Chinese Remaindering with Errors

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
oded@wisdom.weizmann.ac.il.*

Dana Ron
Department of Electrical Engineering – Systems
Tel Aviv University
danar@eng.tau.ac.il.†

Madhu Sudan
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
madhu@mit.edu.‡

Abstract

The Chinese Remainder Theorem states that a positive integer m is uniquely specified by its remainder modulo k relatively prime integers $p_1, \ldots, p_k$, provided $m < \prod_{i=1}^k p_i$. Thus the residues of m modulo relatively prime integers $p_1 < p_2 < \cdots < p_n$ form a redundant representation of m if $m < \prod_{i=1}^k p_i$ and $k < n$. This suggests a number-theoretic construction of an "error-correcting code" that has been implicitly considered often in the past. In this paper we provide a new algorithmic tool to go with this error-correcting code: namely, a polynomial-time algorithm for error-correction. Specifically, given n residues $r_1, \ldots, r_n$ and an agreement parameter t, we find a list of all integers $m < \prod_{i=1}^k p_i$ such that $(m \bmod p_i) = r_i$ for at least t values of $i \in \{1, \ldots, n\}$, provided $t = \Omega(\sqrt{kn\frac{\log p_n}{\log p_1}})$. We also give a simpler algorithm, with a nearly linear time implementation, to decode from a smaller number of errors, i.e., when $t > n - (n-k)\frac{\log p_1}{\log p_1 + \log p_n}$. In such a case there is a unique integer which has such agreement with the sequence of residues.

One consequence of our result is a strengthening of the relationship between average-case complexity of computing the permanent and its worst-case complexity. Specifically we show that if a polynomial time algorithm is able to guess the permanent of a random $n \times n$ matrix on $2n$-bit integers modulo a random n-bit prime with inverse polynomial success rate, then then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$. Previous results of this nature typically worked over a fixed prime moduli or assumed success probability very close to one (as opposed to bounded away from zero).

1 Introduction

The Chinese Remainder Theorem states that a positive integer m is uniquely specified by its remainder modulo k relatively prime integers $p_1, \ldots, p_k$, provided $m < \prod_{i=1}^k p_i$. Thus if we pick $n > k$ relatively prime integers $p_1 < \cdots < p_n$ such that $m < \prod_{i=1}^k p_i$, then the remainders of m modulo the p_i's form a redundant encoding of m. Specifically, m can be recovered given any k of the n remainders. Thus this representation of integers yields a natural error-correcting code: given any two integers $m, m' < \prod_{i=1}^k p_i$, the sequences $\{(m \bmod p_1), \ldots, (m \bmod p_n)\}$ and $\{(m' \bmod p_1), \ldots, (m' \bmod p_n)\}$ differ in at least $n - k + 1$ coordinates.

This redundancy property of the Chinese remainder representation has been exploited often in theoretical computer science. The Karp-Rabin pattern matching algorithm is based on this redundancy [20]. This representation was used to show the strength of probabilistic communication over deterministic communication protocols (cf. [23, Exercise 3.6]). The representation allows for easy arithmetic — addition, multiplication, subtraction and division — on large integers and was even proposed as a potential representation for numbers in computers[1]. The ability to reduce computation over large integers to that over small integers is also employed in complexity-theoretic settings, with a notable example being its use in showing the hardness of computing the permanent of 0/1 matrices [40].

The redundancy of the Chinese remainder representation of integers and its similarity to error-correcting codes raises a natural algorithmic question:

> Given a sequence of integers $\langle r_1, \ldots, r_n \rangle$ that are obtained from taking residues of an integer $m < \prod_{i=1}^k p_i$ modulo relatively prime integers $p_1 < \cdots < p_n$, where some of the residues are erroneous, can we find m?

If the number of residues that are erroneous is less than $\frac{n-k}{2}$, then m is uniquely specified by the vector $\langle r_1, \ldots, r_n \rangle$. However this fact is not algorithmic – it is not clear how to recover m in polynomial time (i.e., in time polynomial in n

*Work done in part while visiting MIT, and partially supported by DARPA grant DABT63-96-C-0018.

