n - 2$**:** $|CIB(n-2)_n^2| = 2n$ and $|CIN(n-2)_n^2| = 0$. The functions are the sum of $n - 1$ variables (note that $|CIN(n-2)_n^2| = 0$ only if $n \geq 4$).

$m = n - 3$**:** $|CIB(n-3)_n^2| = n \cdot (n-1) + \frac{1}{3} n(n-1)(3n-2)(n+1)$. The $n(n-1)$ linear functions are the sums of $n-2$ variables, and the functions with nonlinear order 2 have been characterized and counted recently by P. Camion, C. Carlet, P. Charpin, and N. Sendrier [38]. The techniques they have been using are similar to the techniques developed in this section.

The generalization of these results for arbitrary m is nontrivial, because correlation immunity is not invariant under $AGL(n)$ (cf. section 8.3.3). Therefore we first study a number that is invariant under affine transformations, namely the number of zeroes of the Walsh-Hadamard spectrum (denoted by $N_{\hat{F}}$). Subsequently these results can be applied to $CI(m)$.

Based on Dickson's theorem, the number of zeroes of the Walsh-Hadamard spectrum of a quadratic function can be calculated.

Theorem 8.8 *The number of zeroes of the Walsh-Hadamard transform of a quadratic function is given by* $N_{\hat{F}} = 2^n - 2^{2h}$ *for* $1 \leq h \leq \lfloor \frac{n}{2} \rfloor$. *The number of functions with this number of zeroes equals* $2^{n+1} \cdot M(n, 2h)$. *If* $\hat{F}(\underline{w}) \neq 0$, *then* $|\hat{F}(\underline{w})| = 2^{n-h}$.
If f *is nonbalanced, the coordinates where* $\hat{F}(\underline{w}) \neq 0$ *form a* $2h$*-dimensional subspace of* $\mathbb{Z}_2^n$, *and if* f *is balanced they form a dyadic shift of a* $2h$*-dimensional subspace.*

Proof: It will be shown that $\hat{F}$, the Walsh-Hadamard transform of

$$f(\underline{x}) = \bigoplus_{i=1}^{h} x_{2i-1} x_{2i}$$

is equal in absolute value to 2^{n-h} for $w_i = 0$, with $2h + 1 \leq i \leq n$ and equal to zero elsewhere. The theorem then follows from the application of Dickson's theorem and from the observation that the addition of the affine term ϵ only causes a dyadic shift of

the Walsh-Hadamard spectrum. The Walsh-Hadamard transform of f can be written as:

$$\hat{F}(\underline{w}) = \sum_{\underline{x}} (-1)^{\bigoplus_{i=1}^{h} x_{2i-1}x_{2i}} \cdot (-1)^{\bigoplus_{i=1}^{n} x_i w_i} .$$

Here we assume that $2h < n$. If $2h = n$, f is a bent function and the theorem is clearly true.

- In case $w_i = 0$, $2h+1 \leq i \leq n$, the expression for the Walsh transform reduces to

  $$\hat{F}(\underline{w}) = \sum_{\underline{x}} (-1)^{\bigoplus_{i=1}^{h} x_{2i-1}x_{2i}} \cdot (-1)^{\bigoplus_{i=1}^{2h} x_i w_i} .$$

  As the variables x_{2h+1} through x_n do not occur in this sum, it can be simplified to

  $$\hat{F}(\underline{w}) = 2^{n-2h} \cdot \sum_{\underline{x}'} (-1)^{\bigoplus_{i=1}^{h} x_{2i-1}x_{2i}} \cdot (-1)^{\bigoplus_{i=1}^{2h} x_i w_i} ,$$

  where $\underline{x}'$ denotes $[x_1 \ldots x_{2h}]$. By observing that the remaining sum corresponds to the Walsh-Hadamard transform of a bent function of $2h$ variables, it follows that its absolute value equals 2^h.

- In the other case, let U denote the set of indices $\{i_1, i_2, \ldots, i_k\}$ in the interval $[2h+1, n]$ for which $w_{i_j} = 1$. The Walsh-Hadamard transform can then be written as

  $$\hat{F}(\underline{w}) = \sum_{\underline{x}'} (-1)^{\bigoplus_{i=1}^{h} x_{2i-1}x_{2i}} \cdot (-1)^{\bigoplus_{i=1}^{2h} x_i w_i} \cdot \sum_{\underline{x}''} (-1)^{\bigoplus_{i_j \in U} x_{i_j}} ,$$

  where $\underline{x}'$ denotes $[x_1 \ldots x_{2h}]$ and $\underline{x}''$ denotes $[x_{2h} \ldots x_n]$. It is easily seen that the second sum vanishes, and hence $\hat{F}$ equals zero.

■

If f is nonbalanced, the nonzeroes form a subspace, and this again can be used to count the functions satisfying $CIN(m)$ in a similar way as the functions satisfying $PC(k)$.

Corollary 8.6 *The number of quadratic functions of n variables satisfying $CIN(m)$ is given by*

$$\sum_{h=1}^{\lfloor \frac{n}{2} \rfloor} L(n, 2h, m) M(2h, 2h) 2^{2h+1} ,$$

where $L(n, r, m)$ denotes the number of linear $[n, r, d]$ codes with minimum distance $d > m$.

Proof: The proof is similar to the proof of corollary 8.4. The first two factors correspond to the number of symplectic matrices of dimension n and rank $2h$ for which the nonzeroes of the Walsh-Hadamard spectrum form a linear code with minimum

distance $> m$. The addition of linear terms results in 2^{2h} nonbalanced functions that also satisfy $CIN(m)$. Indeed, according to proposition 8.3, the addition of linear terms corresponds to a dyadic shift in the Walsh-Hadamard spectrum, and the function will only remain nonbalanced if it is shifted over a vector belonging to the subspace of nonzeroes. The number of vectors in this subspace is equal to 2^{2h}. The proof is completed by the observation that $CIN(m)$ is invariant under complementation. ∎

From corollary 8.3 and from combining corollary 8.6 with the observation that $d_{max}(n,2) = \lfloor \frac{2n}{3} \rfloor$ [145], one obtains the following result.

Corollary 8.7 *There exist no quadratic functions of n variables that satisfy $CIN(m)$ if $\lfloor \frac{2n}{3} \rfloor \leq m \leq n$, and no functions satisfying $CIB(m)$ if $n-2 \leq m \leq n$.*

Note that the Walsh-Hadamard spectrum of the quadratic functions was independently studied in [187]. There it was additionally shown that if f is nonbalanced, the subspace of nonzeroes is orthogonal to the kernel of the symplectic matrix B corresponding to f. If f is balanced, one can write $\ker B$ as $M \oplus \{0, d\}$, where M is the part of $\ker B$ on which f vanishes. Then $\hat{F}(\underline{w})$ differs from zero iff $\underline{w}$ is orthogonal to M and not orthogonal to d. This might be used to count the number of functions that satisfy $CIB(m)$, but this problem seems to be harder than the nonbalanced case.

Note that the characterization of the Walsh-Hadamard spectrum also yields the distance of the quadratic functions to affine functions (cf. section 8.3.2). If n is even, the maximal distance is reached by the bent functions, namely $2^{n-1} - 2^{n/2-1}$. If n is odd, the maximal distance to affine functions is given by $2^{n-1} - 2^{\lfloor n/2 \rfloor}$.

Theorem 8.8 can be used to obtain a different expression for the number of quadratic balanced functions. There are $2(2^n - 1)$ balanced linear functions and every quadratic function with no affine terms with q zeroes in the Walsh-Hadamard spectrum corresponds to $2q$ balanced functions through the addition of affine terms (corollary 8.2). Hence the total number of balanced functions with ord ≤ 2 can also be written as

$$A_{2^{n-1}} = 2(2^n - 1) + 2 \left(\sum_{h=1}^{\lceil \frac{n}{2} \rceil - 1} (2^n - 2^{2h}) M(n, 2h) \right) .$$

This corresponds to counting the cosets of the Reed-Muller code of order one into the cosets of order two, based on [199] p. 415.

8.4.1.4 Quadratic functions satisfying higher order PC

In this section the quadratic functions satisfying higher order propagation criteria will be discussed. First, the relatively easy case of higher order $PC(1)$ will be treated, subsequently the functions satisfying $PC(n-1)$ of order 1. A function will be defined that satisfies higher order $PC(2)$ of order 2, and it will be shown that it does not satisfy $EPC(2)$ of order 2.

The following theorem characterizes all quadratic functions satisfying $PC(k)$ of order m.

Theorem 8.9 *Let f be a quadratic Boolean function of n variables, with $n > 2$. Then f satisfies $PC(1)$ of order m $(0 \leq m \leq n-2)$, iff every variable x_i occurs in at least $m+1$ second order terms of the algebraic normal form.*

Proof: For $m = 0$ it is sufficient to show that a quadratic function satisfies $PC(1)$ iff every variable occurs at least once in the second order terms of the algebraic normal form. It is clear that this condition is necessary. It is also sufficient, as it implies that for every i $(1 \leq i \leq n)$ f can be written as $f(\underline{x}) = x_i \cdot g(\underline{x}') \oplus h(\underline{x}')$, where $\underline{x}'$ denotes $[x_1 \ldots x_{i-1} x_{i+1} \ldots x_n]$, $g(\underline{x}')$ is an affine function and $h(\underline{x}')$ is a quadratic function. The value of the autocorrelation function in the unit vector $\underline{e_i}$ can then be computed as

$$r(\underline{e_i}) = \sum_{\underline{x}} f(\underline{x}) \oplus f(\underline{x} \oplus \underline{e_i}) = \sum_{\underline{x}} g(\underline{x}') = 2^{n-1} \text{ or } \hat{r}(\underline{e_i}) = 0\,,$$

which corresponds to the definition of $PC(1)$. The result for $m > 0$ follows from the observation that if m variables are fixed, the number of second order terms in which a remaining variable occurs will be reduced with at most m. ∎

Theorem 8.9 can also be interpreted as follows: the quadratic functions satisfying $PC(1)$ of order m correspond to the undirected simple graphs with n vertices with minimum degree equal to $m+1$. Determining the number of graphs with n vertices and degree $d_{\min}$ for general n and $d_{\min}$ seems to be a difficult problem [140]. In appendix D expressions are derived for the cases $d_{\min} = 0, 1, 2, n-3, n-2$, and $n-1$.

From every graph 2^{n+1} different functions can be obtained by adding an affine function. From theorem 8.4 it follows that only quadratic functions can satisfy $PC(1)$ of order $n-2$ and order $n-3$. Hence theorem 8.9 characterizes all functions with these properties. Note that there is essentially only one function that satisfies $PC(1)$ of order $n-2$, namely $s_n(\underline{x})$, hence the sum of two functions satisfying $PC(1)$ of $n-2$ is always affine.

In [194] it was shown that exactly 2^{n+1} functions satisfy $PC(1)$ of maximal order $n-2$, and a more complex construction and counting method for these functions was given in [3]. In [195] a different characterization and counting method for $PC(1)$ of order $n-3$ was developed.

The following theorem constructs and counts all quadratic functions that satisfy $PC(n-1)$ of order 0 and 1 for n odd.

Theorem 8.10 *Let f be a quadratic Boolean function of n variables, $n > 2$ and odd. Then f satisfies $PC(n-1)$ of order 0 and 1 iff f is obtained with the following construction:*

1. *Let f' be a function of $n-1$ variables satisfying $PC(n-1)$ with algebraic normal form coefficients equal to a'_{ij}.*
2. *Define $a_{ij} = a'_{ij}$ for $1 \leq i < j \leq n-1$ and*

$$a_{in} = \bigoplus_{j=0, j\neq i}^{n-1} a_{ij} \quad \text{for } 1 \leq i \leq n-1\,.$$

The number of functions satisfying $PC(n-1)$ *of order* 0 *and* 1 *is given by* $M(n-1, n-1)$.

Proof: It follows directly from theorem 8.6 that every quadratic function satisfying $PC(n-1)$ of order 0 satisfies $PC(n-1)$ of order 1. To characterize the functions satisfying $PC(n-1)$ of order 1, it is recalled that every function f^* obtained from f by fixing one input bit should satisfy $PC(n-1)$. This can be restated with the symplectic matrices B and B^* that correspond to f and f^* respectively: every matrix B^* obtained from B by deleting one column and the corresponding row should have full rank $n-1$. As the rank of a symplectic matrix is always even, this implies that B has necessarily rank $n-1$ and that any column (row) can be written as a linear combination of the other columns (rows). Any symplectic matrix B^* of rank $n-1$ can be extended in 2^{n-1} ways to a matrix B. However, if any matrix obtained from deleting one column and the corresponding row in B should have rank $n-1$, the only solution is that the added column (row) is the sum of all other columns (rows). This can be shown as follows: if a particular column and row are not selected in the sum, the deletion of this column and row from B will result in a singular matrix B^*, contradicting the requirement. ∎

If corollary 8.5 is combined with the proof of previous theorem, one obtains the following result.

Corollary 8.8 *Let* n *be an odd integer,* $n > 2$. *The number of functions satisfying* $PC(k)$ *of order* 1 *is given by* $M(n-1, n-1)$, *for* $\lfloor \frac{2(n-1)}{3} \rfloor \leq k \leq n-1$.

From the previous discussion it follows that the function of which the algebraic normal form contains all second order terms, has some interesting properties. It will be denoted with s_n or

$$s_n(\underline{x}) = \sum_{1 \leq i < j \leq n} x_i x_j \, .$$

Proposition 8.19 *The function* s_n *satisfies* $PC(k)$ *of order* m *for all values of* $k < n-m$, *and also for* $k = n-m$ *if* k *is odd.*

Proof: If m bits are kept constant in s_n, one obtains the function s_{n-m} plus an affine function. This function of $n-m$ variables satisfies $PC(n-m)$ if $n-m$ is even (theorem 8.6) and $PC(n-m-1)$ if $n-m$ is odd (theorem 8.10). ∎

However, s_n does not satisfy $EPC(2)$ of order 2, and this shows that proposition 8.17 is tight. Indeed, the directional derivative of s_n for a vector $\underline{s}$ with a 1 in position i and j is equal to $x_i \oplus x_j \oplus 1$. This function satisfies $CIB(1)$, but not $CIB(2)$. Hence $s_n(\underline{x})$ does not satisfy $EPC(2)$ of order 2. In this particular example, it is clear that if $x_i = x_j$, then $f(\underline{x} \oplus \underline{s}) \neq f(\underline{x})$, $\forall \underline{x}$ and if $x_i \neq x_j$, then $f(\underline{x} \oplus \underline{s}) = f(\underline{x})$, $\forall \underline{x}$. For $PC(2)$ the average of both cases is considered: f changes on average with a probability of 0.5. For $n \leq 5$, the functions $s_n(\underline{x})$ are the only functions for which PC and EPC are not equivalent.

To conclude this section, table 8.5 gives the number of all quadratic functions satisfying $PC(k)$ of order m for n between 3 and 7 (without affine terms). Note that here the convention is different from table 8.2: if a function satisfies $PC(k)$ of order m it is counted in all entries for $PC(k')$ of order m' with $k' \leq k$ and $m' \leq m$. This is necessary to obtain a consistent representation. The main open problems here are the characterization of functions satisfying $PC(k)$ and order m for $k > 1$, and a study of the difference between $PC(k)$ and $EPC(k)$. It was verified by computer calculation that for $n = 7$ there exists functions that satisfy $EPC(2)$ of order 2.

k \ m	0	1	2
1	4	1	0
2	1	1	–
3	0	–	–

k \ m	0	1	2	3
1	41	10	1	0
2	28	1	1	–
3	28	0	–	–
4	28	–	–	–

k \ m	0	1	2	3	4
1	768	253	26	1	0
2	448	28	1	1	–
3	168	28	0	–	–
4	28	28	–	–	–
5	0	–	–	–	–

k \ m	0	1	2	3	4	5
1	27449	12068	1858	76	1	0
2	18788	3188	1	1	1	–
3	14308	421	1	0	–	–
4	13888	1	1	–	–	–
5	13888	0	–	–	–	–
6	13888	–	–	–	–	–

k \ m	0	1	2	3	4	5	6
1	1887284	1052793	236926	15796	232	1	0
2	1419852	237048	4901	1	1	1	–
3	889672	17668	841	1	0	–	–
4	402752	13888	1	1	–	–	–
5	111104	13888	0	–	–	–	–
6	13888	13888	–	–	–	–	–
7	0	–	–	–	–	–	–

Table 8.5: The number of quadratic Boolean functions of n variables satisfying $PC(k)$ of order m divided by 2^{n+1} for $n = 3$ to 7.

8.4.1.5 Quadratic functions satisfying combined criteria

A natural question that arises is whether there exist functions that satisfy a combination of one or more criteria. From this point of view, one can combine theorem 8.6

with theorem 8.8 to obtain the following corollary.

Corollary 8.9 *Let f be a quadratic Boolean function of n variables. Then*

$$(2^n - N_{\hat{r}}) \cdot (2^n - N_{\hat{F}}) = 2^n \,.$$

It was conjectured by the author in [258] and later shown by C. Carlet in [41] that this can be generalized for arbitrary order as follows:

Theorem 8.11 *Let f be a Boolean function of n variables. Then the product of the number of zeroes of the autocorrelation function and of the Walsh-Hadamard transform lies between the following bounds:*

$$2^n \leq (2^n - N_{\hat{r}}) \cdot (2^n - N_{\hat{F}}) \leq 2^{2n} \,.$$

Proof: The upper bound is trivial. In order to prove the lower bound, it is first shown that

$$2^n - N_{\hat{r}} \geq 2^{-n} \hat{F}^2(\underline{w}),\ \forall \underline{w} \in \mathbb{Z}_2^n \,. \tag{8.1}$$

This clearly holds for $\underline{w} = \underline{0}$, as

$$2^{-n} \hat{F}^2(\underline{0}) = 2^{-n} \sum_{\underline{s}} \hat{r}(\underline{s}) \leq 2^n - N_{\hat{r}} \,.$$

Here we made use of the fact that $\hat{r}(\underline{s}) \leq 2^n$. From proposition 8.3 it follows that a dyadic shift in the Walsh-Hadamard energy spectrum only affects the sign of $\hat{r}(\underline{s})$, and hence equation (8.1) holds.
It is easy to see that

$$2^n - N_{\hat{F}} \geq \frac{\sum_{\underline{w}} \hat{F}^2(\underline{w})}{\max_{\underline{w}} \left\{ \hat{F}^2(\underline{w}) \right\}} = \frac{2^{2n}}{\max_{\underline{w}} \left\{ \hat{F}^2(\underline{w}) \right\}} \,. \tag{8.2}$$

Here the last equality follows from corollary 8.1. The theorem then follows from multiplying equations (8.1) and (8.2). ∎

It is clear that the upper bound will be achieved if $\mathrm{ord}(f) = n$, as $N_{\hat{r}} = 0$ (proposition 8.9) and $N_{\hat{F}} = 0$ (proposition 8.10). From corollary 8.9 it follows that the lower bound will be achieved if $\mathrm{ord}(f) \leq 2$. We conjectured in [258] that the only other cases where equality holds are the functions that satisfy $PC(n)$ if n is even or for functions for which $N_{\hat{r}} = 2^n - 2$ if n is odd.

However, C. Carlet showed in [41] that if $n > 6$ there exist more functions for which equality holds, namely all functions for which there exists a linear form $\underline{b} \cdot \underline{x}$ and two supplementary subspaces E and E', with $\dim E'$ even such that the restriction of f to E' is bent, and

$$\forall \underline{x} \in E, \forall \underline{y} \in E' \ :\ f(\underline{x} \oplus \underline{y}) = f(\underline{y}) \oplus \underline{b} \cdot \underline{x} \,.$$

He called this class of functions partially-bent functions and has described their properties. Determining the number of partially-bent functions seems to be hard, as it

depends on the number of bent functions (cf. section 8.4.2). The autocorrelation function and Walsh-Hadamard spectrum of the partially-bent functions have the same structure as those of the quadratic functions, i.e., they depend on $\dim E'$. For $n \leq 6$, the partially-bent functions with nonlinear order > 2 are bent functions.

An interesting open problem is to study the functions that satisfy both $PC(k)$ (or $PC(k)$ of higher order) and $CI(m)$. For functions satisfying $PC(1)$ of order $n-2$ and $n-3$ and $CI(m)$, a classification has been obtained in [196, 197]. It would be interesting to try to prove these results by combining properties of graphs and symplectic matrices.

8.4.2 Bent functions

From section 8.3, it follows that bent functions have some interesting properties from a cryptographical viewpoint. The most important properties of bent functions have been summarized in section 8.3.4. For a more extensive overview the reader is referred to [287]. In this section two constructions for bent functions are discussed, and an efficient method to count the number of bent functions of 6 variables is proposed. Subsequently some extensions for n odd are considered.

8.4.2.1 Constructions of bent functions

Most known constructions are enumerated in [334]: the two Rothaus constructions [285], the eigenvectors of Walsh-Hadamard matrices [333], constructions based on Kronecker algebra, concatenation, dyadic shifts, and linear transformations of variables. An alternative characterization of bent functions can be given in terms of a combinatorial structure called a difference set.

A dyadic shift of the truth table does not affect the autocorrelation function (proposition 8.6), and in case of a bent function one can show that this yields $2^n - 1$ bent functions with distance 2^{n-1} from the original function.

In this section a new construction will be given based on concatenation of four functions [255], that generalizes the results on concatenation in [334]. A dual result was obtained independently in [2]. The construction of a Boolean function $\hat{f}$ through concatenation implies that the vector $[\hat{f}]$ is obtained as a concatenation of different vectors $[\hat{g}_i]$.

