On the Verification of Cryptographic Protocols

Federico Cerutti

Dipartimento di Ingegneria dell’Informazione, Università di Brescia
Via Branze 38, I-25123 Brescia, Italy

January 11, 2011

Talk at Prof. Giuzzi’s Course: “Algebra for coding theory and cryptography”
Cryptography hides information: send or receive messages in encrypted form

Cryptographic functions can be put to use in more creative ways to design security protocols

[Needham and Schroeder, 1978]’s security goals:
- Establishment of authenticated interactive communication (greeting messages over the phone)
- One-way authentication (both parties could not be available at the same time)
- Signed communication (nonrepudiation feature, the recipient can prove to a third party that the message originated from the claimed sender)
Needham Schroeder Protocol (NS)  
[Needham and Schroeder, 1978]

Notation:
- Principals (e.g. A, B)
- Encryption keys, as shared keys (e.g. $K_{ab}$), public keys (e.g. $K_a$), and the corresponding private keys (e.g. $K_a^{-1}$)
- Statements or formulae (e.g. $N_a$)

Assumption: both parties know S’ public key

Message 1: $A \rightarrow S$: $A, B$
Message 2: $S \rightarrow A$: $\{K_b, B\}_{K_s^{-1}}$
Message 3: $A \rightarrow B$: $\{N_a, A\}_b$
Message 4: $B \rightarrow S$: $B, A$
Message 5: $S \rightarrow B$: $\{K_a, A\}_{K_s^{-1}}$
Message 6: $B \rightarrow A$: $\{N_a, N_b\}_a$
Message 7: $A \rightarrow B$: $\{N_b\}_b$
Important issues in designing of authentication protocols

- The adversary attacking a network protocol can actively interfere (interposition, eavesdropping, alteration, dropping, duplication, rerouting, messages rearranging).
- Encryption is a basic tool for the construction of protocols, but it needs to be used properly in order to achieve the security goal.
- Security goal can be (and should be) defined independently from the underlying cryptographic techniques (symmetric/asymmetric cryptography...) used in the solution.
- Recognize the need for techniques for the verification and analysis of security protocols.

*Protocols such as those developed here are prone to extremely subtle errors that are unlikely to be detected in normal operation. The need for techniques to verify the correctness of such protocols is great, and we encourage those interested in such problems to consider this area. [Needham and Schroeder, 1978]*
Dolev, Yao
Burrows, Abadi, Needham
Delagrande, Grote, Hunter
The first attempt: the Dolev-Yao (DY) model
[Dolev and Yao, 1981]

- Very constrained
- Does not allow to describe many interesting protocols
- But:
  - It is the first paper providing a formal model
  - The adversarial model is still quite general
  - They show that by restricting the class of target protocols one can obtain interesting algorithmic results
- Important requirements:
  - A precise language for the description of protocols
  - A formal execution model that describes how the protocol is run, possibly in the presence of an adversary. This includes a description of the adversary’s capabilities
  - A formal language for specifying desired security protocols
DY Assumptions

- A perfect public key system, where:
  - the one-way functions used are unbreakable
  - the public directory is secure and cannot be tampered with
  - everyone has access to all public keys
  - the private keys are known by the legitimate user

- In a two-party protocol, only the two users who wish to communicate are involved in the transmission process

- Only “active” eavesdroppers, or someone who first taps the communication line to obtain messages and then tries everything he can in order to discover the plaintext. A saboteur:
  - can obtain any message passing through the network
  - is a legitimate user of the network (he can initiate a conversation with any other user)
  - will have the opportunity to be a receiver to any user
Let $\Sigma$ be a finite alphabet of operator symbols and $\Sigma^*$ be the set of all finite strings over $\Sigma$ including the empty word $\lambda$.

$\Sigma = \{d\} \cup \{E_X, D_X, i_X, d_X | X \text{ is a user name}\}$

Reduction rules

$$E_X D_X \rightarrow \lambda \quad D_X E_X \rightarrow \lambda \quad d_X i_X \rightarrow \lambda \quad di_X \rightarrow \lambda$$

A Ping-pong protocol $P(S, R)$ between the users $S$ and $R$ is a sequence of strings $\alpha_1, \alpha_2, \ldots, \alpha_l$ such that $\alpha_i \in \Sigma_S^*$ if $i$ is odd and $\alpha_i \in \Sigma_R^*$ otherwise. [Dolev et al., 1982]

$$\Delta = (\Sigma_Z \cup \{\alpha_i(X, Y) | 1 \leq i \leq l, X \text{ and } Y \text{ are different users}\})^*$$

is an operator-words of the adversary.