†Work done in part while visiting MIT, supported by an ONR Science Scholar Fellowship of the Bunting Institute.

‡Research supported in part by a Sloan Foundation Fellowship and a MIT-NEC Research Initiation Grant.

[1] Unfortunately, it does not allow for easy inequality comparisons — which is presumably why it was not employed.

and $\log p_n$). Even in the case where the number of errors e is larger (but not larger than $n - \sqrt{nk}$), there exists a small list containing all integers whose Chinese remainder representations differ from the vector $\langle r_1, \ldots, r_n \rangle$ in at most e coordinates [16]. Again it is not clear how to recover this list in polynomial time.

In this paper we present efficient algorithms for solving the above problems. Specifically we provide polynomial-time algorithms for the following two tasks:[2]

1. Unique Decoding: Given n relatively prime integers $p_1 < \cdots < p_n$; n residues $r_1, \ldots, r_n$, with $0 \leq r_i < p_i$; and an integer k; find an integer $m < \prod_{i=1}^{k} p_i$ satisfying $(m \bmod p_i) \neq r_i$ for at most $(n-k)\frac{\log p_1}{\log p_1 + \log p_n}$ values of $i \in \{1, \ldots, n\}$, if such an integer exists. (Theorem 6.)

2. List Decoding (for large error): Given n relatively prime integers $p_1 < \cdots < p_n$; n residues $r_1, \ldots, r_n$, with $0 \leq r_i < p_i$; and an integer k; construct a list of all integers m satisfying $m < \prod_{i=1}^{k} p_i$ and $(m \bmod p_i) = r_i$ for at least $\sqrt{2n(k+2)\frac{\log p_n}{\log p_1}} + \frac{k+3}{2} + 2\log n = \Theta(\sqrt{nk\frac{\log p_n}{\log p_1}})$ values of $i \in \{1, \ldots, n\}$. (Theorem 11.) (We comment that this list contains at most $\sqrt{2n/k}$ integers; cf., [16].)

In the context of coding theory, our algorithms add a new dimension to the family of codes that are efficiently correctable. The known examples of asymptotically good error-correcting codes with efficient algorithms can be classified in one of two categories:

1. Algebraic codes: These are codes defined using the properties of low-degree polynomials over finite fields and include a wide variety of codes such as Reed-Solomon codes, BCH codes, Alternant codes and algebraic-geometry codes. Such codes admit efficient error-correction algorithms; in fact all the algorithms (for unique-decoding) are similar in spirit and can be unified quite nicely [31, 22, 10].

2. Combinatorial codes: A second class of codes with efficient decoding algorithms evolve from combinatorial concepts such as expanders, super-concentrators etc. Examples of this family include the codes of Sipser and Spielman [36], and Spielman [37]. In both cases, the description of the code is captured by a graph; and the existence of a decoding algorithm is then related to combinatorial properties of the graph.

Our work provides the first example of a number theoretic code that is efficiently correctable. To the best of our knowledge - this is the only example which does not fall into one of the two classes above.

Our algorithms are obtained by abstracting from known paradigms for correcting algebraic codes: The first of our algorithms abstracts from a large collection of (unique) error-correcting algorithms for algebraic codes [32, 4, 30, 42]. In fact, an elegant unification of these results (see [31, 22, 10] or the full version of this paper [15]) provides the inspiration for our algorithm. The second algorithm described above abstracts from the recent works on "list-decoding" algorithms [3, 38, 35, 18]. We stress however, that the translation of the above mentioned algorithms to our case is not immediate. In particular, the usual "interpolation" methods, that come in very handy in the algebraic case are not applicable here. In fact our code is not even linear in the usual sense and so linear algebra is not applicable in our case. Thus for solving analogies of "simple" problems in the algebraic case, we employ integer programming algorithms (in fixed dimensions) [25] for the Unique Decoding task, and the approximate basis reduction algorithm (in varying dimension) [24] for the List Decoding task. Our final algorithms achieve decoding capabilities comparable to those in algebraic cases and in particular, if $p_n = p_1^{O(1)}$ we can decode uniquely from a constant fraction of errors. We also get a list-decoding algorithm to recover from $n - o(n)$ errors, provided $k = o(n)$.