Theorem 8.12 *The concatenation $\hat{f}$ of dimension $n+2$ of 4 bent functions $\hat{g}_i$ of dimension n is bent iff*

$$\hat{G}_1(\underline{w}) \cdot \hat{G}_2(\underline{w}) \cdot \hat{G}_3(\underline{w}) \cdot \hat{G}_4(\underline{w}) = -2^{2n}, \quad \forall \underline{w} \in \mathbb{Z}_2^n \,.$$

Proof: By applying proposition 8.5 twice, the Walsh-Hadamard transform of $\hat{F}$ can be written as

$$\hat{F}(\underline{w}_{n+2}) = \hat{G}_1(\underline{w}_n) + \hat{G}_2(\underline{w}_n)(-1)^{w_{n+1}} + \hat{G}_3(\underline{w}_n)(-1)^{w_{n+2}} + \hat{G}_4(\underline{w}_n)(-1)^{w_{n+1} \oplus w_{n+2}} \,. \tag{8.3}$$

The fact that the functions $\hat{g}_i$ are bent implies that

$$|\, \hat{G}_i(\underline{w}_{n+2}) \,| = 2^{n/2} \,.$$

Now $\hat{f}$ is bent iff

$$|\hat{F}(\underline{w}_{n+2})| = 2^{(n+2)/2}\,.$$

If w_{n+1} and w_{n+2} are both equal to 0, equality in equation (8.3) can be obtained for a given value of $\underline{w}_n$ iff three values $\hat{G}_i(\underline{w}_n)$ out of four are either positive or negative. It turns out that in these cases equality also holds if the tuple (w_{n+1}, w_{n+2}) takes on one of the other three possible values. The condition on the sign of the $\hat{G}_i(\underline{w})$'s is clearly equivalent to

$$\hat{G}_1(\underline{w}) \cdot \hat{G}_2(\underline{w}) \cdot \hat{G}_3(\underline{w}) \cdot \hat{G}_4(\underline{w}) = -2^{2n}\,,$$

which completes the proof. ∎

The following remarks can be made about this theorem:

- The order of the $\hat{g}_i$ has no importance.
- If $\hat{g}_1 = \hat{g}_2$, the theorem reduces to $\hat{g}_4 = -\hat{g}_3$, and if $\hat{g}_1 = \hat{g}_2 = \hat{g}_3$, then $\hat{g}_4 = -\hat{g}_1$. These special cases have been considered in [334].

One obtains immediately two corollaries:

Corollary 8.10 *If $\hat{f}$, $\hat{g}_1$, $\hat{g}_2$, and $\hat{g}_3$ are bent then $\hat{g}_4$ is also bent.*

Corollary 8.11 *If the concatenation of 4 arbitrary vectors of dimension n is bent, then the concatenation of all 4! permutations of these vectors is bent.*

Note that through dyadic shifting of f only 4 of the 24 permutations can be obtained. This theorem is best possible in the sense that if the function is split up in 8 vectors, all permutations will in general not result in a bent function.

In [334] it was observed that the concatenation of the rows of a Walsh-Hadamard matrix yield a bent function. We obtained the following generalization [255].

Proposition 8.20 *For $n = 2m$, consider the rows of the Walsh-Hadamard matrix H_m. The concatenation of the 2^m rows or their complement in arbitrary order results in $(2^m)!\, 2^{2^m}$ different bent functions of n variables.*

This construction was suggested independently in [2]. Instead of proving this proposition, we will give the proof of K. Nyberg, who showed that this construction is equivalent to Maiorana's construction [238]. The proof requires three simple lemma's.

Lemma 8.2 *Let $f' = \left[(-1)^{f(\underline{x},\underline{y})}\right]$ where $\underline{x}$, $\underline{y} \in \mathbb{Z}_2^n$ and f is a function from $\mathbb{Z}_2^{2n}$ to $\mathbb{Z}_2^n$. Then*

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes f' = \left[(-1)^{g(\underline{x},\underline{y})}\right]\,,$$

where $\underline{x}$, $\underline{y} \in \mathbb{Z}_2^{2n}$,
and $g(x_1, \ldots, x_n, x_{n+1}, y_1, \ldots, y_n, y_{n+1}) = f(x_1, \ldots, x_n, y_1, \ldots, y_n) + x_{n+1} \cdot y_{n+1}$.

Lemma 8.3

$$H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \left[(-1)^{f(x,y)}\right] ,$$

where x, $y \in \mathbb{Z}_2^n$ *and* $f(x,y) = x \cdot y$.

Lemma 8.4 *The Hadamard matrix of order* n *is given by*

$$H_n = \left[(-1)^{f(\underline{x},\underline{y})}\right] ,$$

where $\underline{x}$, $\underline{y} \in \mathbb{Z}_2^n$ *and* $f(\underline{x},\underline{y}) = \underline{x} \cdot \underline{y}$.

The proof follows by induction from the preceding lemmas.

The equivalence between the two constructions can now easily be shown.

Proposition 8.21 *Let* $H_n(\pi, g)$ *be the* $2^n \times 2^n$ *matrix obtained from* H_n *by permutation of the rows (the row* $\underline{x}$ *in* $H_n(\pi, g)$ *is row* $\pi(\underline{x})$ *of* H_n*) followed by eventual complementation of the rows (the row* $\underline{x}$ *in* $H_n(\pi, g)$ *is complemented* $\Leftrightarrow g(\underline{x}) = 1$*). Then*

$$H_n(\pi, g) = \left[(-1)^{f(\underline{x},\underline{y})}\right] ,$$

where $f(\underline{x},\underline{y}) = g(\underline{x}) + \pi(\underline{x}) \cdot \underline{y}$.

This latter construction is exactly the Maiorana construction.

8.4.2.2 Counting bent functions

The number of quadratic bent functions can be obtained from theorem 8.6, but the number of bent functions of larger order and for arbitrary n remains an open problem. A lower bound can be obtained from the Maiorana construction. An improvement was obtained in [2], but for $n \geq 8$ it is negligible. An upper bound for $n > 2$ can be computed based on the restriction on the nonlinear order:

$$2^{2^{n-1}+d(n)} \quad \text{with} \quad d(n) = \frac{1}{2}\binom{n}{n/2} .$$

Table 8.6 gives the lower bound, the number of bent functions (if known), the upper bound, and the number of Boolean functions for n between 2 and 8.

For $n = 6$, the number of bent functions has been determined with a new counting method. In this case the bent functions have nonlinear order 2 or 3. As affine terms can be neglected, an exhaustive search through all 2^{35} functions is feasible but very computer intensive. It was suggested by O. Rothaus [285] that if the third order terms are divided into classes that are not equivalent under affine transformation of variables (second order terms are not considered), four different classes are obtained. However, we were able to show that there are in fact five classes, but the fifth class does not

n	# Boolean	lower bound	# bent	upper bound
2	16	8	8	8
4	65536	384	896	2048
6	2^{64}	$2^{23.3}$	$2^{32.3}$	2^{42}
8	2^{256}	$2^{60.3}$	?	2^{163}

Table 8.6: The number of Boolean functions, the number of bent functions, and an upper and lower bound for the number of bent functions for n between 2 and 8.

lead to bent functions. A computer program was written to determine the size of each class. This could be done by applying affine transformations of variables until a canonical form was obtained. Secondly the number of bent functions in each class was obtained by adding all possible second order terms to a single representative of this class. Table 8.7 describes the different classes, and gives the number of elements per class and the number of bent functions corresponding to each element. The total number of bent functions of 6 variables is equal to $42,386,176 \cdot 128 = 5,425,430,528$.

class nr.	representative
0	(no third order terms)
1	$x_1x_2x_3$
2	$x_1x_2x_3 \oplus x_4x_5x_6$
3	$x_1x_2x_3 \oplus x_1x_4x_5$
4	$x_1x_2x_3 \oplus x_1x_4x_5 \oplus x_2x_4x_6$
5	$x_1x_3x_6 \oplus x_1x_4x_5 \oplus x_1x_5x_6 \oplus x_2x_3x_4 \oplus x_2x_5x_6$

class nr.	# el./class	# bent functions/el.	tot # of bent functions	
0	1	13,888	13,888	$(1 \cdot 13,888)$
1	1,395	1,344	1,874,880	$(135 \cdot 13,888)$
2	357,120	0	0	
3	54,684	192	10,499,328	$(756 \cdot 13,888)$
4	468,720	64	29,998,080	$(2,160 \cdot 13,888)$
5	166,656	0	0	
tot	1,048,576		42,386,176	$(3,052 \cdot 13,888)$

Table 8.7: Representative and size of the equivalence classes for the third order functions for $n = 6$, and the number of bent functions in every class.

This result disproves the conjecture in [2] that all bent functions can be obtained either from the concatenation approach or from the Maiorana construction (or the

Walsh-Hadamard matrix). For $n = 6$, the concatenation approach and the Maiorana approach yield $37,879,808$ respectively $10,321,920$ bent functions. The total of $48,201,728$ represents a fraction of less than 1% of the bent functions of 6 variables.

Note that the weight distribution of all cosets of the first order Reed-Muller code for $n = 6$ has been determined recently by J. Maiorana [200]. The number of bent functions can probably be obtained from these results as well.

8.4.2.3 Extension of bent functions for odd n

As there exist no bent functions for odd n, it is interesting to extend the concept. In [208], the following proposal for extension was made:

$$f(\underline{x}) = (1 \oplus x_n) f_0(x_1, \ldots, x_{n-1}) \oplus x_n f_1(x_1, \ldots, x_{n-1}), \quad \text{with} \quad f_0, f_1 \text{ bent}.$$

It is clear that $\mathrm{ord}(f) \leq (n+1)/2$. It can be shown with proposition 8.5 that for half of the values of $\underline{w}$, $\hat{F}(\underline{w}) = 0$ and for the other half $|\hat{F}(\underline{w})| = 2^{(n+1)/2}$. The maximum distance to affine functions hence equals $2^{n-1} - 2^{(n-1)/2}$. However, these functions do not necessarily satisfy $PC(1)$: if $f_0 = f_1$, then $\hat{r}(0 \ldots 01) \neq 0$, but for all other values of $\underline{s} \neq \underline{0}$, $\hat{r}(\underline{s}) = \underline{0}$. From proposition 8.7 it follows that if $\underline{s}_n \neq \underline{0}$, $\hat{r}(\underline{s}_n\, 0) = 0$. If f_0 and f_1 are chosen such that they are almost orthogonal, or $\hat{c}_{\hat{f}_0, \hat{f}_1}(\underline{s}_n) = 0$, $\forall \underline{s}_n \neq [11 \ldots 1]$, f will satisfy $PC(n-1)$. Another observation is that f will be balanced iff f_0 and f_1 have a different number of zeroes and ones.

If n is odd, the maximal distance to a linear function is equal to the covering radius of the first order Reed-Muller code. Determining this covering radius ρ_n for odd n seems to be a hard problem [55]. It is known that [186, 187]

$$2^{n-1} - 2^{\lceil (n-1)/2 \rceil} \leq \rho_n \leq 2^{n-1} - 2^{(n-2)/2}.$$

For n even both bounds are equal. For n odd the lower bound is achieved by the quadratic functions corresponding to the symplectic matrices with rank $n-1$. For $n = 1$, 3, 5, and 7 ρ_n is equal to its lower bound, and for $m \geq 15$ one has

$$2^{n-1} - \frac{108}{128}\, 2^{(n-1)/2} \leq \rho_n \leq 2^{n-1} - 2^{(n-2)/2}.$$

Two important conjectures are that ρ_n is even for $n \geq 3$ and that the upper bound is asymptotically tight. In [187] it is shown that for $n = 9$ the lower bound can only be improved by functions with nonlinear order at least 4.

8.5 Construction of Boolean functions

If a function with special properties is required, several methods can be applied. The simplest method is exhaustive search, but there are several alternatives. For many requirements, there exist special constructions, either direct or recursive. If several properties have to be satisfied together, it is clear that one will first try to construct a function that meets the stronger requirement, and subsequently try to modify it

such that it satisfies additional conditions. This was suggested in [208] where the example was given of a balanced function with large distance to affine functions: if n is even, one starts with a bent function, and modifies this function in such a way that it becomes balanced. In that case the distance to affine functions can decrease at most with $2^{n/2-1}$. An alternative for exhaustive search might to use techniques for nonlinear optimization, like simulated annealing. This section will overview these methods briefly.

8.5.1 Exhaustive search

The simplest way to find a function with certain properties is to search exhaustively for these functions. For small values of n, this is only feasible if $n \leq 5$, as the number of Boolean functions of 6 variables is already equal to 2^{64}. If $\mathrm{ord}(f)$ is limited to 3, the upper bound is $n = 6$, and for quadratic functions the upper bound is equal to 9. For all larger values, one has to randomly select a function and test whether or not it satisfies certain properties. In that case it is important to have an idea about the probability of success.

In order to increase the number of functions that can be reached by exhaustive search, one can limit the search by considering equivalence classes of the symmetry group of the criterion. Research on the number of equivalence classes of Boolean functions has been carried out in the sixties, with as main goal the design of logic networks [130, 142]. The results are obtained by group theoretic counting techniques. The classes that have been studied are complementation of variables, permutation of variables, $GL(n)$, and $AGL(n)$. A second line of research is to determine the asymptotic size of the number of classes. In [143] it was shown that if the order g of the group is smaller than

$$2^{2^{n-1}(\frac{1}{2}-\epsilon \log_2 n)} \quad \text{with} \quad \epsilon > 0\,,$$

the number of classes is asymptotically equivalent to $2^{2^n}/g$, and this will always be a lower bound. In practice, the number of equivalence classes is known for $n \leq 6$, and the asymptotic estimates give tight lower bounds for $n \geq 6$. However, this does not mean that it is always easy to find a representative for each class.

For simple criteria like balancedness, nonaffinity, completeness, and symmetry ($f(\underline{x}) = f(\underline{y})$ if $\mathrm{hwt}(\underline{x}) = \mathrm{hwt}(\underline{y})$), the number of functions satisfying any combinations of these criteria has been studied in [222]. Earlier work on the same subject can be found in [10, 13, 131]. From [13] it follows that the number of Boolean functions of n variables that are balanced, nonaffine, and complete can for $n \geq 6$ be approximated by

$$\binom{2^n}{2^{n-1}} - n\binom{2^{n-1}}{2^{n-2}} - n2^{2^{n-1}}\,.$$

This means that for large n almost all balanced functions are nonaffine and complete. The balanced functions with maximal nonlinear order $n-1$ have been characterized

in [35]. In this report it was also shown that their number is equal to

$$\frac{2^n-1}{2^n}\left[\binom{2^n}{2^{n-1}}-\binom{2^{n-1}}{2^{n-2}}\right].$$

For more complex criteria like correlation immunity and $PC(k)$, no results are available. Another problem that seems to be very hard is determining the number of Boolean functions of a given nonlinear order > 2 with a given Hamming weight [40], [199], p. 446. One can only determine for small n the number of functions satisfying these criteria.

The results for $PC(k)$ of order m for n equal to 3, 4, and 5 are given in table 8.8. Note that the same convention is used as in table 8.5, i.e., if a function satisfies $PC(k)$ of order m it is counted in all entries for $PC(k')$ of order m' with $k' \leq k$ and $m' \leq m$. From comparison with table 8.5 it can be seen that the only entries that contain third order terms are the functions of four variables satisfying $PC(1)$ of order 0, and the functions of five variables that satisfy $PC(1)$ of order 0 and 1 and $PC(2)$ of order 0. Surprisingly, this last class also contains 192 fourth order functions.

k	$\lvert PC(k,0)_3 \rvert$	$\lvert PC(k,1)_3 \rvert$	$\lvert PC(k,2)_3 \rvert$
1	4	1	0
2	1	1	–
3	0	–	–

k	$\lvert PC(k,0)_4 \rvert$	$\lvert PC(k,1)_4 \rvert$	$\lvert PC(k,2)_4 \rvert$	$\lvert PC(k,3)_4 \rvert$
1	129	10	1	0
2	28	1	1	–
3	28	0	–	–
4	28	–	–	–

k	$\lvert PC(k,0)_5 \rvert$	$\lvert PC(k,1)_5 \rvert$	$\lvert PC(k,2)_5 \rvert$	$\lvert PC(k,3)_5 \rvert$	$\lvert PC(k,4)_5 \rvert$
1	430040	813	26	1	0
2	3760	28	1	1	–
3	168	28	0	–	–
4	28	28	–	–	–
5	0	–	–	–	–

Table 8.8: The number of Boolean functions of n variables satisfying $PC(k)$ of order m divided by 2^{n+1} ($\lvert PC(k,m)_n \rvert$) for $n = 3$, 4, and 5.

The properties of these special fourth order functions will be described here. The nonzeroes of the autocorrelation function have all absolute value 8. The zeroes of the autocorrelation function form the set of all 15 vectors with Hamming weight 1 or 2.

An example of this class is

$$f_1(\underline{x}) = x_1x_2x_3x_4 \oplus x_1x_2x_3x_5 \oplus x_1x_2x_4x_5 \oplus x_1x_3x_4x_5 \oplus x_2x_3x_4x_5 \\ \oplus x_1x_4 \oplus x_1x_5 \oplus x_2x_3 \oplus x_2x_5 \oplus x_3x_4 \,.$$

The value distribution of the Walsh-Hadamard spectrum (table 8.9) shows that this function satisfies $CIN(1)$. A related function f_2 can be defined as

$$f_2(\underline{x}) = f_1(\underline{x}) \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \,.$$

The Walsh-Hadamard spectrum of f_2 is obtained by a dyadic shift over [11111] of the Walsh-Hadamard spectrum of f_1, resulting in a balanced function satisfying $PC(2)$ (table 8.9). Note that it is not possible to obtain from f_1, through an affine transformation of variables, a function that satisfies $CIB(1)$, as this requires the vanishing of all coefficients of order 4 [330].

hwt($\underline{w}$)	0	1	2	3	4	5
$\lvert\hat{F}_1(\underline{w})\rvert$	12	0	4	8	4	0
$\lvert\hat{F}_2(\underline{w})\rvert$	0	4	8	4	0	12

hwt($\underline{s}$)	0	1	2	3	4	5
$\hat{r}_1(\underline{s})$	32	0	0	8	8	−8
$\hat{r}_2(\underline{s})$	32	0	4	−8	8	8

Table 8.9: Value distribution of the Walsh-Hadamard spectrum and the autocorrelation function for the functions f_1 and f_2.

8.5.2 Recursive construction methods

Recursive construction methods have been obtained for several classes of functions. They can be based on the concatenation of two functions: with the results in this chapter one can directly obtain the algebraic normal form, the autocorrelation function and the Walsh-Hadamard spectrum of the new function. For correlation immune functions of maximum nonlinear order, a recursive construction was proposed by T. Siegenthaler in [307] that increases n and (possibly) the nonlinear order by 1. A recursive construction for balanced correlation immune functions was suggested in [38]. Here both n and m are increased by one. The construction can start from the known functions of $n-1$ variables that satisfy $CIB(0)$. For bent functions, recursive constructions have been given in [40, 285]. For functions satisfying $PC(1)$ of order m, a recursive construction has been suggested in [114].

8.5.3 Nonlinear optimization

If a function has to satisfy contradicting requirements, like a good entropy profile (cf. section 8.3.3) and a good autocorrelation profile (cf. section 8.3.4), it is very unlikely

that an explicit construction method will be found. In that case one will not require that certain values of Walsh-Hadamard spectrum and autocorrelation function are zero, but that these values should be small.

In [115] a simulated annealing method is suggested to find balanced functions with a good entropy profile (mainly the values with Hamming weight 1 and 2 are considered) and a relatively large distance to affine functions. Choosing the parameters of the algorithm has to be done on a heuristic basis. For the example of $n = 6$, it is claimed that the success probability of the algorithm is about 300 times better than exhaustive search among balanced functions, but the number of operations for the optimization algorithm is not indicated. Maybe this limited number of criteria could be satisfied easily by slightly modifying a good quadratic function or higher order correlation immune function. Because of the small number of variables, an exhaustive operation might be even feasible here if one takes into account equivalence classes. If however additionally criteria would be imposed, and the number of variables would increase, this method is certainly promising.

8.6 Extensions to S-boxes

In the majority of cryptographic applications, mappings from n to m bits are used, that are generally called S-boxes. It is clear that the formulation of criteria for S-boxes is much more complicated. Most criteria for Boolean functions can be applied to the individual output bits, but many other possibilities exist, and for the time being a systematic theory is certainly lacking. One can require that the mutual information between a single input or a single output and all other input and output variables is small, and impose similar requirements for differences between input and output bits. Moreover, one can impose that the entropy difference between output and input (and between output difference and input difference) is minimal. By considering more output bits at the same time, one can also require that if a single input bit is modified, at least k output bits should be changed, i.e., imposing a minimum requirement instead of an average requirement. Other necessary properties could be that the mapping is invertible, or (if $n > m$) that the mapping is invertible if certain input bits are kept constant. In these cases one can also impose properties to the inverse function.