Let $\alpha_1(S, R)$ be the first word of $P(S, R)$ and $Z$ an adversary. $P$ is insecure, if there exists a string $\gamma \in \Delta$ such that $\overline{\gamma \alpha_1} = \lambda$
Example ([Stamer, 2005])

1. \( A \rightarrow B : \{\{m\}_K^B, A\}^K_B, \alpha_1(A, B) = E_B^i_A E_B \)

2. \( B \rightarrow A : \{\{m\}_K^A, B\}^K_A, \alpha_2(A, B) = E_A^i_B E_A D_B d_A D_B \)

\[ \gamma' = D_Z d_A D_Z E_Z i_A E_Z D_A d_A D_A E_A i_Z d_D Z Z(3) \]

\[ \gamma' \alpha_2 \alpha_1 = \lambda \quad \gamma = \gamma' \alpha_2 \quad \overline{\gamma \alpha_1} = \lambda \]

Theorem ([Dolev et al., 1982])

For “Ping-Pong” protocols there exists a security checking algorithm whose input are the generic cancellation rules and the protocol. The time-complexity is \( O(n^3) \), where \( n \) is the length of the input.
DY explained [Stamer, 2005]

1.1 $A \rightarrow B$: $\{{{m}\}_{K_b}, A\}_{K_b}$

1.2 $B \rightarrow Z/A$: $\{{{m}\}_{K_a}, B\}_{K_a}$

2.1 $Z \rightarrow A$: $\{{{\{m\}_{K_a}, B\}_{K_a}, Z\}_{K_a}}$

2.2 $A \rightarrow Z$: $\{{{\{m\}_{K_a}, B\}_{K_z}, A\}_{K_z}}$

3.1 $Z \rightarrow A$: $\{{{m}\}_{K_a}, Z\}_{K_a}$

3.2 $A \rightarrow Z$: $\{{{m}\}_{K_z}, Z\}_{K_z}$

At the end, $Z$ knows $m$. 
First paper that provides a formal model

Protocol restrictions are mostly on honest participants and the protection goal

The adversarial model is still quite general

Theoretical results
  - Simple security characterization for a subclass of protocols
  - Security problem decidable in polynomial time

Not general enough

Difficult to extend

It does not consider the evolution of the message exchanging
A Logic of Authentication (BAN) [Burrows et al., 1990]

- A more general approach
- Goal of authentication (informally and imprecisely): after authentication, two principals (people, computers, services) should be entitled to believe that they are communicating with each other and not with intruders
- BAN helps in answering the following questions:
  - Does this protocol work? Can it be made to work?
  - Exactly what does this protocol achieve?
  - Does this protocol need more assumptions than another protocol?
  - Does this protocol do anything unnecessary?
- BAN assumes totally corrected implementations of protocols and cryptosystems
- BAN is focused on the beliefs of trustworthy parties involved in the protocols and on the logic evolution of these beliefs as a consequence of communication
What is Logic

- A way for formalizing knowledge through sentences...
  - The sentences have to be expressed according to the syntax of the representation language
  - We need the semantics of the language, which defines the truth of each sentence with respect to each possible world (or model)
- ...and for reasoning over this knowledge
  - Entailment relation: $\alpha \models \beta$ means that the sentence $\alpha$ entails the sentence $\beta$, or that in every model in which $\alpha$ is true, $\beta$ is also true
All men are mortal, Socrates is a man, therefore Socrates is mortal.

\[
\begin{align*}
\text{MP} & \quad \text{man}(X) \quad \text{man}(X) \rightarrow \text{mortal}(X) \\
& \quad \text{mortal}(X)
\end{align*}
\]

\[\text{man}(Socrates) \models \text{mortal}(Socrates)\]

In every model (or in every possible world) where Socrates actually is a man, then actually Socrates is mortal.

\[
\begin{array}{ccc}
\text{man}(X) & \text{mortal}(X) & \text{man}(X) \rightarrow \text{mortal}(X) \\
T & T & T \\
T & F & F \\
F & T & T \\
F & F & F
\end{array}
\]
Definition (Conditions for $L$ being a formal theory)

1. A countable set of symbols is given as the symbols of $L$; every its finite sequence of symbols is called an *expression* of $L$.

2. There is a subset of the set of expressions of $L$ called the set of *well-formed formulas* (wfs) of $L$. There is usually an effective procedure to determine whether a given expression is a wf.

3. There is a set of wfs called the set of *axioms* of $L$. If one can effectively decide whether a given wf is an axiom, then $L$ is called an *axiomatic* theory.