An important observation that was made by W. Van Leekwijck, L. Van Linden, and the author (cf. [320]) is that rather than applying criteria to single output bits, one should impose that these criteria have to be satisfied *by any linear combinations of output bits*. This approach was also used by K. Nyberg in [239] to define perfect nonlinearity of S-boxes from $GF(q^n)$ to $GF(q^m)$. She proposed two constructions for perfect nonlinear S-boxes. In [240] she has shown that the distance to affine functions is invariant under linear permutations of input and output and that the distance to affine functions of a permutation and of its inverse are equal. In the same article, she has introduced another criterion to characterize the nonlinearity of functions from $GF(q^n)$ to $GF(q)$, namely the linearity dimension. She has also studied the linearity dimension of quadratic functions, and showed that it is related to the rank of the

corresponding symplectic matrix. Finally she has described a construction for S-boxes with high nonlinearity consisting of quadratic functions. In the work of J. Pieprzyk [246, 247, 249] S-boxes with high nonlinearity are studied and constructed, but his definition of nonlinearity does not consider all linear combinations of output functions. The S-boxes of [247] have been used in the design of the Feistel cipher LOKI [33, 34]. In [177] a recursive construction is given for S-boxes for which all individual output bits satisfy $PC(1)$. In [131, 222] some results are given on the number of S-boxes that satisfy some simple criteria.

Work on S-boxes has mainly been stimulated by investigations on the properties of the S-boxes of DES [8, 108]. The properties of the S-boxes and related attacks have been described in [20, 23, 31, 48, 50, 75, 77, 78, 89, 90, 101, 131, 146, 286, 301, 324, 338]. Alternative constructions for the DES S-boxes have been described in [1, 3, 178]. The problem with these S-boxes is that they have some nice properties, but that they do not necessarily yield a secure block cipher. It is clear from the differential attack on DES by E. Biham and A. Shamir [20, 23] that if no special design criteria are taken into account when designing the S-boxes, a weaker cipher will be obtained. These criteria do not only depend on individual S-boxes, but on the properties of several S-boxes at the same time. The S-boxes suggested in [178] satisfy $PC(1)$, but the best iterative characteristic has probability 1/51.2 for 2 rounds (an input exor of $\texttt{0B30}_\texttt{x}$ to S-box 6 and 7), while the best iterative characteristic for DES has a probability of about 1/234. This means that a differential attack requires 2^{34} chosen plaintexts instead of 2^{47}, and $2^{48.5}$ known plaintexts instead of 2^{55}. This leads to the conclusion that theoretically interesting criteria are not sufficient, and that ad hoc design criteria are required.

8.7 Conclusion

In this section new and existing criteria for Boolean functions have been described. New constructions and counting methods have been proposed, and an overview has been given of methods to find functions that satisfy certain criteria.

In most cryptographic applications, S-boxes are used. The study of criteria for S-boxes, and the construction of good S-boxes is more difficult than similar problems for Boolean functions.

Many open problems remain, and it is clear that only a limited number of them will be solved in the near future: several problems are related to difficult open problems in combinatorial theory and in coding theory.

The criteria to be imposed on a Boolean function and on an S-box strongly depend on the particular application in which it will be used. In general, a well designed Boolean function or S-box will be stronger than a randomly selected one, and it will always be a compromise between several criteria. Even if no direct relation is established with a specific cryptographic scheme, the study of certain criteria can discover the trade-offs that the designer has to face.

Chapter 9

Conclusions and Open Problems

The more we study the more we discover our ignorance. *Percy Bysshe Shelley*

The main goal of this chapter is to summarize the current status of research on hash functions and to highlight the new contributions.

It has been shown that hash functions play an important role in the protection of authentication of information. Therefore one would like to have the disposal of a provably secure hash function that is efficient in both hardware and software. However, this seems impossible in the near future, and the best one can do is to study the problem according to the three approaches in cryptography.

In the information theoretic approach, provably secure schemes have been studied and the limits have been explored. For practical applications that require provable security, universal hash functions offer an interesting solution, but in general the required key size is too large. One can expect that new schemes will extend existing trade-offs between the size of the hashcode, the size of the key, and the probability of impersonation and substitution.

The most important contribution from the complexity theoretic approach are certainly the definitions. Secondly it suggests some interesting design principles. The shortcoming of this approach is that it deals with asymptotic and worse case results. Moreover, the underlying model of computation does not incorporate the idea of a birthday attack, which is a serious problem in the context of hash functions. The main open problem in the complexity theoretic approach is whether one-way functions are sufficient for collision resistant hash functions. Further research is also required to improve the efficiency of existing constructions.

The largest part of this thesis has been devoted to the system based approach. From the many proposals that have been discussed, only few have survived, from which one might conclude that our knowledge of the design of efficient cryptographic hash functions is very limited. In this thesis twelve proposals for hash functions have been cryptanalyzed. This situation is comparable to the evolution of stream ciphers, where

many schemes were proposed and broken [287]. The main difference is that stream ciphers have a longer history, and that in the case of stream ciphers the commercial 'standard' algorithms have not been published. In case of block ciphers, the situation between 1977 and 1987 was different because of the availability of a widely accepted commercial solution proposed by NBS, namely DES [108]. We now know that the designers had an advantage of at least ten years on the open research community. During the last five years new proposals for block ciphers are emerging.

An important contribution of this thesis is a taxonomy of attacks. It can certainly serve as a caveat for designers of hash functions and assist in the evaluation of a hash function. A successful attack on a hash function (except for a MAC) is in most cases much more dangerous than an attack on an encryption algorithm: if the collision is of a general nature, there is no way in which the system can prohibit from effectively using the collision. In case of an encryption algorithm, one could increase the frequency of key change, which is effective against most attacks.

For hash functions based on block ciphers, the designer tries to reuse the expensive design effort for the block cipher. The case where key size, block length, and size of the hashcode are equal has been completely solved in this thesis. Unfortunately no block ciphers are currently available with a sufficiently large block size that would yield a collision resistant hash function. Three new schemes for a hash function based on a block cipher have been proposed. Many open problems remain on schemes for which the size of the hashcode is larger than the block length, or for which the key size differs significantly from the block size. An additional problem is that the use of a block cipher in a hash function can strengthen the requirements that have to be imposed on the block cipher.

The hash functions based on modular arithmetic have been evaluated in a similar way as the hash functions based on block ciphers. The scheme proposed by F. Cohen and Y.J. Huang was cryptanalyzed. Some doubt has been raised about this type of hash functions, but in a limited number of applications they can offer a good solution that is scalable. If the efficiency is not too important, schemes are available for which the security is provably equivalent to number theoretic problems like discrete logarithm or factoring. For faster schemes, many attacks can be explained by the fact that more designers have suggested a scheme 'on the edge'. If it is acceptable to decrease the performance with a factor 2 to 4, it is much easier to come up with a secure solution.

For dedicated hash functions, the basic questions to be considered by the designer are the general structure (e.g., based on a collision resistant function or not, redundancy in the message, ...), and the way to introduce the nonlinearity. All answers will be influenced by the choice of the implementation: is the algorithm hardware or software oriented, or should it have an acceptable performance in both hardware and software. For the nonlinearity, three basic options seem to be available: select large S-boxes that are weak but efficient (e.g., multiplication), select large random S-boxes or select small but optimal S-boxes. The three options can be influenced by the study of cryptographic properties of Boolean functions and S-boxes. In chapter 8 several new results on the cryptographic properties of Boolean functions have been presented.

It is widely accepted that a new cryptographic scheme should be used only if it has been intensively evaluated, and not only by its designer. If the designer can not afford to pay a number of capable experts to spend a sufficiently long time on evaluation a scheme, one has to hope that publication of the scheme will attract cryptanalysts.

- The main problem with this approach is that most currently available designs are too close 'to the edge', mainly for performance reasons. On the one hand, this attracts cryptanalysts to spend their time on the scheme (and publish a paper on the attack), but on the other hand this leads to a moving target: one (and probably more) new versions appears, which implies that the cryptanalysts will lose interest and, which is even worse, the confidence in the scheme gets lost (whether the new scheme is broken or not). The only way to avoid this problem is to have a sufficient security margin, such that one can show that only a limited number of 'rounds' can be broken.

- A second element to stimulate evaluators is that every design decision is well motivated. This implies that the designer should explain why alternative choices were rejected. In this case an evaluator does not lose time to 'reverse engineer' the algorithm to find out what the designers already knew. This can also help to judge the quality of the design.

- A third observation is that the credit to the cryptanalyst, and hence partially his effort, seems to be proportional to the scientific reputation of the designer or design team. Business aspects can also have an important influence. Standards seem to be especially attractive. The problem here is that the main evaluation will only take place *after* the publication of the standard.

In this view the EEC-funded RIPE project (Race Integrity Primitives Evaluation, RACE 1040) [259, 277, 317], which provided for a thorough evaluation of submitted schemes based on independent criteria, has suggested an interesting approach. Its disadvantage is that the quality of the outcome of the evaluation depends strongly on the quality of the algorithms that are submitted.

If one has to choose a hash function now, one can select one of the proposals in this thesis that has not been broken. For highly critical applications, it is recommended to parallelize (cf. section 2.4.5) two or more hash functions that have already been evaluated for some time, and that are based on different design principles.

In view of the nature of the hash functions that are currently available, there might be a need for a MAC that allows for a very fast software implementation. It could be based on similar principles as the MD4 family. For a hardware oriented MAC, the schemes based on a block cipher are sufficient. A second interesting scheme would be a CRHF that is fast in hardware, or preferably fast in both hardware and software.

Of course it would be more interesting to evaluate schemes of which the security, or at least certain properties, can be proved. This requires that these schemes are based on mathematical structures, like properties of groups, codes, or graphs. This would clearly facilitate the evaluation of the scheme. As long as no provably secure schemes are available, an open evaluation will be necessary.

Appendix A

Modes of Use

Rien ne pèse tant qu'un secret. *La Fontaine*

If a block cipher is used to encrypt data, it is in most cases not recommended to split the data into blocks and encrypt every block separately. Depending on the application, several *modes of use* or *modes of operation* have been proposed and standardized: the four modes of use for DES were specified in the 1981 US Federal Information Processing Standard, Publication 81 [109]. An international standard specifying four modes of use for a 64-bit block cipher algorithm was published in 1987 [152], and a generalization for an n-bit block cipher algorithm was published in 1991 [158]. The main body of the latter standard was also improved, and an informative annex was added that contains some information about the properties of the different modes of use, and about handling incomplete blocks. For a more extensive discussion of the modes of use, the reader is referred to chapter 4 of [74].

In this appendix, the plaintext will consist of t blocks, denoted with P_1 through P_t. The corresponding ciphertext blocks will be denoted with C_1 through C_t. For the first two modes, that are called block modes, the size of a plaintext block will be n bits, where n is the block length of the block cipher algorithm. For the two other modes, that are called stream modes, the size of a plaintext *variable* is equal to j bits, where j is a parameter of the mode. When discussing the error propagation property, it will be assumed that the block cipher has the property that changing one or more plaintext bits results in an average 50% change in the ciphertext.

A.1 The ECB mode

In the Electronic Codebook Mode (ECB) mode, the block cipher is applied in a straightforward way. Every plaintext block is encrypted independently or

$$C_i = E(K, P_i) \qquad \text{for } i = 1, \ldots, t\,.$$

The decryption operation for the ECB mode is given by:

$$P_i = D(K, C_i) \qquad \text{for } i = 1, \ldots, t\,.$$

The properties of the ECB mode are:

1. Encryption or decryption of a block can be carried out independently of the other blocks.
2. Reordering of the ciphertext blocks will result in the corresponding reordering of the plaintext blocks.
3. The same plaintext block always produces the same ciphertext block (for the same key) making it vulnerable to a "dictionary attack".
4. In the ECB mode, a single or multiple bit error within a single ciphertext block will only affect the decryption of the block in which the error occurs. Hence each bit of the recovered plaintext version of this block will have an average error rate of 50%.

The ECB mode is used in a limited number of applications where the repetition characteristic is acceptable or blocks have to be accessed individually. A typical example is key encryption.

A.2 The CBC mode

In the Cipher Block Chaining (CBC) mode the blocks are chained together. The first ciphertext block is obtained from

$$C_1 = E(K, P_1 \oplus SV)\,,$$

where SV is the starting variable. The other ciphertext blocks are calculated as follows:

$$C_i = E(K, P_i \oplus C_{i-1}) \qquad \text{for } i = 2, \ldots, t\,.$$

The first plaintext block can be recovered from

$$P_1 = D(K, C_1) \oplus SV\,.$$

For the other plaintext blocks one has

$$P_i = D(K, C_i) \oplus C_{i-1} \qquad \text{for } i = 2, \ldots, t\,.$$

The properties of the CBC mode are:

1. The chaining operation makes the ciphertext blocks dependent on all preceding plaintext blocks, and therefore blocks can not be rearranged. Note that this does not imply that the CBC mode protects the integrity of the plaintext (cf. chapter 5).

2. The use of different SV values prevents the same plaintext encrypting to the same ciphertext.
3. In the CBC mode, one or more bit errors within a single ciphertext block will affect the decryption of two blocks (the block in which the error occurs and the succeeding block). If the errors occur in the ith ciphertext block, each bit of the ith decrypted plaintext block will have an average error rate of 50%. The $(i+1)$th decrypted plaintext block will have only those bits in error that correspond directly to the ciphertext bits in error.

The CBC mode is the most secure and preferred mode to encrypt data. It will only be replaced by the CFB or the OFB mode in case of special requirements on synchronization or error propagation.

A.3 The CFB mode

The Cipher Feedback Chaining (CFB) mode is a stream mode, i.e., the size of the plaintext variables j can be arbitrarily chosen between 1 and n. The scheme that will be described here is a simplified version of the more general scheme described in the ISO standards.

The CFB mode makes use of an internal n-bit register. The state of this register before the encryption or decryption of the ith block will be denoted with X_i. Before the encryption and decryption operation, this register is initialized with the starting variable or $X_1 = SV$.

The encryption of plaintext variable i consists of the following two steps:

$$\begin{aligned} C_i &= P_i \oplus \text{rchop}_{n-j}(E(K, X_i)) \\ X_{i+1} &= \text{lchop}_j(X_i) \| C_i \, . \end{aligned}$$

Here rchop_t denotes the function that drops the t rightmost bits of its argument, and lchop_t denotes the function that drops the t leftmost bits of its argument. The decryption operates in a similar way:

$$\begin{aligned} P_i &= C_i \oplus \text{rchop}_{n-j}(E(K, X_i)) \\ X_{i+1} &= \text{lchop}_j(X_i) \| C_i \, . \end{aligned}$$

The properties of the CFB mode are:

1. The chaining operation makes the ciphertext variables dependent on all preceding plaintext variables, and therefore j-bit variables are chained together and can not be rearranged. Note that this does not imply that the CFB mode protects the integrity of the plaintext (cf. chapter 5).
2. The use of different SV values prevents the same plaintext encrypting to the same ciphertext.
3. The encryption and decryption processes in the CFB mode both use the encryption form of the algorithm.

4. Selection of a small value of j will require more cycles through the encryption algorithm per unit of plaintext, and thus cause greater processing overheads.
5. If j is chosen equal to the character size, this mode is *self synchronizing*. This means that if one or more j-bit characters between sender and receiver are lost, automatic resynchronization will occur after n bits.
6. In the CFB mode, errors in any j-bit unit of ciphertext will affect the decryption of succeeding ciphertext until the bits in error have been shifted out of the CFB internal register. The first affected j-bit unit of plaintext will be garbled in exactly those places where the ciphertext is in error. Succeeding decrypted plaintext will have an average error rate of 50% until all errors have been shifted out of the internal register.

The main reason to use the CFB mode is the self synchronizing property. This is especially important in a communication environment, where one chooses j equal to the character size (typically 1 or 8 bits). The error propagation properties are comparable to those of the CBC mode. The decrease in performance is proportional to n/j.

A.4 The OFB mode

The Output Feedback Chaining (OFB) mode is also a stream mode, i.e., the size of the plaintext variables j can be arbitrarily chosen between 1 and n.

The OFB mode makes use of an internal n-bit register. The state of this register before the encryption or decryption of the ith block will be denoted with X_i. Before the encryption and decryption operation, this register is initialized with the starting variable or $X_1 = SV$.

The encryption of plaintext variable i consists of the following three steps:

$$\begin{aligned} Y_i &= E(K, X_i) \\ C_i &= P_i \oplus \text{rchop}_{n-j}(Y_i) \\ X_{i+1} &= Y_i \,. \end{aligned}$$

Note that the only difference with the CFB mode is the updating of the internal register. The decryption operates in a similar way:

$$\begin{aligned} Y_i &= E(K, X_i) \\ P_i &= C_i \oplus \text{rchop}_{n-j}(Y_i) \\ X_{i+1} &= Y_i \,. \end{aligned}$$

The properties of the OFB mode are:

1. The absence of chaining makes the OFB mode more vulnerable to specific attacks.
2. The use of different SV values prevents the same plaintext encrypting to the same ciphertext.

3. The encryption and decryption processes in the OFB mode both use the encryption form of the algorithm.
4. The OFB mode does not depend on the plaintext to generate the key stream used to add modulo 2 to the plaintext.
5. Selection of a small value of j will require more cycles through the encryption algorithm per unit of plaintext, and thus cause greater processing overheads.
6. The OFB mode does not extend ciphertext errors in the resultant plaintext output. Every bit in error in the ciphertext causes only one bit to be in error in the decrypted plaintext.

The main reason to use the OFB mode is the error propagation property. In order to optimize the performance, one will preferably choose j equal to 64. It is very important that each re-initialization uses a value of SV different from the SV values used before with the same key. The reason for this is that an identical bit stream would be produced each time for the same parameters. This would be susceptible to a "known plaintext attack".

Appendix B

Birthday Attacks and Collisions

The protection provided by encryption is based on the fact that most people would rather eat liver than do mathematics. *Bill Neugent*

B.1 Introduction

A variety of attacks have surprised the designers of cryptographic schemes because they require only $O(\sqrt{n})$ of operations, where a naive approach would require $O(n)$ trials. Because this holds for the "birthday paradox" as well, all these attacks have been called birthday attacks; sometimes they are called square root attacks. Although their asymptotic efficiency is very similar, different models correspond to slightly different distributions, as will be shown in this appendix.

The birthday problem is the following well known problem in probability theory: under the assumption that the birthdays of n people are distributed independently over the year (365 days), how large should n be in order to have a reasonable large probability (say 0.5) to find two people with the same birthday? The probability that all persons have a different birthday is given by

$$\frac{1}{365^n} \cdot \prod_{i=0}^{n-1} (365 - i)\,.$$

The complement of this probability is the probability that two or more people have the same birthday. For $n = 23$ this is equal to 0.507. This has been called the "birthday paradox" because this number is much smaller than most people would intuitively think.

B.2 Models for matching probabilities

Several models can be considered in matching problems. Every model yields different probabilities and corresponds to different problems in cryptographic applications [235]. In probability theory, matching problems are mostly described in terms of urns and balls: r balls are randomly thrown into n urns and one studies the number of urns containing a certain number of balls (problem 1). An alternative formulation is that an urn contains n balls numbered from 1 to n. Subsequently r balls are drawn, and their numbers are listed. Now one studies the distribution of the numbers in this list (problem 2). One is mostly interested in two distributions, that are the same for both problems.

- The probability distribution of the number t of urns containing one or more balls (problem 1), or of the number t of balls that is listed at least once (problem 2). This allows to determine the number of *coincidences*, defined as $r - t$. In this model an urn that contains k balls (problem 1), or a number that is listed k times (problem 2) will correspond to $k - 1$ coincidences.

- The probability distribution of the number of urns containing k or more balls (problem 1), or of the number of balls that is listed k or more times (problem 2). In both models this will be called a *k-fold collision*.

In cryptography, most attention has been paid to the first approach [56, 123, 235]. However, in this thesis it was shown that results on the second problem are useful as well (cf. chapter 5).

A second distinction that can be made is based on the fact that there are balls of one or two colors, and in the latter case whether drawings are made with or without replacements.

- In case of the birthday problem, there is only one set in which one looks for a match. This corresponds to the problem that was described above. In cryptographic applications, this model is valid in algorithms for finding key collisions (cf. section 2.5.2.6). It is also valid if an attacker wants to produce a collision for a hash function without any restriction on the messages. Note that in this case drawings are necessarily made with replacements.

- In most cryptographic attacks, one looks for a match between two sets. In terms of urns and balls, this means that there are r_1 white and r_2 red balls, and one studies the probability distribution of the number of urns that contain two or more balls of a different color. For each of the balls, one can select the urns with and without replacements (i.e., the same urn can be chosen only once). It is more natural to formulate this in terms of problem 2. One has two urns, the first one contains n white balls numbered from 1 to n, and the second one contains n red balls numbered from 1 to n. First one selects r_1 balls from the first urn, and records their numbers. Subsequently one draws r_2 balls from the second urn, and records their numbers. The number of coincidences between the two lists of

numbers is now counted. One has two ways to select the balls: one can replace the ball after every drawing, or put it aside. This corresponds to drawing with and drawing without replacements. Unless it is explicitly stated, the terminology of problem 2 will be used in the rest of this appendix.

Model A: balls from both urns are drawn with replacements. Here one could make a distinction between coincidences and k-fold collisions, but the latter concept is rather complicated in this context (a combination of a k_1-fold collision of white balls with a k_2-fold collision of red balls with $k = k_1 + k_2$). A example of an application in cryptography is a birthday attack on a hash function with two distinct sets of messages (cf. section 2.5.1.3). Model A holds if the round function is not bijective for a fixed chaining variable.

Model B: balls from both urns are drawn without replacements. In this case it makes only sense to speak about coincidences, as a coincidence can only be found between balls with a different color. An example is a cryptanalytic attack on a cryptosystem that is 'closed', i.e., a cryptosystem for which there exists for any pair of keys K_1, K_2 a key K such that

$$E(K, P) = E(K_2, E(K_1, P)), \quad \forall P\,.$$

An attacker who is given a plaintext ciphertext pair (P, C) with $C = E(K, P)$, will not look for K, but for a pair of keys K_1, K_2 that satisfy the relation above.

Model C: this is a mixed model, where balls from one urn are drawn with replacements and balls from the other urn are drawn without replacements. In this case the extension to k-fold collisions would be easier. The application that is indicated in [235], namely producing 'sensible' collisions for a hash function based on CBC mode (cf. section 5.3.1.1) is rather artificial. A more interesting application is the time memory trade-off [147], where a set of r_1 values will be stored (in order to avoid waste of storage space, one will delete matches in the stored set), and compared afterwards to a set of r_2 values, that are drawn with replacements.

In the rest of this appendix an overview of known results on distributions for coincidences will be given, followed by a detailed discussion of the distribution of k-fold collisions for drawings from a single urn.

B.3 Coincidences

A detailed discussion of the distribution of the number of coincidences for the four models has been given in [235]. The results are summarized below.

For drawings of balls from a single urn with replacements, the probability that t different balls have been drawn (or that there are t urns containing one or more balls

in problem 1) is given by the classical occupancy distribution [134]:

$$\Pr(n, r, t) = \left\{ {r \atop t} \right\} \frac{n^{(t)}}{n^r}, \qquad 1 \le t \le r\,. \tag{B.1}$$

Here $n^{(r)} = n(n-1)\cdots(n-r+1)$ and $\left\{ {r \atop t} \right\}$ denotes the Stirling number of the second kind, defined by the following polynomial identity

$$x^r = \sum_{t=1}^{r} \left\{ {r \atop t} \right\} x^{(t)}\,. \tag{B.2}$$

The probability of a coincidence is given by

$$\Pr(n, r, t < n) = 1 - \Pr(n, r, n) = 1 - \frac{n^{(r)}}{n^r}\,. \tag{B.3}$$

For large n, and $r = O(\sqrt{n})$, this can be approximated by

$$1 - \exp\left(-\frac{r(r-1)}{2n}\right) \approx 1 - \exp\left(-\frac{r^2}{2n}\right)\,. \tag{B.4}$$

For model A, the probability of no coincidence is given by

$$\Pr_A(n, r_1, r_2, 0) = 1 - \frac{1}{n^{r_1+r_2}} \sum_{\nu=r_1+r_2} n^{(\nu)} \left\{ {r_1 \atop t_1} \right\} \left\{ {r_2 \atop t_2} \right\}\,. \tag{B.5}$$

For large values of n and if $r = r_1 = r_2$ with $r = O(\sqrt{n})$ this can be approximated by

$$1 - \exp\left(-\frac{r^2}{n}\right)\,. \tag{B.6}$$

For model B, we have a hypergeometric distribution for the number of coincidences

$$\Pr_B(n, r_1, r_2, t) = \binom{r_1}{t}\binom{n-r_1}{r_2-t} \Big/ \binom{n}{r_2}\,, \tag{B.7}$$

where $\max(0, r_1 + r_2 - n) \le t \le \min(r_1, r_2)$. For $t = 0$ this reduces to

$$\Pr_B(n, r_1, r_2, 0) = \frac{n^{(r_1+r_2)}}{n^{(r_1)} n^{(r_2)}}\,. \tag{B.8}$$

For large values of n and if r_1 and r_2 are $O(\sqrt{n})$ the probability of a coincidence can be approximated by

$$1 - \exp\left(-\frac{r_1 r_2}{n}\left[1 + \frac{r_1 + r_2 - 1}{2n}\right]\right)\,. \tag{B.9}$$

It is clear that if $r_1 = r_2 = r$, this is very close to the asymptotic expression for model A given by (B.6). For model C the occupancy distribution and the hypergeometric distribution are mixed (r_1 balls are drawn with replacements, r_2 balls are drawn without replacements):

$$\Pr_C(n, r_1, r_2, t) = \sum_s \binom{s}{t} \binom{n-s}{r_2-t} \left\{ {r_1 \atop s} \right\} \frac{n^{(s)}}{n^{(r_1)}} \Big/ \binom{n}{r_2} \; . \tag{B.10}$$

For $t = 0$ this corresponds to

$$\Pr_C(n, r_1, r_2, 0) = \left(1 - \frac{r_2}{n}\right)^{r_1} \; . \tag{B.11}$$

For large values of n and if r_1 and r_2 are $O(\sqrt{n})$ the probability of a coincidence can be approximated by

$$1 - \exp\left(-\frac{r_1 r_2}{n}\right) \; . \tag{B.12}$$

This result can also be found in Appendix B of the book of C. Meyer and S. Matyas [215].

In [123] it has been shown that for the first three models (not for model C) the probability distribution converges under certain conditions asymptotically to a Poisson distribution, and expressions for the error terms have been given. It was also proven that the probability distribution of the number of coincidences converges to

$$e^{-\lambda} \frac{\lambda^x}{x!} \, ,$$

where $\lambda = \frac{r^2}{2n}$, $\frac{r^2}{2n}$, and $\frac{r_1 r_2}{n}$ respectively. An interesting consequence is that for large values of r and n, the expected number of coincidences is equal to λ. If cryptographic schemes are evaluated, one has mostly that $r = O(\sqrt{n})$, with n of the order of 2^{56} or even larger. One can conclude that in this case, the asymptotic results are valid, and there is not much difference between the four models: if drawings are made from a single urn, the number of trials has to be a factor of about $\sqrt{2}$ larger than in the other cases to obtain the same success probability.

B.4 k-fold collisions

The number of k-fold collisions will now be studied for drawing with replacements within a single set. This result was shown by R. von Mises in 1939, but we were not able to find the paper containing the proof. Feller ([106], pp. 101–106) gives the asymptotic expression, but proves only the complete expression for $k = 0$ and studies its asymptotic behavior.

Theorem B.1 *If r balls are randomly distributed over n urns, the number t of urns containing exactly k-balls is given by*

$$\binom{n}{t} \frac{r^{(tk)}}{(k!)^t} \frac{(1 - \frac{t}{n})^{r-tk}}{n^{tk}} \sum_{\nu=0}^{n-t} (-1)^\nu \binom{n-t}{\nu} \frac{(r-tk)^{(\nu k)}}{(k!)^\nu n^{\nu k}} \frac{\left(1 - \frac{\nu}{n-t}\right)^{r-(\nu+t)k}}{\left(1 - \frac{t}{n}\right)^{\nu k}} \, . \tag{B.13}$$

Proof: We start with calculating the probability that a specific urn contains k balls:

$$p_1 = \frac{\binom{r}{k}(n-1)^{r-k}}{n^r}\,.$$

Here $\binom{r}{k}$ corresponds to the selection of k balls and $(n-1)^{r-k}$ corresponds to the different ways in which the remaining $r-k$ balls can be distributed over the remaining $n-1$ urns. The total number of distributions is given by n^r. Similarly, for two cells this probability is equal to

$$p_2 = \frac{\binom{r}{k}\binom{r-k}{k}(n-2)^{r-2k}}{n^r}\,.$$

The product of the two binomial coefficients can be simplified to

$$\frac{r!}{(k!)^2(r-2k)!} = \frac{r^{(2k)}}{(k!)^2}\,.$$

The general expression for $\nu \leq n$ is then

$$p_\nu = \frac{r^{(\nu k)}}{(k!)^\nu}\frac{(1-\frac{\nu}{n})^{r-\nu k}}{n^{\nu k}}\,. \tag{B.14}$$

Note that $p_n = 0$. The probability that ν cells contain k balls is given by

$$S_\nu = \binom{n}{\nu} p_\nu\,, \tag{B.15}$$

as there are $\binom{n}{\nu}$ ways to select ν cells out of n. The next step is to calculate the probability that no cell contains exactly k balls. This can be done with the inclusion-exclusion principle:

$$Pr(n,r,k,0) = \sum_{\nu=0}^{n-1}(-1)^\nu S_\nu\,. \tag{B.16}$$

Consider now a distribution where t cells contain exactly k balls. These t cells can be chosen in $\binom{n}{t}$ ways and the balls in these t cells can be chosen in $r^{(tk)}/(k!)^t$ ways. The remaining $r-tk$ balls are distributed over the remaining cells so that none of these cells contains k balls; the number of such distributions is $(n-t)^{r-tk}\Pr(n-t,r-tk,k,0)$. Dividing by n^r on obtains for the probability that exactly t cells contain k balls

$$\begin{aligned}
&\Pr(n,r,k,t)\\
&= \binom{n}{t}\frac{r^{(tk)}}{(k!)^t}(n-t)^{r-tk}\Pr(n-t,r-tk,k,0)/n^r\\
&= \binom{n}{t}\frac{r^{(tk)}}{(k!)^t}\,\frac{(1-\frac{t}{n})^{r-tk}}{n^{tk}}\,\sum_{\nu=0}^{n-t-1}(-1)^\nu\binom{n-t}{\nu}\frac{(r-tk)^{(\nu k)}}{(k!)^\nu n^{\nu k}}\,\frac{\left(1-\frac{\nu}{n-t}\right)^{r-(\nu+t)k}}{(1-\frac{t}{n})^{\nu k}}\,.
\end{aligned}$$

This completes the proof. ∎

The probability distribution of the number of coincidences can be obtained from $\Pr(n, r, 0, t)$. Indeed, if one knows that t urns are empty, the number of coincidences is given by $c = r - n + t$. With the substitution $\alpha = r - c - \nu$ one finds for the number of coincidences

$$Q(n, r, c) = \frac{\binom{n}{r-c}}{n^r} \sum_{\alpha=1}^{r-c} (-1)^{r-c-\alpha} \binom{r-c}{\alpha} \alpha^r \,.$$

This expression was also given in [123], but with an error in the exponent of (-1).

As it is not possible to evaluate (B.13) for large values of r and n, the asymptotic behavior of this equation will be studied. This will be done for the case $n \to \infty$ and $r \to \infty$, but under certain constraints. They will be discussed in the terminology of problem 1.

- If $k > 1$ one has the following: if r/n is too small, then we can expect no cells containing k balls; in this case $\Pr(n, r, k, 0)$ is near unity and all $\Pr(n, r, k, t)$ with $t \geq 1$ are small. On the other hand, if r/n is excessively large, most cells will contain about $k = r/n$ balls; this case will be considered later.
- If $k \leq 1$ the following occurs: if r/n is too large, then no cells will contain 0 or 1 balls; in this case $\Pr(n, r, k, 0)$ is near unity and all $\Pr(n, r, k, t)$ with $t \geq 1$ are small. On the other hand if r/n is very small, then nearly all cells will contain 0 or 1 balls; in this case $\Pr(n, r, 0, t)$ will be near unity for $t = (n - r)/n$, and zero elsewhere, and $\Pr(n, r, 1, t)$ will be near unity for $t = r/n$, and zero elsewhere.

Therefore we will discuss only the intermediate case.

Theorem B.2 *If r balls are randomly distributed over n urns, the number t of urns containing exactly k balls follows asymptotically a Poisson distribution, with*

$$\Pr(n, r, k, t) = e^{-\lambda_k} \frac{\lambda_k^t}{t!} \quad \textit{and} \quad \lambda_k = n \frac{\exp\left(-\frac{r}{n}\right)}{k!} \left(\frac{r}{n}\right)^k . \tag{B.17}$$

This holds if r and n tend to infinity such that λ_k remains bounded.

Proof: The first part consists in estimating the quantity S_ν of (B.15). Based on the inequality $x^{(s)} \leq x^s$ for $s \geq 1$ one obtains

$$\nu!\, S_\nu \leq \frac{n^\nu}{n^{\nu k}} \frac{r^{\nu k}}{(k!)^\nu} \left(1 - \frac{\nu}{n}\right)^{r - \nu k} .$$

For $0 < x < 1$ one finds from the Taylor expansion that

$$\frac{-t}{1-t} < \ln(1-t) < -t \,.$$

Therefore

$$\nu!\, S_\nu < \frac{n^\nu}{(k!)^\nu} \left(\frac{r}{n}\right)^{\nu k} \exp\left(-\frac{r - \nu k}{n} \nu\right) .$$

In order to obtain a lower bound one uses the inequality $x^{(s)} \geq (x-s)^s$ for $s \geq 1$:

$$\nu!\, S_\nu \geq \frac{(n-\nu)^\nu}{n^{\nu k}} \frac{(r-\nu k)^{\nu k}}{(k!)^\nu} \left(1 - \frac{\nu}{n}\right)^{r-\nu k} .$$

The lower bound for the logarithm yields

$$\nu!\, S_\nu > \frac{n^\nu}{(k!)^\nu} \left(\frac{r}{n}\right)^{\nu k} \left(1 - \frac{\nu k}{r}\right)^{\nu k} \exp\left(-\frac{r-\nu(k-1)}{n-\nu}\nu\right) .$$

Now define

$$\lambda_k = n \frac{\exp\left(-\frac{r}{n}\right)}{k!} \left(\frac{r}{n}\right)^k ,$$

and suppose that r and n increase in such a way that λ_k remains constrained to a finite interval $0 < a < \lambda_k < b$. For each fixed ν the ratio of upper and lower bound then tends to unity, under the condition that $\nu k \ll r$. Hence

$$0 \leq \frac{\lambda_k^n}{\nu!} - S_\nu \to 0 . \tag{B.18}$$

This relation holds trivially when $\lambda_k \to 0$, and hence (B.18) remains true whenever r and n increase in such a way that λ_k remains bounded. Now

$$e^{-\lambda_k} - \Pr(n, r, k, 0) = \sum_{\nu=0}^{\infty} (-1)^\nu \left(\frac{\lambda_k^\nu}{\nu!} - S_\nu\right) \tag{B.19}$$

and (B.18) implies that the right side tends to zero. The observation that (B.13) can be rewritten as $S_t \cdot \Pr(n, r, k, 0)$ shows that for each fixed t

$$\Pr(n, r, k, t) - e^{-\lambda_k} \frac{\lambda_k^t}{t!} \to 0 .$$

This completes the proof. ∎

As a corollary of this theorem, one can easily derive the asymptotic probability for the case that the number of coincidences are relevant: it can be verified that

$$\lambda^\infty = \sum_{k=2}^{\infty} \lambda_k = n \left[1 - \left(1 + \frac{r}{n}\right) \exp\left(-\frac{r}{n}\right)\right] . \tag{B.20}$$

A Taylor expansion up to third order terms results in the familiar

$$\lambda^\infty \approx \frac{r^2}{2n} \left[1 - \frac{2r}{3n}\right] \approx \frac{r^2}{2n} . \tag{B.21}$$

This first approximation explains more exactly the behavior for values of n between 100 and 625 as given in [123].

The probability that exactly one k-fold collision occurs is $\lambda_k \exp(-\lambda_k)$, and the probability that at least one k-fold collision occurs is given by $1 - \exp(-\lambda_k)$. If one

k	$\log_2(r)$	$\log_2(\tilde{r})$	λ_{k+1}
2	32.50	32.50	$1.10 \cdot 10^{-10}$
3	43.53	43.53	$1.72 \cdot 10^{-7}$
4	49.15	49.15	$6.76 \cdot 10^{-6}$
5	52.58	52.58	$6.09 \cdot 10^{-5}$
6	54.92	54.92	$2.63 \cdot 10^{-4}$
7	56.62	56.61	$7.48 \cdot 10^{-4}$
8	57.92	57.91	$1.64 \cdot 10^{-3}$
16	62.81	62.77	$2.57 \cdot 10^{-2}$
20	63.92	63.85	$4.51 \cdot 10^{-2}$
24	64.73	64.63	$6.62 \cdot 10^{-2}$
28	65.34	65.21	$8.75 \cdot 10^{-2}$
32	65.84	65.68	$1.08 \cdot 10^{-1}$
48	67.16	66.89	$1.83 \cdot 10^{-1}$
64	67.98	67.62	$2.43 \cdot 10^{-1}$
96	69.00	68.52	$3.31 \cdot 10^{-1}$
128	69.67	69.10	$3.94 \cdot 10^{-1}$

Table B.1: Numerical solutions for the number of trials r such that $\lambda_k = 1$ for $n = 2^{64}$. Also indicated are an approximation $\tilde{r}$ and λ_{k+1}.

wants to calculate the number of trials in order to have at least one k-fold collision with probability $1 - e^{-1} \approx 0.63$, one has to solve numerically the equation $\lambda_k = 1$ or

$$r \exp\left(-\frac{r}{nk}\right) = n^{\frac{k-1}{k}} (k!)^{\frac{1}{k}} .$$

For the case $n = 2^{64}$, and $2 \leq k \leq 64$ the values of r are indicated in table B.1. If $r \leq n$ a good approximation can be obtained as follows:

$$\tilde{r} = n^{\frac{k-1}{k}} (k!)^{\frac{1}{k}} .$$

This approximation has also been given in table B.1. For the case $r = n$, $\lambda_k = ne/k!$, table B.2 indicates what the largest value of k is that corresponds to $\lambda_k \geq 1$ for a given value of n.

However, in cryptographic applications one is in fact interested in the probability that at least one l-fold collision occurs, for some $l \geq k$. The probability that at least one $(k+1)$-fold collision occurs is given by $1 - \exp(-\lambda_{k+1})$. From the relation

$$\lambda_{k+1} = \frac{r}{n(k+1)} \lambda_k ,$$

it follows that if r is chosen such that $\lambda_k = 1$, the probability of a $k+1$-fold match is much smaller. For $n = 2^{64}$ it can be seen from table B.1 that λ_{k+1} is smaller than

$\log_2(n)$	k
16	7
32	12
48	16
64	20
80	23
96	27

Table B.2: Largest value of k for which $\lambda_k \geq 1$ if $r = n$.

about 0.1 if $k < 32$. It is clear that for larger values of k the probabilities for a $k + i$-fold match with $i > 1$ will play a more important role. However, if $r \gg n$, a different approximation can be used as will be shown in the following paragraph.

If a large number of drawings has been made, the law of large numbers specifies in terms of problem 1 that the number of balls in every urn will approximate r/n, and the number of urns containing k balls will be distributed according to a Gaussian distribution with mean $\mu = r/n$ and standard deviation $\sigma = \sqrt{r/n}$ [106]. The substitution $x = (y - \mu)/\sigma$ results in the normalized Gaussian distribution:

$$G(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) .$$

In this case it is possible to determine the number of urns that contain k or more balls from the cumulated normal distribution. If $Q(z)$ is defined as

$$Q(z) = \int_z^\infty G(x)\, dx\,,$$

this probability is equal to $Q((k - \mu)/\sigma)$. If z is large, $Q(z)$ can be approximated by

$$Q(z) \approx \frac{1}{z\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) .$$

Hence if r trials are made, with $r \gg n$, the probability that there is an urn with more than

$$\frac{r}{n} + z\sqrt{\frac{r}{n}}$$

balls is equal to $Q(z)$. With the definitions $r = 2^{r'}$, $n = 2^{n'}$, $k = 2^{k'}$, $z = 2^{z'}$ and $r' = n' + k' - x'$, one can rephrase this as follows: if $2^{r'} = 2^{n'+k'-x'}$ trials are made, the probability that there is an urn with more than $2^{k'}$ balls is equal to $Q(z)$ if

$$2^{k'} = 2^{k'-x'} + 2^{z'+(k'-x')/2}\,. \tag{B.22}$$

Note that this relation is independent of n'. It can be shown that the solution of (B.22) is given by

$$x' = 2\log_2\left(\sqrt{2^{2\cdot(z'-1-k'/2)}+1} - 2^{z'-1-k'/2}\right). \tag{B.23}$$

For large values of k' this can be approximated by $x' = -2^{z'-k'/2}$. From the asymptotic expressions for $Q(z)$ one obtains that $Q(8) = 2^{-50.5}$ and $Q(16) = 2^{-190.0}$. Therefore the choice of $z = 16$ or $z' = 4$ yields a safe lower bound for the number of trials. Table B.3 indicates the relation between k' and x' for $z' = 4$. It can be seen from (B.23) that replacing z' by $z' + \delta'_z$ yields the value of x' corresponding to $k' + 2\delta'_z$. We have selected a sufficient security margin because for smaller values of k' ($k' \leq 10$) this can compensate for the errors that may have been introduced by replacing the distribution by a normal distribution. For larger values of k' the table entries tend to 0, which means that the choice of a different value of z would make not much difference. One can conclude from this table that for $k' \geq 16$ the number of operations to produce a $2^{k'}$-fold collision is given by $2^{n'+k'}$. For $k \geq 10$ this formula gives a good approximation.

k'	x'	k'	x'
4	4.165	12	0.507
5	3.307	13	0.360
6	2.543	14	0.180
7	1.900	15	0.127
8	1.338	16	0.090
9	1.000	32	$3.52 \cdot 10^{-4}$
10	0.714	64	$5.35 \cdot 10^{-9}$

Table B.3: Values of k' and x' for the equation (B.23) with $z' = 4$.

A combination of the Poisson and the Gaussian approximation was used to produce figure B.1. It gives the relation between the number of trials r and k for $n = 2^{48}$ and $n = 2^{64}$. The approximation by the normal distribution was made with a value of $z = 16$. For small values of $k' = \log_2(k)$, only a limited number of points are present, corresponding to the discrete values $k = 2, 3, \ldots$ The number of operations for a k-fold collision increases very fast with k in this area. For $k' > 15$ one notes that the behavior is linear because of the logarithmic scales.