4. There is a finite set $R_1, \ldots, R_n$ of relations among wfs, called *rules of inference*. For each $R_i$, there is a unique positive integer $j$ such that, for every set of $j$ wfs and each wf $B$, one can effectively decide whether the given $j$ wfs are in the relation $R_i$ to $B$, and, if so, $B$ is said to *follow from* or to be a *direct consequence of* the given wfs by virtue of $R_i$. 

© 2011 Federico Cerutti <federico.cerutti@ing.unibs.it>
Definition

A *proof* in a formal theory $L$ is a sequence $B_1, \ldots, B_j$ of wfs such that, for each $i$ either $B_i$ is an axiom of $L$ or $B_i$ i- a direct consequence of some of the preceding wfs in the sequence by virtue of one of the rules of inference of $L$.

Definition

A *theorem* of $L$ is a wf $B$ of $L$ such that $B$ is the last wf of some proof in $L$. Such a proof is called a *proof of $B$ in $L$*. 
**Definition**

A wf $B$ is said to be a *consequence* in $L$ of a set $\Gamma$ of wfs if and only if there is a sequence $B_1, \ldots, B_k$ of wfs such that $C$ is $B_k$ and, for each $i$, either $B_i$ is an axiom or $B_i$ is $\Gamma$, or $B_i$ is a direct consequence by some rule of inference of some of the preceding wfs in the sequence. Such a sequence is called a *proof* (or a *deduction*) of $C$ *from* $\Gamma$. The members of $\Gamma$ are called the *hypotheses* or *premises* of the proof. We use $\Gamma \vdash C$ as an abbreviation for “$C$ is a consequence of $\Gamma$”.

The following are simple properties of the notion of consequence:

1. If $\Gamma \subseteq \Delta$ and $\Gamma \vdash C$, then $\Delta \vdash C$;
2. $\Gamma \vdash C$ if and only if there is a finite subset $\Delta$ of $\Gamma$ such that $\Delta \vdash C$;
3. If $\Delta \vdash C$, and for each $B$ in $\Delta$, $\Gamma \vdash B$, then $\Gamma \vdash C$. 
The symbols of $L_1$ are $\neg$, $\Rightarrow$, $(,)$, $\land$, $\lor$, and the letters $A_i$ with positive integers $i$ as subscripts: $A_1, A_2, A_3, \ldots$. The symbols $\neg$ and $\Rightarrow$ are called *primitive connectives*, and the letters $A_i$ are called *statement letters*.

1. All statement letters are well-formed formulas (wfs).
2. If $B$ and $C$ are wfs, then so are $(\neg B)$, $(B \Rightarrow C)$, $(B \land C)$, and $(B \lor C)$.
3. $C$ is a wfs if and only if there is a finite sequence $B_1, \ldots, B_n$, ($n \geq 1$) such that $B_n = C$ and, if $1 \leq i \leq n$, $B_i$ is either a wfs or a negation ($B_i = (\neg B_k)$, $k < i$), or conditional ($B_i = (B_j \Rightarrow B_l)$, $j < i$, $l < i$), or a conjunction, or a disjunction constructed from previous expression in the sequence.
If $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ are wfs of $L_1$, then the following are axioms of $L_1$: 

1. $\mathcal{B} \Rightarrow (\mathcal{C} \Rightarrow \mathcal{B})$;
2. $(\mathcal{B} \Rightarrow (\mathcal{C} \Rightarrow \mathcal{D})) \Rightarrow ((\mathcal{B} \Rightarrow \mathcal{C}) \Rightarrow (\mathcal{B} \Rightarrow \mathcal{D}))$;
3. $(\mathcal{B} \land \mathcal{C}) \Rightarrow \mathcal{B}$;
4. $(\mathcal{B} \land \mathcal{C}) \Rightarrow \mathcal{C}$;
5. $\mathcal{B} \Rightarrow (\mathcal{C} \Rightarrow (\mathcal{B} \land \mathcal{C}))$;
6. $\mathcal{B} \Rightarrow (\mathcal{B} \lor \mathcal{C})$;
7. $\mathcal{C} \Rightarrow (\mathcal{B} \lor \mathcal{C})$;
8. $(\mathcal{B} \Rightarrow \mathcal{D}) \Rightarrow ((\mathcal{C} \Rightarrow \mathcal{D}) \Rightarrow (\mathcal{B} \lor \mathcal{C} \Rightarrow \mathcal{D}))$;
9. $(\mathcal{B} \Rightarrow \mathcal{C}) \Rightarrow ((\mathcal{B} \Rightarrow \neg \mathcal{C}) \Rightarrow \neg \mathcal{B})$;
10. $\neg \neg \mathcal{B} \Rightarrow \mathcal{B}$.