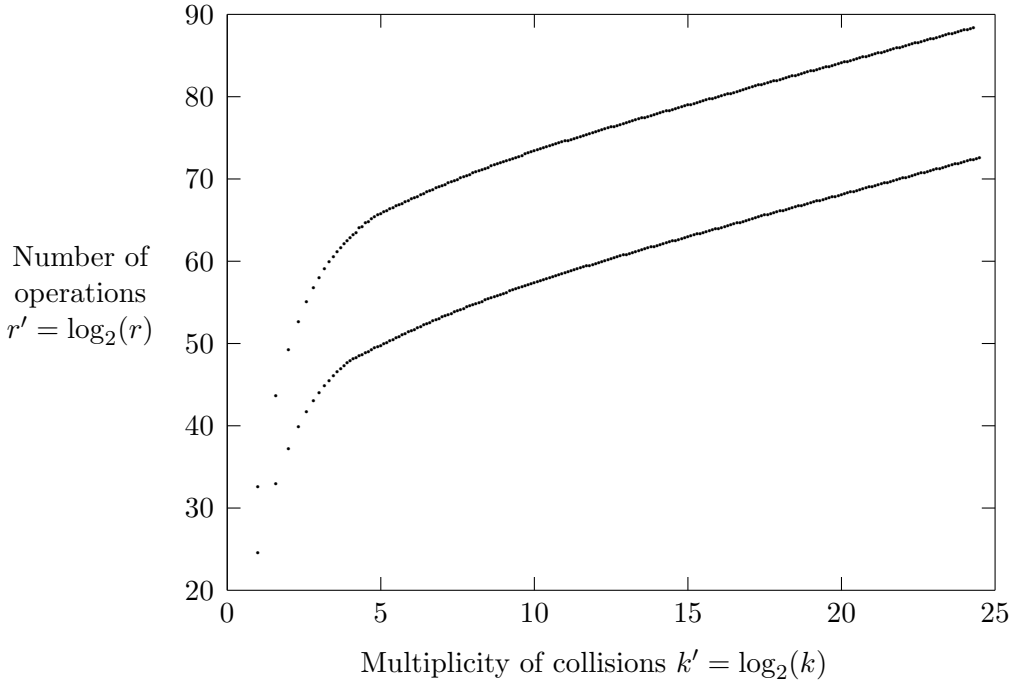

Figure B.1: Binary logarithm of the number of trials r to obtain a k-fold collision for $n = 2^{48}$ and $n = 2^{64}$.

Appendix C

Differential Cryptanalysis of Hash Functions Based on Block Ciphers

When there is no difference, there is only indifference. *Louis Nizer*

C.1 Introduction

This appendix describes in more detail a differential attack on several hash functions based on a block cipher. It will be mainly concerned with results for DES [108] as the underlying block cipher, but the case of FEAL-N [225, 228] will also be discussed. Differential cryptanalysis, proposed by E. Biham and A. Shamir, is a very powerful technique for the cryptanalysis of both block ciphers and hash functions (cfr. section 2.5.2.7). It will be assumed that the reader is familiar with the main principles and results of differential cryptanalysis as described in [20, 21, 22, 23, 182].

The use of differential attacks on hash functions based on block ciphers was suggested to the author by I. Damgård [67]. Similar ideas were developed by E. Biham [24], without studying a detailed optimization.

The basic idea of the attack is to look for a characteristic of the underlying block cipher for which input exor equals output exor. It is assumed that the attacker has control over the plaintext input. Because of the feed forward that is present in all secure hash functions based on block ciphers, these two exors will cancel out in the result and will produce two blocks with the same hash value.

Differential cryptanalysis of a block cipher and of a hash function based on that block cipher are closely related but differ in some aspects. In the first place, the goal is not to find a key, but to find a single right pair, i.e., a pair that follows the

characteristic. Note that the most recent attack on DES as a block cipher requires only one right pair as well [23]. The number of rounds of the characteristic and of the hash function have to be exactly the same, while for the attack on a block cipher the characteristic can be 1, 2, or 3 rounds shorter, which increases its probability. The fact that the key is known can be exploited to reduce the effort to find a right pair. The main difference is that the attack can be performed off-line and in parallel. This implies that even if the number of computations would be 2^{47}, the attack would be practical, which is not the case for an attack requiring 2^{47} chosen plaintexts.

In a first section, an analysis will be made for scheme 1–4 of section 5.3.1.4 that yield a single length hash function. Then the differential attacks on MDC-2 and MDC-4 will be treated. Finally the security of these schemes with FEAL-N as block cipher will be discussed, and the conclusion will be presented.

C.2 Differential attacks on single length hash functions

The attack on this system will look for two plaintext blocks X and $X' \neq X$ that satisfy

$$E(K, X) \oplus X = E(K, X') \oplus X'$$

for a given DES key K. The difference between the X and X' will be chosen in such a way that the probability that this equation holds is maximized. If one is looking for a collision, one can freely choose X and X'. However, the same principle could be used to find a second preimage for a given X. One only has a very limited number of trials, but one hopes that the success probability of a single trial will be better than 2^{-56}.

For DES, the best known characteristics with input exor equal to output exor for 7 rounds or more are iterative characteristics for an odd number of rounds [21]. This implies that the attack will be limited to an odd number of rounds. In this section it will be explained how knowledge of the key can reduce the effort to find a right pair.

For DES, the two best iterative characteristics modify only the inputs of the first 3 S-boxes. They will be indicated with ψ, for which the input exor to the S-boxes equals $\texttt{19 60 00 00}_\texttt{x}$, and ϕ, for which this exor equals $\texttt{1B 60 00 00}_\texttt{x}$. The probability of both characteristics for 3 rounds equals $p_\psi = p_\phi = 14/64 \cdot 8/64 \cdot 10/64 \approx 1/234$. However, this does not tell the complete story: if key bit 6 of S2 is different from key bit 2 of S6 (because of the expansion E they are added to the same bit of the plaintext), the probability of ϕ is $7/1024 \approx 1/146$, while the probability of ψ equals $7/4096 \approx 1/585$. If both key bits are equal the probabilities will be interchanged. This means that for $2t + 1$ rounds, one can always find a characteristic with probability [20], p. 42:

$$\max\left(\frac{1}{2^{2n}}, \frac{1}{2^{2t-2n}}\right)\left(\frac{7}{2^{10}}\right)^t.$$

Here n, $0 \leq n \leq t$ is the number of rounds in which the indicated key bits are equal. The probability of the characteristic will be maximal and equal to $(7/1024)^t$ if all key bits are different or if all are equal. This will be the case for 1 key out of 64. As the key is known in our case, the probability of success can be increased by attacking hash

values where such a key is used in the hashing process. For a 15 round characteristic, a probability of $2^{-50.3}$ can be obtained if the following key bits are all different or all equal:

$$(11, 60), (44, 57), (41, 25), (9, 58), (50, 34), (18, 2), (51, 35).$$

An overview of the characteristics with the largest probability is given in table C.1.

number of rounds	$-\log_2(p)$
7	21.6
9	28.8
11	36.0
13	43.2
15	50.3

Table C.1: The probabilities of the best known characteristics for DES for which input exor equals output exor.

The second way in which knowledge of the key can be used is to choose the input bits to the first three S-boxes in such a way that the characteristic will have a higher probability. For the second round it is possible to take inputs for which the characteristic will be satisfied with probability 1. This can be done by selecting the inputs that go to the first three S-boxes from the 112 right pairs that satisfy the characteristic. This means that only a 13-round characteristic is required to attack the 15-round hash function. For the fourth round this is much more complicated: the design properties of DES make it impossible to predict actual values of the inputs to the S-boxes (note that it is possible to predict input differences). A more sophisticated approach could start in the middle of the algorithm: in this case it might be possible to eliminate two additional rounds, but more work has to be done on this approach.

A third way in which advantage can be taken from knowledge of the key is that wrong pairs can be discarded faster: every round one can check whether the characteristic is satisfied or not, which means that in most cases only 4 rounds will have to be calculated in stead of 15 rounds. This makes the attack about four times faster.

Note that the first two methods can not help an attacker directly when he tries to construct a second preimage. What he can do to increase his success probability is to limit his attack to inputs that satisfy additional constraints. For the number of operations to construct a collision and a second preimage for a DES-based single length hash functions with t rounds (t odd), the reader is referred to table 5.6. From this table it follows that finding a second preimage for 15 rounds can be made slightly more efficient than exhaustive search, and the advantage will be substantial (2^{14}) if the attacker can limit his attacks to optimal messages (in this case the number of operations for a collision and for a second preimage are equal).

One could try to optimize that attack by looking for other good characteristics for which input and output exor are equal. Even if the probability is slightly worse, one

could perform more trials in order to find a second preimage.

The main open problem is the extension of the attack to an even number of rounds. This would require more work on the construction of characteristics. A possibility is the use of fixed points. If χ is a fixed point with a certain probability, one could make use of the following characteristic: (χ, χ) (with $\chi \rightarrow 0$), (χ, χ) (with $\chi \rightarrow \chi$), the iterative characteristic χ for an odd number of rounds, (alternating $\chi \rightarrow 0$ with $0 \rightarrow 0$), and finally $(\chi, 0)$ (with $\chi \rightarrow \chi$) yielding an identical output and input exor. It should be possible to construct the inputs to the first two rounds such that the characteristic is satisfied with very high probability. The only problem would be the transition $\chi \rightarrow \chi$ in the last round. In [179] it is suggested that such a characteristic will have rather low probability, but further investigations are clearly necessary.

C.3 Differential attacks on MDC-2 and MDC-4

For a differential attack on MDC-2, the input pair has to be a right pair for both characteristics. Essentially all properties can be transferred, but the overall probability of success will be the product of the probabilities, under the reasonable assumption that both key values are independent. In order to maximize both probabilities, it is necessary that both keys satisfy the constraints discussed in the previous section, which means that only 1 value in 4096 will be attacked in case of 15 rounds. From the definition of MDC-2, it follows that a collision for the basic compression function of MDC-2 with identical chaining variables yields a similar collision for MDC-4 as well. Hence a differential attack on MDC-4 is not harder than a differential attack on MDC-2.

The only step where the approach is more complicated is the second round. Here a careful selection of the inputs is necessary to produce two inputs that yield a right pair for both key values with probability 1. Let K_i and K_i^* be the key bits in the ith round of the first and second DES respectively. The values that enter the S-boxes in the second round are $b = L \oplus K_2 \oplus F(K_1, R)$ and $b^* = L \oplus K_2^* \oplus F(K_1^*, R)$. It is required that the first 18 bits of b and b^* both yield a right pair. This can be obtained as follows: look for an R such that the first 18 bits of $b \oplus b^*$ (this is independent of L) correspond to the difference between two right pairs. Subsequently, L can be chosen such that both b and b^* yield a right pair. The number of possible differences between right pairs is expected to be 112^2, which means that about $2^{18+1}/112^2 \approx 42$ trials will be necessary. If one wants to estimate the exact number of trials, the distribution of these differences should be verified. Note that the fact that the second and third bits of the keys are different (this is imposed in the definition of the scheme) excludes a trivial solution, namely the case where the relevant bits of (K_1, K_2) and (K_1^*, K_2^*) would be equal.

One can conclude that it is possible to deal with the first two rounds, resulting in the figures indicated in table 5.9. The probabilities of the characteristics have to be multiplied, and the overall work is increased by a factor 2 since there are two chains. The conclusion is that for only MDC-2 or MDC-4 with 11 rounds or less this attack

is faster than a birthday attack. In order to be of the same order of magnitude as a birthday attack for 15 rounds, a characteristic with probability of 2^{-32} or better would be required (this would be 1/40 per two rounds for an iterative characteristic). For the second preimage attack the situation is slightly different, as finding a second preimage for MDC-4 is harder. In case of MDC-2, the differential attack is only faster for 11 rounds or less, while for MDC-4 it is faster for 15 rounds or less.

Note that the initial values that are proposed by the designers ($2525252525252525_{\mathtt{x}}$ and $5252525252525252_{\mathtt{x}}$) have 3 different and 4 equal key bits and 4 different and 3 equal bits respectively. This means that in this case both ψ and ϕ have a probability that is a factor 128 smaller for both chains together. In a practical attack, the attacker will calculate the chaining variables for about 4096 random message blocks. He expects then to find a chaining variable that brings him in the optimal situation.

C.4 Differential attacks on FEAL-N based hash functions

Differential attacks on hash functions based on the block cipher FEAL-N [225, 228] are more effective. The reason is that for FEAL-N there exists a 4-round iterative characteristic with relatively high probability [21]. Let $\psi = \mathtt{80\,60\,80\,00_x}$ and let $\phi = \mathtt{80\,E0\,80\,00_x}$, then the following pattern of exors is obtained at the beginning of every round:

$$(\psi, \psi) \longrightarrow (\psi, \phi) \longrightarrow (\phi, \phi) \longrightarrow (\phi, \psi)\,.$$

The corresponding output exor is then equal to the input exor. All four transitions have probability 1/4, which yields an overall probability of 1/256. The actual input and output exor pattern is equal to $(\psi, 0)$, as the left part is added modulo 2 to the right part at the beginning and at the end of the encryption. Note that this attack assumes that N is a multiple of 4.

Knowledge of the key can speed up the search for a right pair in two ways: the plaintext can be chosen such that the characteristic is always satisfied in the first round, and an early abort strategy can be used. Note that in case of FEAL-N there are no keys that increase the probability of the characteristic.

- The characteristic will be satisfied in the first round if an input exor of $\mathtt{E0_x}$ and $\mathtt{80_x}$ of the second S-box yield an output exor $\mathtt{80_x}$. There are exactly $2^{16}/4 = 16,384$ input pairs for which this holds. If the plaintext bytes are denoted with $P[0]$ through $P[8]$, and the round keys are numbered as in [21], the inputs $S2a$ and $S2b$ of this S-box can be written as follows:

$$S2a = P[0] \oplus P[1] \oplus P[4] \oplus P[5] \oplus K2a \tag{C.1a}$$
$$S2b = P[2] \oplus P[3] \oplus P[6] \oplus P[7] \oplus K2b\,, \tag{C.1b}$$

  with

$$K2a = K89[0] \oplus K89[1] \oplus Kab[0] \oplus Kab[1] \oplus K0[0]$$
$$K2b = K89[2] \oplus K89[3] \oplus Kab[2] \oplus Kab[3] \oplus K0[2]\,.$$

One possibility to obtain a right pair is to choose $S2a = S2b = \mathtt{A0_X}$. One can now solve $P[5]$ and $P[7]$ (or 2 other variables) from (C.1a) and (C.1b) respectively, such that the characteristic will be always satisfied in the first round.

- The average number of rounds that have to be computed can be reduced to 7/3 if after every round it is verified whether the pair is a right pair. This yields a speedup with a factor $3N/7$.

From this it follows that the number of full encryptions to find a right pair is about

$$\frac{2^{2N}}{2N}\,.$$

The number of operations to find a collision for a single length hash function based on FEAL-N is indicated in table C.2. For a second preimage the number of encryptions has to be multiplied by 4. A solution for N equal to 4, 8, 12, and 16 can also be found in table C.2. The scheme suggested in the Japanese ISO contribution [161] can be described as follows:

$$H_i = E(H_{i-1}, X_i) \oplus X_i \oplus H_{i-1}\,.$$

It corresponds to scheme 3 of section 5.3.1.4, and it will thus be vulnerable to a differential attack.

A second scheme that was described in this contribution is a variant on MDC-2. The only difference is that before the plaintext enters the second block cipher, the 8-byte constant A consisting of 8 bytes with the value $\mathtt{AA_X}$ is added modulo 2. Also both key and plaintext are added modulo 2 to the ciphertexts, both *after* the exchange:

$$\begin{aligned}
T1_i &= E(H1_{i-1}, X_i) = LT1_i \parallel RT1_i \\
T2_i &= E(H2_{i-1}, X_i \oplus A) = LT2_i \parallel RT2_i \\
V1_i &= LT1_i \parallel RT2_i \\
V2_i &= LT2_i \parallel RT1_i \\
H1_i &= V1_i \oplus X_i \oplus H1_{i-1} \\
H2_i &= V2_i \oplus X_i \oplus H1_{i-1}\,.
\end{aligned}$$

In this case one has to obtain a good pair for both keys. The exchange of the right halves has no influence on the result, as the same characteristic is used in both encryptions. A simple way to guarantee that X_i satisfies the characteristic in both encryptions is to restrict the attack to pairs $(H1_{i-1}, H2_{i-1})$ that yield the same value of $K2a$ and $K2b$ (this holds with a probability of $1/65,536$). It is clear that one could easily generalize the solution. The number of full encryptions is then approximately equal to

$$\frac{2^{4N}}{3N}\,.$$

The number of operations to find a collision for this double length hash function based on FEAL-N and a solution for N equal to 4 and 8 are given in table C.2.

type	N	# encryptions	Example of collisions	
			key(s)	plaintext
single length	4	2^5	$\texttt{F8F5D8605FF0F6F1}_\texttt{x}$	$\texttt{7326DA39FD2420BF}_\texttt{x}$
	8	2^{12}	$\texttt{9186875F60D26F00}_\texttt{x}$	$\texttt{87BE2AB529A5760F}_\texttt{x}$
	12	$2^{19.6}$	$\texttt{416827E2BF080FCD}_\texttt{x}$	$\texttt{DA24259F826B76D3}_\texttt{x}$
	16	2^{27}	$\texttt{BFABEE5EF51AAF65}_\texttt{x}$	$\texttt{7E3D20303BBA7B23}_\texttt{x}$
double length	4	$2^{12.2}$	$\texttt{E2033C42A611CCCF}_\texttt{x}$ $\texttt{BFD26A87CCE9EBD9}_\texttt{x}$	$\texttt{7BFE2B7075B74FBF}_\texttt{x}$
	8	$2^{27.2}$	$\texttt{2A6DE5B70AB4B489}_\texttt{x}$ $\texttt{F3BFA9B41BED3A46}_\texttt{x}$	$\texttt{185B91454CC4CB87}_\texttt{x}$
	12	$2^{42.6}$		
	16	$2^{58.2}$		

Table C.2: Number of operations to find a collision for hash functions based on FEAL-N and examples of collisions.

One can conclude from table C.2 that more than 16 rounds are required to make a differential attack less efficient than a birthday attack. If one imposes that a differential second preimage attack is slower than an exhaustive search, N has to be larger than 32.

C.5 Conclusion

Differential attacks on hash functions based on DES form no serious threat because no good characteristics are known with an even number of rounds. If the number of rounds of DES would be decreased by one, the only attack that would be more efficient than a random attack is the second preimage attack on the single length hash functions. MDC-2 and MDC-4 are resistant to differential attacks if the number of rounds is 13 or more. Further improvements might be expected on non-iterative characteristics with a high probability and on characteristics for an even number of rounds. If however a good characteristic exists with the same length as the block cipher, as is the case for FEAL-N, one has to increase the number of rounds in order to lower the probability of the characteristic. Contrary to a differential chosen or known plaintext attack on a block cipher, a differential attack on the hash function based on the same block cipher can be a *practical* attack.

Appendix D

The Number of Graphs with a Given Minimum Degree

Est quadam prodire tenus, si non datur ultra. *Horatius*

The purpose of this appendix is to determine the number of labeled undirected simple graphs with n vertices with minimum degree $d_{\min}$. This problem is interesting in its own right, but it is studied here because it allows to determine the number of quadratic Boolean functions of n variables satisfying $PC(1)$ of order m. Indeed, theorem 8.9 states that this number is equal to the number of labeled undirected simple graphs with n vertices with minimum degree $d_{\min}$ equal to $m+1$.

D.1 Definitions and basic properties

In this appendix, all definitions and results are restricted to labeled undirected simple graphs. For more general definitions the reader is referred to [139]. The degree of a vertex is equal to the number of edges incident to that vertex, and the minimum degree $d_{\min}$ of a graph is the minimum of the set consisting of the degrees of the vertices. Similarly, the maximum degree $d_{\max}$ of a graph is the maximum of this set. The number of graphs with n vertices and minimum (maximum) degree $d_{\min}$ ($d_{\max}$) will be denoted with $A_n^{d_{\min}}$ ($B_n^{d_{\max}}$). Determining $A_n^{d_{\min}}$ and $B_n^{d_{\max}}$ for general n and $d_{\min}$ seems to be a difficult problem [140]. By complementing the corresponding graphs, one obtains that

$$A_n^{d_{\min}} = B_n^{n-1-d_{\min}} .$$

Note that this does not necessarily holds if one considers the restriction to connected graphs.

Let a_k $(k = 1, 2, \ldots)$ be a series of numbers. Then the formal power series

$$A(x) = \sum_{k=1}^{\infty} a_k \frac{x^k}{k!}$$

is called the *exponential generating function* for the series a_k.

The number of labeled graphs with k vertices will be denoted with g_k and the number of connected graphs with k vertices will be denoted with g'_k. The corresponding exponential generating functions are then $G(x)$ and $G'(x)$ respectively. The following relation can be established between $G(x)$ and $G'(x)$ [140].

Theorem D.1 *The exponential generating functions $G(x)$ and $G'(x)$ for labeled graphs and labeled connected graphs satisfy the following relation*

$$1 + G(x) = e^{G'(x)}\,.$$

This theorem can be generalized [140]: if the exponential generating function for a class $H(x)$ of graphs is known, then the exponential generating function for the corresponding connected graphs $H'(x)$ can be obtained as the formal logarithm of the first series. The relation between the two series is then given by the following proposition.

Proposition D.1 *If $\sum_{k=0}^{\infty} h_k x^k = \exp\left[\sum_{k=0}^{\infty} h'_k x_k\right]$, then for $n \geq 1$*

$$h'_n = h_n - \frac{1}{n} \sum_{k=1}^{n-1} k\, h'_k h_{n-k}\,. \tag{D.1}$$

It is now straightforward to determine the number of connected graphs.

$$\frac{g'_n}{n!} = 2^{\Delta(n)} - \frac{1}{n} \sum_{k=1}^{n-1} k\, \frac{g_k}{k!}\, \frac{g'_{n-k}}{n-k!}\,,$$

with $\Delta(n) = \frac{n\cdot(n-1)}{2}$. This can be rewritten as

$$g'_n = 2^{\Delta(n)} - \sum_{k=1}^{n-1} \binom{n-1}{k-1} g'_k\, 2^{\Delta(n-k)}\,.$$

A second result that is required is about trees. A tree is a connected graph that has no cycles. The number of trees was determined by Cayley in 1889 [140].

Theorem D.2 (Cayley) *The number T_n of trees with n vertices is*

$$T_n = n^{n-2}\,. \tag{D.2}$$

The next result can be used to show that certain graphs are always connected.

Proposition D.2 *Let G be a graph with n vertices and with $d_{\min} \geq n-k$. A sufficient condition for G to be connected is that*

$$k \leq \left\lceil \frac{n}{2} \right\rceil . \tag{D.3}$$

Proof: The proof consists in showing that if G is not connected, then

$$d_{\min} < \left\lfloor \frac{n}{2} \right\rfloor .$$

If G is not connected, G will consist of at least 2 components. The minimum degree $d_{\min}$ of G will be maximal if G consists of 2 components $G1$ and $G2$. Indeed, if there are more than 2 components, $d_{\min}$ will not decrease if components are merged by adding vertices. Now

$$d_{\min} = \min \{d_{\min}(G_1), d_{\min}(G_2)\} < \left\lfloor \frac{n}{2} \right\rfloor .$$

This completes the proof. ∎

D.2 A solution for some values of the minimum degree

In this section a solution will be given for the following cases: $d_{\min} = 0, 1, 2, n-3, n-2$, and $n-1$. The cases $d_{\min} = 0$ and $d_{\min} = n-1$ are trivial, as they correspond to all graphs and the complete graph respectively. The cases $d_{\min} = 1$ and $d_{\min} = n-2$ are not too difficult to solve, but the cases $d_{\min} = 2$ and $d_{\min} = n-3$ require more complicated derivations. We believe that it might be possible to extend these techniques for $d_{\min} = 3$ and $d_{\min} = n-4$.

The case $d_{\min} = n-2$ corresponds to $d_{\max} = 1$. Hence one has to count the number of ways in which one can construct a graph with 0 through $\lfloor n/2 \rfloor$ edges, such that none of these are adjacent:

$$B_n^1 = \sum_{0 \leq 2i \leq n} \frac{1}{i!} \prod_{j=0}^{i-1} \binom{n-2j}{2} .$$

This can be simplified to

$$B_n^1 = \sum_{0 \leq 2i \leq n} \frac{n!}{(n-2i)!\; i!\; 2^i} .$$

The case $d_{\min} = 1$ corresponds to the number of graphs with no isolated points. The number of these graphs is given by the following recursive equation (for $n > 2$):

$$A_n^1 = 2^{\Delta(n)} - 1 - \sum_{k=2}^{n-1} \binom{n}{k} A_k^1 ,$$

with $A_2^1 = 1$. The solution of this equation is given by

$$A_n^1 = \sum_{k=1}^{n-1} (-1)^{n-k-1} \binom{n-1}{k} 2^{\Delta(k)} \left(2^k - 1\right).$$

This can be further simplified to

$$A_n^1 = 2^{\Delta(n)} \sum_{k=0}^{n} (-1)^k \binom{n}{k} 2^{-\frac{k(2n-k-1)}{2}}.$$

One can show that asymptotically

$$\frac{A_n^1}{2^{\Delta(n)}} = 1 - \frac{n}{2^{n-1}} + \frac{n(n-1)}{\left(2^{n-1}\right)^2} + O\left(\frac{n^3}{2^{3n}}\right).$$

In order to solve the case $d_{\min} = n-3$ or to determine A_n^{n-3}, one will first compute the number B_n^2 of graphs with $d_{\max} = 2$. The corresponding number ${B'}_n^2$ of connected graphs can be found directly. For $n = 1$ and $n = 2$, there is only 1 solution, and for $n \geq 3$ one finds $(n! + (n-1)!)/2$ graphs. The first term corresponds to the number of paths of length $n-1$, and the second terms to the number of cycles. Hence the following expression is valid for $n \geq 1$:

$${B'}_n^2 = \left\lfloor \frac{(n+1)!}{2n} \right\rfloor.$$

With equation (D.1) one obtains the following recursion formula

$$B_n^2 = \left\lfloor \frac{(n+1)!}{2n} \right\rfloor + \sum_{k=1}^{n-1} \binom{n-1}{k-1} \left\lfloor \frac{(k+1)!}{2k} \right\rfloor B_{n-k}^2. \tag{D.4}$$

From complementation of these graphs, one also has obtained A_n^{n-3}. Note that ${A'}_n^{n-3}$ is not equal to ${B'}_n^2$, as complementing a connected graph with $d_{\min} = n-3$ will not always yield a connected graph. For $n \leq 4$, the results for $d_{\min} = 1$ can be used, and for $n \geq 5$ it follows from proposition D.3 that all graphs with $d_{\min} = n-3$ are connected.

A more complicated derivation yields the number A_n^2 of graphs with $d_{\min} = 2$, or the graphs with no end points (and no isolated points). In [276] it is shown that the number ${A'}_n^2$ of connected graphs with $d_{\min} = 2$ satisfies the following equation:

$$\sum_{k=3}^{\infty} \frac{{A'}_k^2}{k!} x^k = \sum_{k=1}^{\infty} \frac{C_k}{k!} x^k e^{-kx}.$$

Here $C_n = g'_n - T_n$, is equal to the number of connected graphs that are not trees. From this recursion one obtains

$${A'}_n^2 = \sum_{k=3}^{n} (-1)^{n-k} \binom{n}{k} C_k \, n^{n-k}.$$

The corresponding number of graphs can then be found with equation (D.1):

$$A_n^2 = A'^2_n + \sum_{k=3}^{n-3} \binom{n-1}{k-1} A'^2_n A^2_{n-k} \,. \tag{D.5}$$

Numerical results for graphs and connected graphs with up to 10 vertices are collected in table D.1 and D.2 respectively. For small values of n, the upper index in A_n^{n-k} can become negative. In that case the convention $A_n^{-k} = A_n^0$ $(k > 0)$ was used.

n	A_n^0	A_n^1	A_n^2	A_n^{n-3}	A_n^{n-2}	A_n^{n-1}
1	1	0	0	1	1	1
2	2	1	0	2	2	1
3	8	4	1	8	4	1
4	64	41	10	41	10	1
5	1024	768	253	253	26	1
6	32768	27449	12068	1858	76	1
7	2097152	1887284	1052793	15796	232	1
8	268435456	252522481	169505868	152219	764	1
9	68719476736	66376424160	51046350021	1638323	2620	1
10	35184372088832	34509011894545	29184353055900	19467494	9496	1

Table D.1: The number of graphs with minimum degree $0, 1, 2, n-3, n-2$, and $n-1$.

n	A'^0_n	A'^1_n	A'^2_n	A'^{n-3}_n	A'^{n-2}_n	A'^{n-1}_n
1	1	1	0	1	1	1
2	1	1	0	1	1	1
3	4	4	1	4	4	1
4	38	38	10	38	10	1
5	728	728	253	253	26	1
6	26704	26704	12058	1858	76	1
7	1866256	1866256	1052443	15796	232	1
8	251548592	251548592	169488200	152219	764	1
9	66296291072	66296291072	51045018089	1638323	2620	1
10	34496488594816	34496488594816	29184193354806	19467494	9496	1

Table D.2: The number of connected graphs with minimum degree $0, 1, 2, n-3, n-2$, and $n-1$.

Bibliography

The art of being wise is the art of knowing what to overlook. — *William James*

[1] C.M. Adams and S.E. Tavares, "The structured design of cryptographically good S-boxes," *Journal of Cryptology*, Vol. 3, No. 1, 1990, pp. 27–41.

[2] C.M. Adams and S.E. Tavares, "A note on the generation and counting of bent sequences," *IEEE Trans. on Information Theory*, Vol. IT–36, No. 5, 1990, pp. 1170–1173.

[3] C.M. Adams and S.E. Tavares, "The use of bent sequences to achieve higher-order strict avalanche criterion in S-box design," *IEE Proceedings-E*, to appear.

[4] G.B. Agnew, R.C. Mullin, and S.A. Vanstone, "Common application protocols and their security characteristics," *CALMOS CA34C168 Application Notes*, U.S. Patent Number 4,745,568, August 1989.

[5] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *"The Design and Analysis of Computer Algorithms,"* Addison-Wesley, 1974.

[6] S.G. Akl, "On the security of compressed encodings," *Advances in Cryptology, Proc. Crypto'83*, D. Chaum, Ed., Plenum Press, New York, 1984, pp. 209–230.

[7] E. Allender, "Some consequences on the existence of pseudo-random generators," *Journal of Computer and System Sciences*, Vol. 39, 1989, pp. 101–124.

[8] *"American National Standard for Data Encryption Algorithm (DEA),"* X3.92-1981, ANSI, New York.

[9] *"American National Standard for Financial Institution Message Authentication (Wholesale),"* X9.9-1986 (Revised), ANSI, New York.

[10] F. Ayoub, "Probabilistic completeness of substitution-permutation encryption networks," *IEE Proceedings-E*, Vol. 129, 1982, pp. 195–199.

[11] S. Babbage, "On the relevance of the strict avalanche criterion," *Electronic Letters*, Vol. 26, No. 7, 1990, pp. 461–462.

[12] T. Baritaud, H. Gilbert, and M. Girault, "FFT hashing is not collision-free," *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 35–44.

[13] M. Beale and M.F. Monaghan, "Encryption using random Boolean functions," *Proc. of the IMA Conference on Cryptography and Coding, Cirencester, December 1986*, H. Beker and F. Piper, Eds., Oxford University Press, Oxford, pp. 219–230.

[14] K.G. Beauchamp, *"Walsh Functions and Their Applications,"* Academic Press, New York, 1975.

[15] M. Bellare and S. Micali, "How to sign given any trapdoor function," *Proc. 20th ACM Symposium on the Theory of Computing*, 1988, pp. 32–42.

[16] I. Bellefroid and K. Beyen, *"Evaluatie van de Cryptografische Veiligheid van Anti-Virus Paketten (Evaluation of the Security of Anti-Virus Software – in Dutch),"* ESAT Laboratorium, Katholieke Universiteit Leuven, Thesis grad. eng., 1992.

[17] E. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. on Information Theory*, Vol. IT–24, No. 3, 1978, pp. 384–386.

[18] Th. Berson, "Differential cryptanalysis mod 2^{32} with applications to MD5," *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 71–80.

[19] C. Besnard and J. Martin, "DABO: proposed additional message authentication algorithms for ISO 8731," preprint, 1992.

[20] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, Vol. 4, No. 1, 1991, pp. 3–72.

[21] E. Biham and A. Shamir, "Differential cryptanalysis of Feal and N-hash," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 1–16.

[22] E. Biham and A. Shamir, "Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI, and Lucifer," *Advances in Cryptology, Proc. Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 156–171.

[23] E. Biham and A. Shamir, "Differential cryptanalysis of the full 16-round DES," *Technion Technical Report # 708*, December 1991.

[24] E. Biham, "On the applicability of differential cryptanalysis to hash functions," *E.I.S.S. Workshop on Cryptographic Hash Functions*, Oberwolfach (D), March 25-27, 1992.

[25] E. Biham and A. Shamir, *"Differential Cryptanalysis of Iterated Cryptosystems,"* Springer-Verlag, 1992.

[26] G.R. Blakley and I. Borosh, "Rivest-Shamir-Adleman public-key cryptosystems do not always conceal messages," *Comp. and Maths. with Applications*, Vol. 5, 1979, pp. 169–178.

[27] J. Bosset, "Contre les risques d'altération, un système de certification des informations," *01 Informatique*, No. 107, February 1977.

[28] B.O. Brachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, C.H. Meyer, J. Oseas, S. Pilpel, and M. Schilling, *"Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function,"* U.S. Patent Number 4,908,861, March 13, 1990.

[29] G. Brassard, "On computationally secure authentication tags requiring short secret shared keys," *Advances in Cryptology, Proc. Crypto'82*, D. Chaum, R.L. Rivest, and A.T. Sherman, Eds., Plenum Press, New York, 1983, pp. 79–86.

[30] G. Brassard and C. Crépeau, "Non-transitive transfer of confidence: a perfect zero-knowledge protocol for SAT and beyond," *Proc. 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 188–195.

[31] E.F. Brickell, J.H. Moore, and M.R. Purtill, "Structures in the S-boxes of the DES," *Advances in Cryptology, Proc. Crypto'86, LNCS 263*, A.M. Odlyzko, Ed., Springer-Verlag, 1987, pp. 3–8.

[32] E.F. Brickell and A.M. Odlyzko, "Cryptanalysis: a survey of recent results," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 501–540.

[33] L. Brown, J. Pieprzyk, and J. Seberry, "LOKI – a cryptographic primitive for authentication and secrecy applications," *Advances in Cryptology, Proc. Auscrypt'90, LNCS 453*, J. Seberry and J. Pieprzyk, Eds., Springer-Verlag, 1990, pp. 229–236.

[34] L. Brown, M. Kwan, J. Pieprzyk, and J. Seberry, " Improving resistance to differential cryptanalysis and the redesign of LOKI," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[35] P. Camion, "Étude de codes binaires abéliens modulaires autoduaux de petites longueurs," *Revue du CETHEDEC*, NS 79-2, 1979, pp. 3–24.

[36] P. Camion, "Can a fast signature scheme without secret be secure?" *Proc. 2nd International Conference on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, LNCS 228*, A. Poli, Ed., Springer-Verlag, 1986, pp. 215–241.

[37] P. Camion and J. Patarin, "The knapsack hash function proposed at Crypto'89 can be broken," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 39–53.

[38] P. Camion, C. Carlet, P. Charpin, and N. Sendrier, "On correlation-immune functions," *Advances in Cryptology, Proc. Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 86–100.

[39] M. Campana and M. Girault, "How to use compressed encoding mechanisms in data protection (Comment utiliser les fonctions de condensation dans la protection des données – in French)," *Proc. Securicom 1988*, pp. 91–110.

[40] C. Carlet, "A transformation on Boolean functions, its consequences on some problems related to Reed-Muller codes," *Proc. Eurocode'90, LNCS 514*, G. Cohen and P. Charpin, Eds., Springer-Verlag, 1991, pp. 42–50.

[41] C. Carlet, "Partially-bent functions," *Advances in Cryptology, Proc. Crypto'92, LNCS*, E.F. Brickell, Ed., Springer-Verlag, to appear.

[42] J.L. Carter and M.N. Wegman, "Universal classes of hash functions," *Proc. 9th ACM Symposium on the Theory of Computing*, 1977, pp. 106–112.

[43] J.L. Carter and M.N. Wegman, "Universal classes of hash functions," *Journal of Computer and System Sciences*, Vol. 18, 1979, pp. 143–154.

[44] *"Message Handling/Information Processing Systems,"* C.C.I.T.T. Recommendation X.400, 1988.

[45] *"The Directory — Overview of Concepts,"* C.C.I.T.T. Recommendation X.500, 1988, (same as IS 9594-1, 1989).

[46] *"The Directory — Authentication Framework,"* C.C.I.T.T. Recommendation X.509, 1988, (same as IS 9594-8, 1989).

[47] *"Echanges Télématiques Entre les Banques et leurs Clients,"* Comité Français d'Organisation et de Normalisation Bancaires (CFONB), Standard ETEBAC 5, Version 1.2, May 1989.

[48] D. Chaum and J.-H. Evertse, "Cryptanalysis of DES with a reduced number of rounds," *Advances in Cryptology, Proc. Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 192–211.

[49] D. Chaum, M. van der Ham, and B. den Boer, "A provably secure and efficient message authentication scheme," preprint, 1992.

[50] H. Cloetens, Y. Desmedt, L. Bierens, J. Vandewalle and R. Govaerts, "Additional properties in the S-boxes of the DES," *Abstracts Eurocrypt'86, May 20–22, 1986, Linköping, Sweden*, p. 2.3. (Full paper available from the authors.)

[51] F. Cohen, "Computer viruses — theory and experiments," *Computers & Security*, Vol. 6, 1987, pp. 22-35.

[52] F. Cohen, "A cryptographic checksum for integrity protection," *Computers & Security*, Vol. 6, 1987, pp. 505-510.

[53] F. Cohen, *"A Short Course on Computer Viruses,"* ASP Press, Pittsburgh (PA), 1990.

[54] F. Cohen, *"The ASP integrity toolkit. Version 3.5,"* ASP Press, Pittsburgh (PA), 1991.

[55] G.D. Cohen, M.G. Karpovsky, H.F. Mattson, and J.R. Schatz, "Covering radius – survey and recent results," *IEEE Trans. on Information Theory*, Vol. IT–31, No. 3, 1985, pp. 328–343.

[56] D. Coppersmith, "Another birthday attack," *Advances in Cryptology, Proc. Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 14–17.

[57] D. Coppersmith, "Analysis of Jueneman's MDC scheme," unpublished result, 1988.

[58] D. Coppersmith, "Analysis of ISO/CCITT Document X.509 Annex D," *IBM T.J. Watson Center, Yorktown Heights, N.Y., 10598, Internal Memo*, June 11, 1989, (also ISO/IEC JTC1/SC20/WG2/N160).

[59] D. Coppersmith, "Two broken hash functions," *IBM T.J. Watson Center, Yorktown Heights, N.Y., 10598, Research Report RC 18397*, October 6, 1992.

[60] M.J. Coster, B.A. LaMacchia, A.M. Odlyzko, and C. P. Schnorr, "An improved low-density subset sum algorithm," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 54–67.

[61] J. Daemen, R. Govaerts, and J. Vandewalle, "A framework for the design of one-way hash functions including cryptanalysis of Damgård's one-way function based on a cellular automaton," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[62] J. Daemen, A. Bosselaers, R. Govaerts, and J. Vandewalle, "Collisions for Schnorr's FFT-hash," *Presented at the rump session of Asiacrypt'91.*

[63] J. Daemen, R. Govaerts, and J. Vandewalle, "A hardware design model for cryptographic algorithms," *Computer Security – ESORICS 92, Proc. Second European Symposium on Research in Computer Security, LNCS 648*, Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, Eds., Springer-Verlag, 1992, pp. 419–434.

[64] I.B. Damgård, "Collision free hash functions and public key signature schemes," *Advances in Cryptology, Proc. Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 203–216.

[65] I.B. Damgård, "The application of claw free functions in cryptography," *PhD Thesis*, Aarhus University, Mathematical Institute, 1988.

[66] I.B. Damgård, "A design principle for hash functions," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 416–427.

[67] I.B. Damgård, personal communication, 1991.

[68] I.B. Damgård and L.R. Knudsen, "Some attacks on the ARL hash function," *Presented at the rump session of Auscrypt'92.*

[69] G. Davida, Y. Desmedt, and B. Matt, "An approach to containing computer viruses," in *"Rogue Programs: Viruses, Worms and Trojan Horses,"* L.J. Hoffman, Ed., Van Nostrand Reinhold, 1990, pp. 261–272.

[70] D. Davies and W. L. Price, "The application of digital signatures based on public key cryptosystems," *NPL Report* DNACS 39/80, December 1980.

[71] D. Davies, "Applying the RSA digital signature to electronic mail," *IEEE Computer*, Vol. 16, February 1983, pp. 55–62.

[72] D. Davies, "A message authenticator algorithm suitable for a mainframe computer," *Advances in Cryptology, Proc. Crypto'84, LNCS 196*, G.R. Blakley and D. Chaum, Eds., Springer-Verlag, 1985, pp. 393–400.

[73] D. Davies and W. L. Price, "Digital signatures, an update," *Proc. 5th International Conference on Computer Communication*, October 1984, pp. 845–849.

[74] D. Davies and W.L. Price, *"Security for Computer Networks: an Introduction to Data Security in Teleprocessing and Electronic Funds Transfer (2nd edition),"* Wiley & Sons, 1989.

[75] D. Davies, "Investigation of a potential weakness in the DES algorithm," preprint, July 1987 (revised January 1990).

[76] M. Davio, J.-P. Deschamps, and A. Thayse, *"Discrete and Switching Functions,"* McGraw-Hill, 1978.

[77] M. Davio, Y. Desmedt, and J.-J. Quisquater, "Propagation characteristics of the DES," *Advances in Cryptology, Proc. Eurocrypt'84, LNCS 209*, N. Cot, T. Beth, and I. Ingemarsson, Eds., Springer-Verlag, 1985, pp. 62–73.

[78] M.H. Dawson and S.E. Tavares, "An expanded set of S-box design criteria based on information theory and its relation to differential-like attacks," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 352–367.

[79] W. de Jonge and D. Chaum, "Attacks on some RSA signatures," *Advances in Cryptology, Proc. Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 18–27.

[80] W. de Jonge and D. Chaum, "Some variations on RSA signatures and their security," *Advances in Cryptology, Proc. Crypto'86, LNCS 263*, A.M. Odlyzko, Ed., Springer-Verlag, 1987, pp. 49–59.

[81] B. den Boer and A. Bosselaers, "An attack on the last two rounds of MD4," *Advances in Cryptology, Proc. Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 194–203.

[82] B. den Boer, personal communication.