The only rule of inference of $L_1$ is modus ponens: $\mathcal{C}$ is a direct consequence of $\mathcal{B}$ and $(\mathcal{B} \Rightarrow \mathcal{C})$. 
The BAN formalism

- Logic with principals \((P, Q, R\ldots)\), encryption keys \((K\ldots)\) and formulas or statements \((X, Y, \ldots)\). Conjunction is the only propositional connective

- **P believes** \(X\), the principal \(P\) may act as though \(X\) is true

- **P sees** \(X\), someone has sent a message containing \(X\) to \(P\), who can read and repeat \(X\)

- **P said** \(X\), \(P\) at some time sent a message including the statement \(X\); it is known that \(P\) believed \(X\) then

- **P controls** \(X\), \(P\) is an authority on \(X\) and should be trusted on this matter

- **fresh** \(X\), \(X\) has not been sent in a message at any time before the current run of the protocol

- \(P \overset{K}{\leftrightarrow} Q\), \(P\) and \(Q\) may use the shared key \(K\) to communicate

- \(K \overset{P}{\rightarrow}\), \(P\) has public key \(K\) (and the relative private key \(K^{-1}\) which is good

- \(P \overset{X}{\Rightarrow} Q\), the formula \(X\) is a secret known only to \(P\) and \(Q\), and possibly to principals trusted by them

- \(\{X\}_K\), the formula \(X\) is encrypted under the key \(K\)

- \(\langle X\rangle_Y\), the formula \(X\) is combined with the formula \(Y\), it is intended that \(Y\) be a secret and that its presence prove the identity of whoever utters \(\langle X\rangle_Y\)
Logical Postulates: preliminaries

- Two epochs:
  - **present**: that begins at the start of the particular run of the protocol under consideration
  - **past**: indeed all messages sent before the present are considered to be in the past and should not be accepted as recent

- All beliefs held in the present are stable for the entirety of the protocol run

- Beliefs held in the past are not necessarily carried forward into the present

- A message cannot be understood by a principals who does not know the key

- The key cannot be deduced from the encrypted message

- Each encrypted message contains sufficient redundancy to allow a principal who decrypts it to verify that he has used the right key
The postulates

1. Message-meaning

\[
P \text{ believes } \overset{K}{\leftrightarrow} Q \quad P \text{ sees } \{X\}_{K^{-1}}
\]

\[
P \text{ believes } Q \text{ said } X
\]

2. Nonce-verification

\[
P \text{ believes fresh } X \quad P \text{ believes } Q \text{ said } X
\]

\[
P \text{ believes } Q \text{ believes } X
\]

3. Jurisdiction

\[
P \text{ believes } Q \text{ controls } X \quad P \text{ believes } Q \text{ believes } X
\]

\[
P \text{ believes } X
\]

4. Composition

\[
\ldots
\]

5. Freshness of composed messages with nonces

\[
\ldots
\]
Needham Schroeder Protocol (NS)  
[Needham and Schroeder, 1978]

Notation:
- Principals (e.g. A, B)
- Encryption keys, as shared keys (e.g. $K_{ab}$), public keys (e.g. $K_a$), and the corresponding private keys (e.g. $K_a^{-1}$)
- Statements or formulae (e.g. $N_a$)

Assumption: both parties know S’ public key

Message 1: $A \rightarrow S$: $A, B$  
Message 2: $S \rightarrow A$: $\{K_b, B\}_{K_s^{-1}}$  
Message 3: $A \rightarrow B$: $\{N_a, A\}_{K_b}$  
Message 4: $B \rightarrow S$: $B, A$  
Message 5: $S \rightarrow B$: $\{K_a, A\}_{K_s^{-1}}$  
Message 6: $B \rightarrow A$: $\{N_a, N_b\}_{K_a}$  
Message 7: $A \rightarrow B$: $\{N_b\}_{K_b}$
**NS idealized**

<table>
<thead>
<tr>
<th>Original</th>
<th>Idealized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message 1: $A \rightarrow S$: $A, B$</td>
<td></td>
</tr>
<tr>
<td>Message 2: $S \rightarrow A$: ${K_b, B}_{K_s^{-1}}$</td>
<td>$S \rightarrow A$: ${\rightarrow B}_{K_s^{-1}}$</td>
</tr>
<tr>
<td>Message 3: $A \rightarrow B$: ${N_a, A}_{K_b}$</td>
<td>$A \rightarrow B$: ${N_a}_{K_b}$</td>
</tr>
<tr>
<td>Message 4: $B \rightarrow S$: $B, A$</td>
<td>$S \rightarrow B$: ${\rightarrow A}_{K_s^{-1}}$</td>
</tr>
<tr>
<td>Message 5: $S \rightarrow B$: ${K_a, A}_{K_s^{-1}}$</td>
<td></td>
</tr>
<tr>
<td>Message 6: $B \rightarrow A$: ${N_a, N_b}_{K_a}$</td>
<td>$B \rightarrow A$: ${\langle A \rightleftharpoons B\rangle_{N_a}}_{K_a}$</td>
</tr>
<tr>
<td>Message 7: $A \rightarrow B$: ${N_b}_{K_b}$</td>
<td>$A \rightarrow B$: ${\langle A \rightleftharpoons B\rangle_{N_b}}_{K_b}$</td>
</tr>
</tbody>
</table>