[83] B. den Boer and A. Bosselaers, "Collisions for the compression function of MD5," preprint, April 1992.

[84] B. den Boer, "A simple and key-economical authentication scheme," preprint, 1992.

[85] D. Denning, *"Cryptography and Data Security,"* Addison-Wesley, 1982.

[86] D. Denning, "Digital signatures with RSA and other public-key cryptosystems," *Communications ACM*, Vol. 27, April 1984, pp. 388–392.

[87] A. De Santis and M. Yung, "On the design of provably-secure cryptographic hash functions," *Advances in Cryptology, Proc. Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 412–431.

[88] Y. Desmedt, J. Vandewalle, and R. Govaerts, "The mathematical relation between the economic, cryptographic and information theoretical aspects of authentication," *Proc. Fourth Symposium on Information Theory in the Benelux, Haasrode, Belgium, 26-27 May 1983*, pp. 63-65.

[89] Y. Desmedt, J.-J. Quisquater, and M. Davio, "Dependence of output on input in DES: small avalanche characteristics," *Advances in Cryptology, Proc. Crypto'84, LNCS 196*, G.R. Blakley and D. Chaum, Eds., Springer-Verlag, 1985, pp. 359–376.

[90] Y. Desmedt, "Analysis of the security and new algorithms for modern industrial cryptography," *Doctoral Dissertation*, Katholieke Universiteit Leuven, 1984.

[91] Y. Desmedt, "Unconditionally secure authentication schemes and practical and theoretical consequences," *Advances in Cryptology, Proc. Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 42–55.

[92] Y. Desmedt, "What happened with knapsack cryptographic schemes?" in *"Performance Limits in Communication, Theory and Practice,"* J.K. Skwirzynski, Ed., Kluwer, 1988, pp. 113–134.

[93] M. Desoete, K. Vedder, and M. Walker, "Cartesian authentication schemes," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 476–490.

[94] M. Dichtl, personal communication, 1991.

[95] W. Diffie and M.E. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory*, Vol. IT–22, No. 6, 1976, pp. 644–654.

[96] W. Diffie and M.E. Hellman, "Privacy and authentication: an introduction to cryptography," *Proc. IEEE*, Vol. 67, No. 3, March 1979, pp. 397-427.

[97] C. Ding, G. Xiao, and W. Shan, *"The Stability Theory of Stream Ciphers,"* LNCS 561, Springer-Verlag, 1991.

[98] P.E. Dunne, *"The Complexity of Boolean Networks,"* A.P.I.C. Studies in Data Processing No. 29, Academic Press, 1988.

[99] J. Ekberg, S. Herda, and J. Virtamo, "TeleTrusT — Technical concepts and basic mechanisms," in *"Research into Networks and Distributed Applications,"* R. Speth, Ed., Elsevier Science Publishers, 1988, pp. 523–533.

[100] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. on Information Theory*, Vol. IT–31, No. 4, 1985, pp. 469–472.

[101] J.-H. Evertse, "Linear structures in block ciphers," *Advances in Cryptology, Proc. Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 249–266.

[102] J.H. Evertse and E. Van Heyst, "Which new RSA-signatures can be computed from certain given RSA-signatures?" *Journal of Cryptology*, Vol. 5, No. 1, 1992, pp. 41–52.

[103] V. Fåk, "Repeated uses of codes which detect deception," *IEEE Trans. on Information Theory*, Vol. IT–25, No. 2, 1979, pp. 233–234.

[104] H. Feistel, "Cryptography and computer privacy," *Scientific American*, Vol. 228, No. 5, May 1973, pp. 15–23.

[105] H. Feistel, W.A. Notz, and J.L. Smith, "Some cryptographic techniques for machine-to-machine data communications," *Proc. IEEE*, Vol. 63, No. 11, November 1975, pp. 1545–1554.

[106] W. Feller, *"An Introduction to Probability Theory and Its Applications, Vol. 1,"* Wiley, 1968.

[107] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," *Advances in Cryptology, Proc. Crypto'86, LNCS 263*, A.M. Odlyzko, Ed., Springer-Verlag, 1987, pp. 186–194.

[108] *"Data Encryption Standard,"* Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[109] *"DES Modes of Operation,"* Federal Information Processing Standard (FIPS), Publication 81, National Bureau of Standards, US Department of Commerce, Washington D.C., December 1980.

[110] *"Computer Data Authentication,"* Federal Information Processing Standard (FIPS), Publication 131, National Bureau of Standards, US Department of Commerce, Washington D.C., May 1985.

[111] *"Digital Signature Standard,"* Federal Information Processing Standard (FIPS), Draft, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., August 30, 1991.

[112] *"Secure Hash Standard,"* Federal Information Processing Standard (FIPS), Draft, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., January 31, 1992.

[113] P. Flajolet and A.M. Odlyzko, "Random mapping statistics," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 329–354.

[114] R. Forré, "The strict avalanche criterion: spectral properties of Boolean functions and an extended definition," *Advances in Cryptology, Proc. Crypto'88, LNCS 403*, S. Goldwasser, Ed., Springer-Verlag, 1990, pp. 450–468.

[115] R. Forré, "Methods and instruments for designing S-boxes," *Journal of Cryptology*, Vol. 2, No. 3, 1990, pp. 115–130.

[116] M.R. Garey and D.S. Johnson, *"Computers and Intractability. A Guide to the Theory of NP-Completeness,"* W.H. Freeman and Company, New York, 1979.

[117] R. Garon and R. Outerbridge, "DES watch: an examination of the sufficiency of the Data Encryption Standard for financial institution information security in the 1990's," *Cryptologia*, Vol. XV, No. 3, 1991, pp. 177–193.

[118] E. Gilbert, F. MacWilliams, and N. Sloane, "Codes which detect deception," *Bell System Technical Journal*, Vol. 53, No. 3, 1974, pp. 405–424.

[119] J.K. Gibson, "Some comments on Damgård's hashing principle," *Electronic Letters*, Vol. 26, No. 15, 1990, pp. 1178–1179.

[120] J.K. Gibson, "Discrete logarithm hash function that is collision free and one way," *IEE Proceedings-E*, Vol. 138, No. 6, November 1991, pp. 407–410.

[121] Y. Girardot, "Bull CP8 smart card uses in cryptology," *Advances in Cryptology, Proc. Eurocrypt'84, LNCS 209*, N. Cot, T. Beth, and I. Ingemarsson, Eds., Springer-Verlag, 1985, pp. 464–469.

[122] M. Girault, "Hash-functions using modulo-n operations," *Advances in Cryptology, Proc. Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 217–226.

[123] M. Girault, R. Cohen, and M. Campana, "A generalized birthday attack," *Advances in Cryptology, Proc. Eurocrypt'88, LNCS 330*, C.G. Günther, Ed., Springer-Verlag, 1988, pp. 129–156.

[124] M. Girault, P. Toffin, and B. Vallée, "Computation of approximate L-th roots modulo n and application to cryptography," *Advances in Cryptology, Proc. Crypto'88, LNCS 403*, S. Goldwasser, Ed., Springer-Verlag, 1990, pp. 100–117.

[125] M. Girault, "On consequences of a recent Coppersmith attack against the CCITT X.509 hash function," *Presented at the rump session of Crypto'89.*

[126] Ph. Godlewski and P. Camion, "Manipulations and errors, detection and localization," *Advances in Cryptology, Proc. Eurocrypt'88, LNCS 330*, C.G. Günther, Ed., Springer-Verlag, 1988, pp. 97–106.

[127] O. Goldreich and L.A. Levin, "A hard-core predicate for all one-way functions," *Proc. 21th ACM Symposium on the Theory of Computing*, 1989, pp. 25-32.

[128] S. Goldwasser, S. Micali, and R. Rivest, "A "paradoxical" solution to the signature problem," *Proc. 25th IEEE Symposium on Foundations of Computer Science*, October 1984, pp. 441–448.

[129] S. Goldwasser, S. Micali, and R.L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, Vol. 17, No. 2, 1988, pp. 281-308.

[130] S.W. Golomb, *"Shift register sequences,"* Holden-Day, San Francisco, 1967.

[131] J.A. Gordon and H. Retkin, "Are big S-boxes best?" *Cryptography: Proc. Workshop Cryptography (Burg Feuerstein 1982), LNCS 149*, T. Beth, Ed., Springer Verlag, 1983, pp. 257–262.

[132] J.A. Gordon, "Strong RSA keys," *Electronic Letters*, Vol. 20, No. 12, 1984, pp. 514–516.

[133] J.A. Gordon, "How to forge RSA certificates," *Electronic Letters*, Vol. 21, No. 9, 1985, pp. 377–379.

[134] R.L. Graham, D.E. Knuth, and O. Patashnik, *"Concrete Mathematics,"* Addison-Wesley, 1989.

[135] L.C. Guillou, "Smart cards and conditional access," *Advances in Cryptology, Proc. Eurocrypt'84, LNCS 209*, N. Cot, T. Beth, and I. Ingemarsson, Eds., Springer-Verlag, 1985, pp. 480–489.

[136] L.C. Guillou, M. Davio, and J.-J. Quisquater, "Public-key techniques: randomness and redundancy," *Cryptologia*, Vol. 13, April 1989, pp. 167–189.

[137] L.C. Guillou, J.-J. Quisquater, M. Walker, P. Landrock, and C. Shaer, "Precautions taken against various potential attacks in ISO/IEC DIS 9796," *Advances in Cryptology, Proc. Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 465–473.

[138] S. Harari, "Non linear non commutative functions for data integrity," *Advances in Cryptology, Proc. Eurocrypt'84, LNCS 209*, N. Cot, T. Beth, and I. Ingemarsson, Eds., Springer-Verlag, 1985, pp. 25–32.

[139] F. Harary, *"Graph Theory,"* Addison-Wesley, 1969.

[140] F. Harary and E.M. Palmer, *"Graphical Enumeration,"* Academic Press, 1973.

[141] G.H. Hardy and E.M. Wright, *"An Introduction to the Theory of Numbers (5th edition),"* Oxford University Press, 1979.

[142] M.A. Harrison, *"Introduction to Switching and Automata Theory,"* McGraw-Hill, 1965.

[143] M.A. Harrison, "On asymptotic estimates in switching and automata theory," *Journal of the Association for Computing Machinery*, Vol. 13, No. 1, January 1966, pp. 151–157.

[144] F. Heider, D. Kraus, and M. Welschenbach, "Some preliminary remarks on the Decimal Shift and Add algorithm (DSA)," *Abstracts Eurocrypt'86, May 20–22, 1986, Linköping, Sweden*, p. 1.2. (Full paper available from the authors.)

[145] H.J. Helgert and R.D. Stinaff, "Minimum-distance bounds for binary linear codes," *IEEE Trans. on Information Theory*, Vol. IT–19, No. 3, 1973, pp. 344–356.

[146] M. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer, "Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard," *Information Systems Lab., Dept. of Electrical Eng., Stanford Univ.*, 1976.

[147] M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Trans. on Information Theory*, Vol. IT–26, No. 4, 1980, pp. 401–406.

[148] Y.J. Huang and F. Cohen, "Some weak points of one fast cryptographic checksum algorithm and its improvement," *Computers & Security*, Vol. 7, 1988, pp. 503-505.

[149] R. Impagliazzo and M. Naor, "Efficient cryptographic schemes provably as secure as subset sum," *Proc. 30th IEEE Symposium on Foundations of Computer Science*, 1989, pp. 236–241.

[150] R. Impagliazzo and S. Rudich, "Limits on the provable consequences of one-way permutations," *Proc. 21st ACM Symposium on the Theory of Computing*, 1990, pp. 44–61.

[151] *"Information processing – Open systems interconnection – Basic reference model – Part 2: Security architecture,"* IS 7498/2, ISO/IEC, 1987.

[152] *"Information technology - Data cryptographic techniques - Modes of operation for a 64-bit block cipher algorithm,"* IS 8372, ISO/IEC, 1987.

[153] *"Banking - Requirements for message authentication (wholesale),"* IS 8730, ISO, 1990.

[154] *"Banking - approved algorithms for message authentication, Part 1, DEA,"* IS 8731-1, ISO, 1987. *"Part 2, Message Authentication Algorithm (MAA),"* IS 8731-2, ISO, 1987.

[155] *"Information technology - Security techniques - Digital signature scheme giving message recovery,"* IS 9796, ISO/IEC, 1991.

[156] *"Information technology - Data cryptographic techniques - Data integrity mechanisms using a cryptographic check function employing a block cipher algorithm,"* IS 9797, ISO/IEC, 1989.

[157] *"Information technology - Security techniques - Procedures for the registration of cryptographic algorithms,"* IS 9979, ISO/IEC, 1991.

[158] *"Information technology - Security techniques - Modes of operation of an n-bit block cipher algorithm,"* IS 10116, ISO/IEC, 1991.

[159] *"Information technology - Security techniques - Hash-functions, Part 1: General and Part 2: Hash-functions using an n-bit block cipher algorithm,"* DIS 10118, ISO/IEC, 1992.

[160] *"Report of two ISO/IEC JTC1/SC20/WG2 ad-hoc editorial meetings on 'Hash functions for digital signatures',"* ISO-IEC/JTC1/SC20/WG2 N70, 1990.

[161] *"Hash functions using a pseudo random algorithm,"* ISO-IEC/JTC1/SC27/WG2 N98, Japanese contribution, 1991.

[162] *"AR fingerprint function,"* ISO-IEC/JTC1/SC27/WG2 N179, working document, 1992.

[163] C.J.A. Jansen and D.E. Boekee, "Modes of blockcipher algorithms an their protection against active eavesdropping," *Advances in Cryptology, Proc. Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 327–347.

[164] C.J.A. Jansen, "Investigations on nonlinear streamcipher systems: construction and evaluation methods," *Doctoral Dissertation*, Technische Universiteit Delft, 1989.

[165] F. Jorissen, "A security evaluation of the public-key cipher system proposed by McEliece, used as a combined scheme," *ESAT report K.U. Leuven*, 1986.

[166] R.R. Jueneman, "Analysis of certain aspects of Output Feedback Mode," *Advances in Cryptology, Proc. Crypto'82*, D. Chaum, R.L. Rivest, and A.T. Sherman, Eds., Plenum Press, New York, 1983, pp. 99–127.

[167] R.R. Jueneman, S.M. Matyas, and C.H. Meyer, "Message authentication with Manipulation Detection Codes," *Proc. 1983 IEEE Symposium on Security and Privacy*, 1984, pp. 33-54.

[168] R.R. Jueneman, S.M. Matyas, and C.H. Meyer, "Message authentication," *IEEE Communications Mag.*, Vol. 23, No. 9, 1985, pp. 29-40.

[169] R.R. Jueneman, "A high speed Manipulation Detection Code," *Advances in Cryptology, Proc. Crypto'86, LNCS 263*, A.M. Odlyzko, Ed., Springer-Verlag, 1987, pp. 327–347.

[170] R.R. Jueneman, "Electronic document authentication," *IEEE Network Mag.*, Vol. 1, No. 2, 1987, pp. 17-23.

[171] A. Jung, "Implementing the RSA cryptosystem," *Computers & Security*, Vol. 6, 1987, pp. 342–350.

[172] A. Jung, "The strength of the ISO/CCITT hash function," preprint, October 1990.

[173] A. Jung, personal communication, 1992.

[174] D. Kahn, *"The Codebreakers. The Story of Secret Writing,"* MacMillan, New York, 1967.

[175] B.S. Kaliski, "The MD2 Message-Digest algorithm," *Request for Comments (RFC) 1319*, Internet Activities Board, Internet Privacy Task Force, April 1992.

[176] J.B. Kam and G.I. Davida, "Structured design of substitution-permutation encryption networks," *IEEE Trans. on Computers*, Vol. C–28, 1979, pp. 747–753.

[177] K. Kim, T. Matsumoto, and H. Imai, "A recursive construction method of S-boxes satisfying strict avalanche criterion," *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 564–573.

[178] K. Kim, "Constructions of DES-like S-boxes based on Boolean functions satisfying the SAC," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[179] L.R. Knudsen, "Cryptanalysis of LOKI," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[180] M. Kwan and J. Pieprzyk, "A general purpose technique for locating key scheduling weaknesses in DES-like cryptosystems," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[181] X. Lai and J.L. Massey, "A proposal for a new block encryption standard," *Advances in Cryptology, Proc. Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 389–404.

[182] X. Lai, J.L. Massey, and S. Murphy, "Markov ciphers and differential cryptanalysis," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 17–38.

[183] X. Lai and J.L. Massey, "Hash functions based on block ciphers," *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 55–70.

[184] X. Lai, *"On the Design and Security of Block Ciphers,"* ETH Series in Information Processing, Vol. 1, J. Massey, Ed., Hartung-Gorre Verlag, Konstanz, 1992.

[185] X. Lai, R.A. Rueppel, and J. Woollven, "A fast cryptographic checksum algorithm based on stream ciphers," *Advances in Cryptology, Proc. Auscrypt'92, LNCS*, Springer-Verlag, to appear.

[186] Ph. Langevin, "Covering radius of RM(1,9) in RM(3,9)," *Proc. Eurocode'90, LNCS 514*, G. Cohen and P. Charpin, Eds., Springer-Verlag, 1991, pp. 51–59.

[187] Ph. Langevin, "On the orphans and covering radius of the Reed-Muller codes," *Proc. 9th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, LNCS 539*, H.F. Mattson, T. Mora, and T.R.N. Rao, Eds., Springer-Verlag, 1991, pp. 234–240.

[188] J.Y. Lee, E.H. Lu, and P.C.H. Chen, "Random code chaining," *Electronic Letters*, Vol. 24, No. 10, 1988, pp. 579–580.

[189] Y. Li and X. Wang, "A joint authentication and encryption scheme based on algebraic coding theory," *Proc. 9th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, LNCS 539*, H.F. Mattson, T. Mora, and T.R.N. Rao, Eds., Springer-Verlag, 1991, pp. 241–245.

[190] R. Lidl and H. Niederreiter, *"Introduction to Finite Fields and Their Applications,"* Cambridge University Press, Cambridge, 1986.

[191] C. Linden and H. Block, "Sealing electronic money in Sweden," *Computers & Security*, Vol. 1, No. 3, 1982, p. 226.

[192] J. Linn, "Privacy enhancement for Internet electronic mail, Part I: Message encipherment and authentication procedures," *Request for Comments (RFC) 989*, Internet Activities Board, Internet Privacy Task Force, February 1987.

[193] J. Linn, "Privacy enhancement for Internet electronic mail, Part I: Message encipherment and authentication procedures," *Request for Comments (RFC) 1040*, Internet Activities Board, Internet Privacy Task Force, January 1988.

[194] S. Lloyd, "Counting functions satisfying a higher order strict avalanche criterion," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 63–74.

[195] S. Lloyd, "Characterising and counting functions satisfying the strict avalanche criterion of order $(n-3)$," *Proc. 2nd IMA Conference on Cryptography and Coding, 1989*, Clarendon Press, Oxford, 1992, pp. 165–172.

[196] S. Lloyd, "Properties of binary functions, " *Advances in Cryptology, Proc. Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 124–139.

[197] S. Lloyd, "Counting binary functions with certain cryptographic properties, " *Journal of Cryptology*, Vol. 5, No. 2, 1992, pp. 107–131.

[198] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM Journal on Computing*, Vol 17, No. 2, April 1988, pp. 373–386.

[199] F.J. MacWilliams and N.J.A. Sloane, *"The Theory of Error-Correcting Codes,"* North-Holland Publishing Company, Amsterdam, 1978.

[200] J.A. Maiorana, "A classification of the cosets of the Reed-Muller code $\mathcal{R}(1,6)$," *Mathematics of Computation*, Vol. 57, No. 195, July 1991, pp. 403–414.

[201] J.L. Massey, "Cryptography — A selective survey," *Digital Communications (Proc. 1985 International Tirrenia Workshop)*, E. Biglieri and G. Prati, Eds., Elsevier Science Publ., 1986, pp. 3–25.

[202] J.L. Massey, "An introduction to contemporary cryptology," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 3–39.

[203] S.M. Matyas, C.H. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Techn. Disclosure Bull.*, Vol. 27, No. 10A, 1985, pp. 5658–5659.

[204] S.M. Matyas, "Key handling with control vectors," *IBM Systems Journal*, Vol. 30, No. 2, 1991, pp. 151–174.

[205] U.M. Maurer and J.L. Massey, "Cascade ciphers: the importance of being first," *Presented at the 1990 IEEE International Symposium on Information Theory*, San Diego, CA, Jan. 14-19, 1990.

[206] U.M. Maurer, "New approaches to the design of self-synchronizing stream ciphers," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 458–471.

[207] R.J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *DSN Progress Report 42–44*, Jet Propulsion Laboratory, Pasadena, CA, 1978, pp. 114–116.

[208] W. Meier and O. Staffelbach, "Nonlinearity criteria for cryptographic functions," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 549–562.

[209] H. Meijer and S. Akl, "Remarks on a digital signature scheme," *Cryptologia*, Vol. 7, No. 2, April 1983, pp. 183–185.

[210] R. Merkle and M. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Trans. on Information Theory*, Vol. IT–24, No. 5, 1978, pp. 525–530.

[211] R. Merkle, *"Secrecy, Authentication, and Public Key Systems,"* UMI Research Press, 1979.

[212] R. Merkle, "A certified digital signature," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 218–238.

[213] R. Merkle, "One way hash functions and DES," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428–446.

[214] R. Merkle, "A fast software one-way hash function," *Journal of Cryptology*, Vol. 3, No. 1, 1990, pp. 43–58.

[215] C.H. Meyer and S.M. Matyas, *"Cryptography: a New Dimension in Data Security,"* Wiley & Sons, 1982.

[216] C.H. Meyer and M. Schilling, "Secure program load with Manipulation Detection Code," *Proc. Securicom 1988*, pp. 111–130.

[217] C.H. Meyer, "Cryptography - A state of the art review," *Proc. Compeuro 89, VLSI and Computer Peripherals, 3rd Annual European Computer Conference*, IEEE Computer Society Press, 1989, pp. 150-154.

[218] C. Mitchell and M. Walker, "Solutions to the multidestination secure electronic mail problem," *Computers & Security*, Vol. 7, 1988, pp. 483–488.

[219] C. Mitchell, M. Walker, and D. Rush, "CCITT/ISO standards for secure message handling," *IEEE Journal on Selected Areas in Communications*, Vol. 7, May 1989, pp. 517–524.

[220] C. Mitchell, D. Rush, and M. Walker, "A remark on hash functions for message authentication," *Computers & Security*, Vol. 8, 1989, pp. 517-524.

[221] C. Mitchell, "Multi-destination secure electronic mail," *The Computer Journal*, Vol. 32, No. 1, 1989, pp. 13-15.

[222] C. Mitchell, "Enumerating Boolean functions of cryptographic significance," *Journal of Cryptology*, Vol. 2, No. 3, 1990, pp. 155–170.

[223] C. Mitchell, F. Piper, and P. Wild, "Digital signatures," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 325–378.

[224] C. Mitchell, "Authenticating multicast electronic mail messages using a bidirectional MAC is insecure," *IEEE Trans. on Computers*, Vol. 41, No. 4, 1992, pp. 505–507.

[225] S. Miyaguchi, A. Shiraisi, and A. Shimizu, "Fast data encryption algorithm Feal-8," *Review of Electrical Communications Laboratories*, Vol. 36, No. 4, 1988, pp. 433–437.

[226] S. Miyaguchi, M. Iwata, and K. Ohta, "New 128-bit hash function," *Proc. 4th International Joint Workshop on Computer Communications*, Tokyo, Japan, July 13–15, 1989, pp. 279–288.

[227] S. Miyaguchi, K. Ohta, and M. Iwata, "Confirmation that some hash functions are not collision free," *Advances in Cryptology, Proc. Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 326–343.

[228] S. Miyaguchi, "The FEAL cipher family," *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 627–638.

[229] S. Miyaguchi, K. Ohta, and M. Iwata, "128-bit hash function (N-hash)," *Proc. Securicom 1990*, pp. 127–137.

[230] S.F. Mjølsnes, "A hash of some one-way hash functions and birthdays," preprint, 1989.

[231] J.H. Moore and G.J. Simmons, "Cycle structure of the DES for keys having palindromic (or antipalindromic) sequences of round keys," *IEEE Trans. on Software Engineering*, Vol. 13, 1987, pp. 262-273.

[232] J.H. Moore, "Protocol failures in cryptosystems," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 543–558.

[233] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," *Proc. 21st ACM Symposium on the Theory of Computing*, 1990, pp. 387–394.

[234] H. Niederreiter and C.P. Schnorr, "Local randomness in candidate one-way functions," *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 408–419.

[235] K. Nishimura and M. Sibuya, "Probability to meet in the middle," *Journal of Cryptology*, Vol. 2, No. 1, 1990, pp. 13–22.

[236] M. Nuttin, *"Cryptanalyse van het DES Algoritme in de CFB Mode (Cryptanalysis of the CFB Mode of the DES – in Dutch),"* ESAT Laboratorium, Katholieke Universiteit Leuven, Thesis grad. eng., 1992.

[237] K. Nyberg, "Constructions of bent functions and difference sets," *Advances in Cryptology, Proc. Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 151–160.

[238] K. Nyberg, personal communication, 1990.

[239] K. Nyberg, "Perfect nonlinear S-boxes," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 378–386.

[240] K. Nyberg, "On the construction of highly nonlinear permutations," *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 92–98.

[241] L. O'Connor, "An analysis of product ciphers based on the properties of Boolean functions," *PhD Thesis*, University of Waterloo, Canada, 1992.

[242] W. Ogata and K. Kurosawa, "On claw free families," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[243] J.D. Olsen, R.A. Scholtz, and L.R. Welch, "Bent-function sequences," *IEEE Trans. on Information Theory*, Vol. IT–28, No. 6, 1982, pp. 858–864.

[244] J.C. Pailles and M. Girault, "The security processor CRIPT," *4th IFIP SEC*, Monte-Carlo, December 1986, pp. 127–139.

[245] J. Patarin, personal communication, 1991.

[246] J. Pieprzyk and G. Finkelstein, "Towards effective non-linear cryptosystem design," *IEE Proceedings-E*, Vol. 135, 1988, pp. 325–335.

[247] J. Pieprzyk, "Non-linearity of exponent permutations," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 80–92.

[248] J. Pieprzyk and R. Safavi-Naini, "Randomized authentication systems," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 472–481.

[249] J. Pieprzyk, "On bent permutations," in *"Finite Fields, Coding Theory, and Advances in Communications and Computing," Lecture Notes in Pure and Applied Mathematics 141*, G.L. Mullen and P.J.-S. Shiue, Eds., Marcel Dekker Inc., New York, 1993, pp. 173–181.

[250] D. Pinkas, "The need for a standardized compression algorithm for digital signatures," *Abstracts Eurocrypt'86, May 20–22, 1986, Linköping, Sweden*, p. 1.4.

[251] J.M. Pollard, "Theorems on factorisation and primality testing," *Proc. Cambridge Philos. Soc.*," Vol. 76, 1974, pp. 521-528.

[252] B. Preneel, A. Bosselaers, R. Govaerts, and J. Vandewalle, " A chosen text attack on the modified cryptographic checksum algorithm of Cohen and Huang," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 154–163.

[253] B. Preneel, A. Bosselaers, R. Govaerts, and J. Vandewalle, "Collision free hash functions based on blockcipher algorithms," *Proc. 1989 International Carnahan Conference on Security Technology*, pp. 203-210.

[254] B. Preneel, R. Govaerts, and J. Vandewalle, "Cryptographically secure hash functions: an overview," *ESAT Internal Report, K.U. Leuven*, 1989.

[255] B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle, "Propagation characteristics of Boolean functions," *Advances in Cryptology, Proc. Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 161–173.

[256] B. Preneel, R. Govaerts, and J. Vandewalle, "Information integrity protection and authentication in a banking environment," *Proc. Secubank'90.*

[257] B. Preneel, A. Bosselaers, R. Govaerts, and J. Vandewalle, "Cryptanalysis of a fast cryptographic checksum algorithm," *Computers & Security*, Vol. 9, 1990, pp. 257–262.

[258] B. Preneel, R. Govaerts, and J. Vandewalle, "Boolean functions satisfying higher order propagation criteria," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 141–152.

[259] B. Preneel, D. Chaum, W. Fumy, C.J.A. Jansen, P. Landrock, and G. Roelofsen, "Race Integrity Primitives Evaluation (RIPE): a status report," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 547–551.

[260] B. Preneel, R. Govaerts, and J. Vandewalle, "Collision resistant hash functions based on block ciphers, " *E.I.S.S. Workshop on Cryptographic Hash Functions*, Oberwolfach (D), March 25-27, 1992.

[261] B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions for information authentication," *Proc. Compeuro 92, Computer Systems and Software Engineering, 6th Annual European Computer Conference*, IEEE Computer Society Press, 1992, pp. 475–480.

[262] B. Preneel, R. Govaerts, and J. Vandewalle, "On the power of memory in the design of collision resistant hash functions," *Advances in Cryptology, Proc. Auscrypt'92, LNCS*, Springer-Verlag, to appear.

[263] B. Preneel, R. Govaerts, and J. Vandewalle, "An attack on two hash functions by Zheng, Matsumoto, and Imai," *Presented at the rump session of Auscrypt'92.*

[264] W.L. Price, "Hash functions — a tutorial and status report," *NPL Report DITC 151/89*, November 1989.

[265] W.E. Proebster, "The evolution of data memory and storage: an overview," in *"Computer Systems and Software Engineering,"* P. Dewilde and J. Vandewalle, Eds., Kluwer Academic Publishers, 1992, pp. 1–23.

[266] J.-J. Quisquater and L. Guillou, "A "paradoxical" identity-based signature scheme resulting from zero-knowledge," *Advances in Cryptology, Proc. Crypto'88, LNCS 403*, S. Goldwasser, Ed., Springer-Verlag, 1990, pp. 216–231.

[267] J.-J. Quisquater and J.-P. Delescaille, "How easy is collision search? Application to DES," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 429–434.

[268] J.-J. Quisquater and M. Girault, "2n-bit hash-functions using n-bit symmetric block cipher algorithms," *Abstracts Eurocrypt'89, April 10-13, 1989, Houthalen, Belgium.*

[269] J.-J. Quisquater and M. Girault, "2n-bit hash-functions using n-bit symmetric block cipher algorithms," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 102–109.

[270] J.-J. Quisquater and J.-P. Delescaille, "How easy is collision search. New results and applications to DES," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 408–413.

[271] J.-J. Quisquater, personal communication, 1989.

[272] J.-J. Quisquater and Y. Desmedt, "Chinese lotto as an exhaustive code-breaking machine," *Computer*, November 1991, pp. 14-22.

[273] J.-J. Quisquater, "Collisions," *E.I.S.S. Workshop on Cryptographic Hash Functions*, Oberwolfach (D), March 25-27, 1992.

[274] M.O. Rabin, "Digitalized signatures," in *"Foundations of Secure Computation,"* R. Lipton and R. DeMillo, Eds., Academic Press, New York, 1978, pp. 155-166.

[275] T.R.N. Rao and K.H. Nam, "A private-key algebraic-coded cryptosystem," *Advances in Cryptology, Proc. Crypto'86, LNCS 263*, A.M. Odlyzko, Ed., Springer-Verlag, 1987, pp. 35–48.

[276] R.C. Read, "Some unusual enumeration problems," *Annals New York Acad. Sc.*, Vol. 175, 1970, pp. 314–326.

[277] "Race Integrity Primitives Evaluation (RIPE): final report," RACE 1040, 1993.

[278] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications ACM*, Vol. 21, February 1978, pp. 120–126.

[279] R.L. Rivest, "The MD4 message digest algorithm," *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 303–311.

[280] R.L. Rivest, "Cryptography," in *"Handbook of Theoretical Computer Science. Vol. A: Algorithms and Complexity,"* J. Van Leeuwen, Ed., Elsevier, pp.719–755.

[281] R.L. Rivest, "The MD5 message digest algorithm," *Presented at the rump session of Crypto'91.*

[282] R.L. Rivest, "The MD4 message-digest algorithm," *Request for Comments (RFC) 1320*, Internet Activities Board, Internet Privacy Task Force, April 1992.

[283] R.L. Rivest, "The MD5 message-digest algorithm," *Request for Comments (RFC) 1321*, Internet Activities Board, Internet Privacy Task Force, April 1992.

[284] J. Rompel, "One-way functions are necessary and sufficient for secure signatures," *Proc. 22nd ACM Symposium on the Theory of Computing*, 1990, pp. 387–394.

[285] O.S. Rothaus, "On bent functions," *Journal of Combinatorial Theory (A)*, Vol. 20, 1976, pp. 300–305.

[286] R.A. Rueppel, *"Analysis and Design of Stream Ciphers,"* Springer-Verlag, 1986.

[287] R.A. Rueppel, "Stream ciphers," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 65–134.

[288] A. Russell, "Necessary and sufficient conditions for collision-free hashing," *Advances in Cryptology, Proc. Crypto'92, LNCS*, E.F. Brickell, Ed., Springer-Verlag, to appear.

[289] B. Sadeghiyan and J. Pieprzyk, "A construction for one way hash functions and pseudorandom bit generators," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 431–445.

[290] B. Sadeghiyan and J. Pieprzyk, " How to construct a family of strong one way permutations," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[291] R. Safavi-Naini and J. Seberry, "Error-correcting codes for authentication and subliminal channels," *IEEE Trans. on Information Theory*, Vol. IT–37, No. 1, 1991, pp. 13–17.

[292] R. Safavi-Naini, "Feistel type authentication codes," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[293] D.V. Sarwate, "A note on universal classes of hash functions," *Information Processing Letters*, Vol. 10, 1980, pp. 41–45.

[294] W.G. Schneeweiss, *"Boolean Functions with Engineering Applications and Computer Programs,"* Springer Verlag, 1989.

[295] B. Schneier, "One-way hash functions," *Dr. Dobb's Journal*, Vol. 16, No. 9, 1991, pp. 148–151.

[296] C.P. Schnorr, "Efficient identification and signatures for smart cards," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 239–252.

[297] C.P. Schnorr, "An efficient cryptographic hash function," *Presented at the rump session of Crypto'91.*

[298] C.P. Schnorr, "FFT-Hash II, efficient cryptographic hashing," *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 45–54.

[299] J. Seberry and J. Pieprzyk, *"Cryptography: an Introduction to Computer Security*, Prentice Hall, 1988.

[300] R. Sedgewick, T.G. Szymanski, and A.C. Yao, "The complexity of finding cycles in periodic functions," *SIAM Journal Comput.*, Vol. 11, No. 2, 1982, pp. 376–390.

[301] A. Shamir, "On the security of DES," *Advances in Cryptology, Proc. Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 280–281.

[302] C.E. Shannon, "A symbolic analysis of relay and switching circuits," *Trans. of the AIEE*, Vol. 57, 1938, pp. 713–723. pp. 623-656.

[303] C.E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, Vol. 28, 1949, pp. 656-715.

[304] A. Shimizu and S. Miyaguchi, "Fast data encipherment algorithm FEAL," *Abstracts Eurocrypt'87, April 13-15, 1987, Amsterdam, The Netherlands*, pp. VII-11–VII-14.

[305] A. Shimizu and S. Miyaguchi, "Fast data encipherment algorithm FEAL," *Advances in Cryptology, Proc. Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 267–278.

[306] J.F. Shoch and J.A. Hupp, "The "worm" programs—early experience with a distributed computation," *Communications ACM*, Vol. 25, No. 3, March 1982, pp. 172–180.

[307] T. Siegenthaler, "Correlation immunity of non-linear combining functions for cryptographic applications," *IEEE Trans. on Information Theory*, Vol. IT–30, No. 5, 1984, pp. 776–780.

[308] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Trans. on Computers*, Vol. C–34, 1985, pp. 81–85.

[309] G.J. Simmons, "A natural taxonomy for digital information authentication schemes," *Advances in Cryptology, Proc. Crypto'87, LNCS 293*, C. Pomerance, Ed., Springer-Verlag, 1988, pp. 269–288.

[310] G.J. Simmons, "A survey of information authentication," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 381–419.

[311] O. Staffelbach and W. Meier, "Analysis of pseudo random sequences generated by cellular automata," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 186–199.

[312] D.R. Stinson, "Combinatorial techniques for universal hashing," preprint, 1990.

[313] D.R. Stinson, "Combinatorial characterizations of authentication codes," *Advances in Cryptology, Proc. Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 62–73.

[314] D.R. Stinson, "Universal hashing and authentication codes," *Advances in Cryptology, Proc. Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 74–85.

[315] R. Struik and J. van Tilburg, "The Rao-Nam scheme is insecure against a chosen-plaintext attack," *Advances in Cryptology, Proc. Crypto'87, LNCS 293*, C. Pomerance, Ed., Springer-Verlag, 1988, pp. 445–457.

[316] R.C. Titsworth, "Optimal ranging codes," *IEEE Trans. Space Elec. Telem.*, Vol. SET-10, March 1964, pp. 19–20.

[317] J. Vandewalle, D. Chaum, W. Fumy, C.J.A. Jansen, P. Landrock, and G. Roelofsen, "A European call for cryptographic algorithms: RIPE; Race Integrity Primitives Evaluation," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 267–271.

[318] K. Van Espen and J. Van Mieghem, *"Evaluatie en Implementatie van Authentiseringsalgoritmen (Evaluation and Implementation of Authentication Algorithms – in Dutch),"* ESAT Laboratorium, Katholieke Universiteit Leuven, Thesis grad. eng., 1989.

[319] Ph. Van Heurck, "Trasec: Belgian security system for electronic funds transfers," *Computers & Security*, Vol. 6, 1987, pp. 261–268.

[320] W. Van Leekwijck and L. Van Linden, *"Cryptografische Eigenschappen van Booleaanse Functies (Cryptographic Properties of Boolean Functions – in Dutch),"* ESAT Katholieke Universiteit Leuven, Thesis grad. eng., 1990.

[321] S. Vaudenay, "FFT-hash-II is not yet collision-free," *Advances in Cryptology, Proc. Crypto'92, LNCS*, E.F. Brickell, Ed., Springer-Verlag, to appear.

[322] G.S. Vernam, "Cipher printing telegraph system for secret wire and radio telegraph communications," *Journal American Institute of Electrical Engineers*, Vol. XLV, 1926, pp. 109-115.

[323] M. Walker, "Security in mobile and cordless telecommunications," *Proc. Compeuro 92, Computer Systems and Software Engineering, 6th Annual European Computer Conference*, IEEE Computer Society Press, 1992, pp. 493–496.

[324] A.F. Webster and S.E. Tavares, "On the design of S-boxes," *Advances in Cryptology, Proc. Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 523–534.

[325] M.N. Wegman and J.L. Carter, "New hash functions and their use in authentication and set equality," *Journal of Computer and System Sciences*, Vol. 22, 1981, pp. 265–279.

[326] J. Williams, "Collisions for two pass Snefru," *Internet News, sci.crypt*, November 8, 1990.

[327] R.S. Winternitz, "Producing a one-way hash function from DES," *Advances in Cryptology, Proc. Crypto'83*, D. Chaum, Ed., Plenum Press, New York, 1984, pp. 203–207.

[328] R.S. Winternitz, "A secure one–way hash function built from DES," *Proc. IEEE Symposium on Information Security and Privacy 1984*, 1984, pp. 88-90.

[329] S. Wolfram, "Random sequence generation by cellular automata," *Advances in Applied Mathematics*, Vol. 7, 1986, pp. 123–169.

[330] G. Xiao and J.L. Massey, "A spectral characterization of correlation-immune combining functions," *IEEE Trans. on Information Theory*, Vol. IT–34, No. 3, 1988, pp. 569–571.

[331] A.C. Yao, "Theory and applications of trapdoor functions," *Proc. 23rd IEEE Symposium on Foundations of Computer Science*, 1982, pp. 80–91.

[332] A.C. Yao, "Computational information theory," in *"Complexity in Information Theory,"* Y.S. Abu-Mostafa, Ed., Springer-Verlag, 1988, pp. 1–15.

[333] R. Yarlagadda and J.E. Hershey, "A note on the eigenvectors of Hadamard matrices of order 2^n," *Linear Algebra & Applications*, Vol. 45, 1982, pp. 43–53.

[334] R. Yarlagadda and J.E. Hershey, "Analysis and synthesis of bent sequences," *IEE Proceedings-E*, Vol. 136, 1989, pp. 112–123.

[335] G. Yuval, "How to swindle Rabin," *Cryptologia*, Vol. 3, 1979, pp. 187–189.

[336] G. Zémor, "Hash functions and graphs with large girths," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 508–511.

[337] G. Zémor, "Hash functions and Cayley graphs," preprint, 1991.

[338] K.C. Zeng, J.H. Yang, and Z.T. Dai, "Patterns of entropy drop of the key in an S-box of the DES," *Advances in Cryptology, Proc. Crypto'87, LNCS 293*, C. Pomerance, Ed., Springer-Verlag, 1988, pp. 438–444.

[339] Y. Zheng, T. Matsumoto, and H. Imai, "Duality between two cryptographic primitives," *Papers of Technical Group for Information Security, IEICE of Japan*, March 16, 1989, pp. 47–57.

[340] Y. Zheng, T. Matsumoto, and H. Imai, "On the construction of block ciphers provably secure and not relying on any unproved hypothesis," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 461–480.

[341] Y. Zheng, T. Matsumoto, and H. Imai, "Connections between several versions of one-way hash functions," *Proc. SCIS90, The 1990 Symposium on Cryptography and Information Security*, Nihondaira, Japan, Jan. 31–Feb.2, 1990.

[342] Y. Zheng, T. Matsumoto, and H. Imai, "Structural properties of one-way hash functions," *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 285–302.

[343] Y. Zheng, T. Matsumoto, and H. Imai, "Duality between two cryptographic primitives," *Proc. 8th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, LNCS 508*, S. Sakata, Ed., Springer-Verlag, 1991, pp. 379–390.

[344] Y. Zheng, T. Hardjono, and J. Pieprzyk, "Sibling intractible function families and their applications," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS*, Springer-Verlag, to appear.

[345] Y. Zheng, personal communication, 1992.

[346] Y. Zheng, J. Pieprzyk, and J. Seberry, "HAVAL — a one-way hashing algorithm with variable length output," *Advances in Cryptology, Proc. Auscrypt'92, LNCS*, Springer-Verlag, to appear.

[347] Ph. Zimmerman, "A proposed standard format for RSA cryptosystems," *Computer*, Vol. 19, No. 9, 1986, pp. 21–34.

[348] G. Zorpette, "Large computers," *IEEE Spectrum*, Vol. 29, No. 1, 1992, pp. 33-35.

[349] G. Zorpette (Ed.), "Special issue on supercomputing," *IEEE Spectrum*, Vol. 29, No. 9, 1992, pp. 26–76.