- Messages 1 and 4 do not contribute to the logical properties.
- In message 3, $N_a$ is not known by $B$ and thus is not being used to prove the identity of $A$.
- In messages 6 and 7, $N_a$ and $N_b$ are used as secrets.
NS analyzed

\begin{align*}
A \text{ believes } & K_a \rightarrow A \\
B \text{ believes } & K_b \rightarrow B \\
A \text{ believes } & K_s \rightarrow S \\
B \text{ believes } & K_s \rightarrow B \\
S \text{ believes } & K_s \rightarrow A \\
S \text{ believes } & K_s \rightarrow S
\end{align*}

\begin{align*}
A \text{ believes } & \forall K \ S \text{ controls } K \rightarrow B \\
B \text{ believes } & \forall K \ S \text{ controls } K \rightarrow A \\
A \text{ believes fresh } & N_a \\
B \text{ believes fresh } & N_b \\
A \text{ believes } & A \nleftrightarrow B \\
B \text{ believes } & A \nleftrightarrow B
\end{align*}

\begin{align*}
A \text{ believes fresh } & K_b \rightarrow B \\
B \text{ believes fresh } & K_a \rightarrow A
\end{align*}

\begin{align*}
A \text{ believes } & K_b \rightarrow B \\
A \text{ believes } & B \text{ believes } A \nleftrightarrow B \\
B \text{ believes } & K_a \rightarrow A \\
B \text{ believes } & A \text{ believes } A \nleftrightarrow B
\end{align*}
### BAN Explained [Burrows et al., 1989]

<table>
<thead>
<tr>
<th>where</th>
</tr>
</thead>
<tbody>
<tr>
<td>A believes ( S ) controls ( K_b \mapsto B )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(1) Message Meaning</th>
<th>( K_s \mapsto S )</th>
<th>A sees ( { \mapsto B }^{-1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[assumption]</td>
<td>A believes ( K_b \mapsto B )</td>
<td>A believes ( S ) said ( K_b \mapsto B )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(2) Nonce verification</th>
<th>[assumption] A believes fresh ( K_b \mapsto B )</th>
<th>((1)) A believes ( S ) said ( K_b \mapsto B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[assumption]</td>
<td>A believes ( S ) believes ( K_b \mapsto B )</td>
<td>((2)) A believes ( S ) believes ( K_b \mapsto B )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jurisdiction rule</th>
<th>[assumption] A believes ( S ) controls ( K_b \mapsto B )</th>
<th>((2)) A believes ( S ) believes ( K_b \mapsto B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A believes ( K_b \mapsto B )</td>
<td>A believes ( K_b \mapsto B )</td>
<td></td>
</tr>
</tbody>
</table>
BAN Conclusions

- First logic for protocol verification
- Description of proof systems
- BAN enables us to exhibit step by step how beliefs are built up to the point of mutual authentication
- It can guide in identifying mistakes and in suggesting corrections
- It lacks on semantics [Abadi and Tuttle, 1991, Mao and Boyd, 1993, Delgrande et al., 2009]
- It is not in a pure declarative form rather it has to idealize a protocol
- It cannot reveal some kinds of attack (like reflection attack)
New Idea in Logic Verification of Cryptographic Protocols [Delgrande et al., 2009]

- Standard specification of protocols is both incomplete and imprecise
- The Goal of the protocol in not explicitly given (incomplete), therefore it is difficult to determine if a given trace constitutes an attack
- It is not clear what $A \rightarrow B$ means (imprecise): it does not means that $A$ successfully sends the message to $B$; instead:
  - $A$ intends to send the message to $B$
  - $A$ actually sends the message to someone
- Not only beliefs but also intentions
- Aim: define a logic program in which the answer sets correspond to sequences of exchanged messages between agents
A logic program

Definitions:

- It is a picture of why and how you believe a program/policy will work
- The foundation of program planning and the key tool of program evaluation
- A series of “if-then” relationships that, if implemented as intended, lead to the desired outcomes

Example ([Lamperti, 2011])

\[
speak(al, russian).
speak(robert, english).
speak(mary, russian).
speak(mary, english).
communicate(X, Y) \leftarrow speak(X, L), speak(Y, L), X \neq Y.
\]
Declarative programming based on the stable model semantics of logic programming [Gelfond and Lifschitz, 1988], autoepistemic logic [Moore, 1985], and default logic [Reiter, 1980]

- **Literals**: $p(\hat{t}), \neg p(\hat{t})$ (\hat{t} is a term)
- **Rules**: $l_0 \text{ or } \ldots \text{ or } l_k \leftarrow l_{k+1}, \ldots, l_m, \neg l_{m+1}, \ldots, \neg l_n$.
  - \text{or} is the epistemic disjunction
  - \text{not} is default negation (aka negation as failure): literals possibly preceded by default negation are called extended literals
- **Constraint**: $\leftarrow l_0, \ldots, l_n$.
- **Fact**: $l_0 \text{ or } \ldots \text{ or } l_k$.
- **Choice rules**: $i \{p, q\} j$ (at least $i$ of the enclosed atoms are true, but not more that $j$)
- **Conditions**: $p(X) : q(X)$ (e.g. if we have $q(a), q(b)$, then we can instantiate $p(a), p(b)$)
The Answer Set \{Programming | Prolog\} (ASP) in a Nutshell [Gelfond, 2008]

**Definition**

A program of Answer Set Prolog is a pair \{\sigma, \Pi\} where \sigma is a signature and \Pi is a collection of logic programming rules over \sigma. Given a logic program \Pi, the corresponding signature is denoted by \sigma(\Pi). If \sigma(\Pi) is not explicitly given, it is assumed it consists of symbols occurring in the program.

Terms, literals, and rules of \Pi are called ground if they contain no variables and no symbols for arithmetic functions. A rule \(r'\) is called a ground instance of a rule \(r\) of \(\Pi\) if it is obtained from \(r\) by:

1. replacing \(r\)'s non-integer variables by properly typed ground terms of \(\sigma(\Pi)\)
2. replacing \(r\)'s variables for non-negative integers by numbers
3. replacing the remaining occurrences of numerical terms by their values
**ASP partial interpretation [Gelfond, 2008]**

**Definition (Partial interpretation of \( \sigma \))**

A set of ground literals \( G = \{l_0, \ldots, l_k\} \) is consistent iff \( \nexists i, j \text{ s.t. } l_i = \neg l_j \). Consistent sets of ground literals over \( \sigma \) are called partial interpretations of \( \sigma \).

Given a partial interpretation \( S \):

- \( l \) ground literal, \( \bar{l} = \neg l \), \( l \) is **true** if \( l \in S \), **false** if \( \bar{l} \in S \), **unknown** otherwise
- **not** \( l \) extended literal is **true** if \( l \notin S \), **false** otherwise
- conjunction of extended literals \( U \) is **true** if all elements of \( U \) is **true** in \( S \), **false** if at least one element of \( U \) is **false** in \( S \), **unknown** otherwise
- disjunction of literals \( D \) is **true** if at least one of its members is **true** in \( S \), **false** if all members of \( D \) is **false** in \( S \), **unknown** otherwise
ASP: negation as failure [Gelfond, 2008]

- Non-monotonic inference rule in logic programming
- \( \text{not } p \) is derived from failure of deriving \( p \)

**Example**

cross_railroad ← \( \neg \) train_approaches.

“If you know that the train not approaches, then cross the railroad”.

cross_railroad ← not train_approaches.

“If you don’t known if the train approaches or not, then cross the railroad”. Incomplete information!

**Example**

\( \text{fly} (X) ← \text{bird} (X), \text{not} \text{abnormal} (X) \).

\( \text{abnormal} (X) ← \text{penguin} (X) \).

“Each bird flies, except abnormal ones”. Defaults!
In classical logic, $A \lor B$ forbids that both $A$ and $B$ are false.

In epistemic logic, $A$ or $B$ means that every possible set of reasoner’s belief must satisfy $A$ or satisfy $B$.

**Example**

```
node(X) ← arc(X, Y).
node(Y) ← arc(X, Y).
color(X, red) or color(X, green) or color(X, blue) ← node(X).
```

I know that an arc is between two nodes of a graph, and I know that each node has a colour that is one of the following, red, green, or blue.
ASP: the answers set semantics [Gelfond, 2008]

Definition

A partial interpretation $S$ of $\sigma(\Pi)$ is an answer set for $\Pi$ if $S$ is minimal (w.r.t. set inclusion) partial interpretation satisfying the rules of $\Pi$.

An answer set corresponds to a possible set of beliefs which can be built by a rational reasoner on the basis of rules of $\Pi$. In the construction of such a set, $S$, the reasoner is assumed to be guided by the following informal principles:

- $S$ must satisfy the rules of $\Pi$
- the reasoner should adhere to the rationality principle which says that one shall not believe anything one is not forced to believe (minimality)
An example in ASP

Example ([Eiter, 2008])

\[\text{person}(\text{joey}).\]
\[\text{male}(X) \text{ or } \text{female}(X) \leftarrow \text{person}(X).\]
\[\text{bachelor}(X) \leftarrow \text{male}(X), \text{ not married}(X).\]

Two answer sets:

1. \{\text{person}(\text{joey}), \text{male}(\text{joey}), \text{bachelor}(\text{joey})\}
2. \{\text{person}(\text{joey}), \text{female}(\text{joey})\}
The DGH approach

Three independent modules:

1. Protocol module: it includes generic information about message passing and protocols
2. Intruder module: it includes a specification of the intruders capabilities
3. Instance module: it includes the structure of a specific protocol
The DGH approach: Protocol module

- General message passing framework
- Principals’ holdings, capabilities, ...
  - Keys: asymmetric and symmetric key encryption
  - Nonces or timestamp
  - Actions (e.g. send and receive)
  - Time (discrete): actions occur at some point in time (to keep the search space manageable, a single time step is used for one agent to receive a message, decrypt its contents, compose a reply, encrypt and send it)
- Notion of appropriateness: matching of messages
- Two auxiliary predicates:
  1. talked: to indicate that two agents have successfully communicated
  2. authenticated: is part of goal specification
The DGH approach: Protocol module

\[
\text{send}(A, B, M, T) :- \  \text{wants}(A, \text{send}(A, B, M), T), \ \text{has}(A, M, T).
\]

\[
\text{receive}(A, B, M, T+1) :- \ \text{send}(B, A, M, T), \ \text{not intercept}(M, T+1)
\]

\[
\{ \text{wants}(A, \text{send}(A, B, M), T) \} 1 :- \ \text{receive}(A, B, M2, T), \ \text{fit}(\text{msg}(J, B, A, M2), \text{msg}(J+1, A, B, M))
\]

\[
\text{talked}(A, B, T+1) :- \ \text{send}(A, B, M, T), \ \text{receive}(B, A, M, T+1).
\]

\[
\text{authenticated}(A, B, T) :- \ \text{send}(A, B, \text{enc}(M1, K1), T1), \ \text{fresh}(A, \text{nonce}(A, Na), T1), \ \text{part_m}(\text{nonce}(A, Na), M1), \ \text{key_pair}(K1, Kinv1), \ \text{has}(A, K1, T1), \ \text{has}(B, Kinv1, T1), \ \text{not has}(C, Kinv1, T1) : \ \text{agent}(C) : C \neq B, \ \text{send}(B, A, \text{enc}(M2, K2), T2), \ \text{receive}(A, B, \text{enc}(M2, K2), T), \ \text{part_m}(\text{nonce}(A, Na), M2), \ \text{key_pair}(K2, Kinv2), \ \text{has}(B, K2, T1), \ \text{has}(A, Kinv2, T1), \ \text{not has}(C, Kinv2, T1) : \ \text{agent}(C) : C \neq A, T1 < T2, T2 < T.
\]
The DGH approach: Intruder module

- This module specifies all aspects of the intruder
- The intruder model is flexible and easy to modify
- Holdings:
  - the public key of every agent
  - a public-private key pair (in order to pretend to be an honest agent)
- Capabilities:
  - intercept messages and it receives the messages that it intercepts
  - send messages whenever it wants to
  - fake the sender name of messages it sends

\[
\begin{align*}
0 \{ \text{intercept}(M, T+1) \} 1 & : - \text{send}(A, B, M, T). \\
\text{receive}(I, A, M, T+1) & : - \text{send}(A, B, M, T), \\
& \quad \text{intercept}(M, T+1). \\
1 \{ \text{receive}(A, B, M, T+1) : \text{principal}(B) \} 1 & : - \text{send}(I, A, M, T).
\end{align*}
\]
The DGH approach: Instance module
Messages (NS simplified case)

msg(1, A, B, enc(m(nonce(C, Na), principal(A)), pub_key(B))).
msg(2, B, A, enc(m(nonce(C, Na), nonce(D, Nb)), pub_key(A))).
msg(3, A, B, enc(m(nonce(D, Nb)), pub_key(B))).
Goals: both principals should believe that the other one is authenticated and that they actually are

\[
\text{goal}(A, B, T) :- \text{authenticated}(A, B, T),
\text{believes}(A, \text{authenticated}(A, B), T),
\text{authenticated}(B, A, T),
\text{believes}(B, \text{authenticated}(B, A), T).
\]

Attacks: run of the protocol that causes an agent to believe the goal is true, when in fact the goal is not true

\[
\text{attack} :- \text{believes}(A, \text{authenticated}(A, B), T),
\text{not authenticated}(A, B, T).
\]
DGH Executed

send(a, i, enc(m(nonce(a, 1), principal(a)), pub_key(i)), 0)

receive(i, a, enc(m(nonce(a, 1), principal(a)), pub_key(i)), 1)
send(i, b, enc(m(nonce(a, 1), principal(a)), pub_key(b)), 1)

receive(b, a, enc(m(nonce(a, 1), principal(a)), pub_key(b)), 2)
send(b, a, enc(m(nonce(a, 1), nonce(b, 1)), pub_key(a)), 2)

receive(i, b, enc(m(nonce(a, 1), nonce(b, 1)), pub_key(a)), 3)
send(i, a, enc(m(nonce(a, 1), nonce(b, 1)), pub_key(a)), 3)

authenticated(a, i, 4)
believes(a, authenticated(a, i), 4)
believes(a, completed(a, i), 4)
receive(a, i, enc(m(nonce(a, 1), nonce(b, 1)), pub_key(a)), 4)
send(a, i, enc(m(nonce(b, 1)), pub_key(i)), 4)

authenticated(a, i, 5)
believes(a, authenticated(a, i), 5)
believes(a, completed(a, i), 5)
receive(i, a, enc(m(nonce(b, 1)), pub_key(i)), 5)
send(i, b, enc(m(nonce(b, 1)), pub_key(b)), 5)

authenticated(a, i, 6)
believes(a, authenticated(a, i), 6)
believes(a, completed(a, i), 6)
believes(b, authenticated(b, a), 6)
believes(b, completed(b, a), 6)
receive(b, a, enc(m(nonce(b, 1)), pub_key(b)), 6)
Two parallel protocol runs.

1.1 $A \rightarrow I$: $\{N_a, A\}_{K_i}$

1.2 $I \rightarrow A$: $\{N_a, N_b\}_{K_a}$

1.3 $A \rightarrow I$: $\{N_b\}_{k_i}$

2.1 $I_A \rightarrow B$: $\{N_a, A\}_{K_b}$

2.2 $B \rightarrow I_A$: $\{N_a, N_b\}_{K_a}$

2.3 $I_A \rightarrow B$: $\{N_b\}_{k_b}$

$I$ knows $N_b$. 
Conclusions

- Cryptography is not enough
- Protocols use cryptography to reach goals like authentication
- The verification of protocols is still an open challenge
- DY
  - ’80
  - First attempt
  - Computable algorithm
- BAN Logic
  - ’90
  - Specific logic for authentication
  - Focused on beliefs
- DGH
  - 2009
  - “General purpose” logic: reuse of results in ASP theory
  - Deal with both beliefs and intentions
- More on [Chen et al., 2008]
A semantics for a logic of authentication (extended abstract).
In Proceedings of the tenth annual ACM symposium on Principles of distributed computing, PODC ’91, pages 201–216, New York, NY, USA. ACM.

Jape.
http://jape.comlab.ox.ac.uk:8080/jape/.

A logic of authentication.

A logic of authentication, rep. 39.

Secure transaction protocol analysis: models and applications.
Springer-Verlag, Berlin, Heidelberg.

A general approach to the verification of cryptographic protocols using answer set programming.
On the security of ping-pong protocols.

On the security of public key protocols.
Technical report, Stanford, CA, USA.

On the security of public key protocols.

Answer set programming in a nutshell.
http://gradlog.informatik.uni-freiburg.de/gradlog/slides_ak/eiter_asp.pdf.

Chapter 7 answer sets.

The stable model semantics for logic programming.
In *ICLP/SLP*, pages 1070–1080.
Introduction to Metamathematics.

Programmazione logica.

[Leone et al., ] Leone, N., Pfeifer, G., and Faber, W.
The dlv project.
http://www.dbai.tuwien.ac.at/proj/dlv/.

What is answer set programming?

Towards formal analysis of security protocols.


http://cseweb.ucsd.edu/classes/sp05/cse208/index.html.
Semantical considerations on nonmonotonic logic.

Using encryption for authentication in large networks of computers.

A logic for default reasoning.

Verification of cryptographic protocols part 1: The dolev-yao model and “ping-pong” protocols.
http://www.theory.informatik.uni-kassel.de/~stamer/RewritingOverview_slides.pdf.