Permanent of random matrices One motivation for studying the Chinese remainder representation of integers was to study the "random self-reducibility" property of the permanent [26].

The standard presentation of this property *fixes* a prime $p > n+1$, and consists of a randomized reduction of computing the permanent modulo p of a given $n \times n$ matrix to computing the permanent modulo p over uniformly distributed $n \times n$ matrices. Thus we are taking a two parameter problem (such as Quadratic Non-Residuosity and DLP) and the process of self-reduction fixes one parameter (here, the prime p) and randomizes over the second (here, the matrix). This is analogous to the results of [17, 6] but not to the recent result of Ajtai [1]. Thus, unlike Ajtai's result, the above only relates the average and worst case complexities of computing the permanent modulo p for any fixed p. What we want is a relation between the average and worst case complexities, when average-case complexity refers to all parts of the input.

> Consider, for example, the product distribution on pairs (M, p), parameterized by size n, where p is a uniformly distributed n-bit prime and M is a uniformly distributed n-by-n matrix with $2n$-bit entries.

A naive analysis of the complexity of the permanent on such instances would work as follows. Suppose we have a heuristic to compute the permanent on instances from the above distribution. Then, given any pair (M, p), pick at random many primes $p_1, \ldots, p_t$, and then compute the permanent of M modulo p_i for every i. In each case use the random-self-reducibility of the permanent modulo p_i to reduce the computation of the permanent of M modulo p_i to $n+1$ "random" (but not independent) instances of the permanent modulo p_i. If the heuristic does not make errors very often (say has error probability less than $\frac{1}{3(n+1)t}$) then with high probability (resp., probability at least $2/3$) all calls to the heuristic get answered correctly. Thus if t is large enough (e.g., $t = O(n)$ will do), then (applying the Chinese Remainder Theorem) we obtain the value of the permanent of M (over the integers), and can now reduce this modulo p to get the desired output.

However the reduction as described above is not very tolerant of errors. This problem has been addressed before in the case of one of the two parameters, namely in the choice of the matrix: The results of [13, 14, 38] imply that if for

[2] In fact, the first task can be performed in nearly linear time (Theorem 16).

any prime p, the heuristic computes (M, p) on even a tiny but non-negligible fraction of the instances correctly then the permanent can be computed correctly on worst case instances of matrices, but over the same fixed prime p.

Our result complements the above, by allowing a similar treatment of the second parameter as well. Thus by combining the two results, we get the following natural statement (see Theorem 14):

> If there exists a heuristic that computes the permanent of a random pair (M, p), from the above distribution, with non-negligible probability (over the choice of (M, p)), then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$.

In independent related work, Cai et al. [9], provide an alternate formulation of the average-case hardness of the permanent, which is also hard on all parts of the input. They consider the hardness of computing the permanent directly over the integers. They show that if a BPP algorithm computes the permanent (over the integers) of a random $n \times n$ matrix with its entries chosen uniformly from among n-bit integers with non-negligible property then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$. In fact their techniques also extend to providing an alternate proof of Theorem 14 that does not use the decoding algorithm for the Chinese Remainder code.

Organization of this paper: In Section 2 we define the Chinese Remainder Code. In Sections 3 and 4 we give decoding algorithms for the Chinese Remainder Code, for small and large error, respectively. Section 5 gives the application to the permanent, and in Section 6 we describe an improved (nearly linear time) decoding algorithm for small error, and give an application of the Chinese Remainder Code to secret sharing.

2 The Chinese Remainder Code

Notation: For positive integers M, N, Let $\mathbb{Z}_M$ denote the set $\{0, \ldots, M-1\}$, and let $[N]_M$ denote the remainder of N when divided by M. Note that $[N]_M \in \mathbb{Z}_M$.

Definition 1 (Chinese Remainder Code) *Let $p_1 < \cdots < p_n$ be relatively prime integers, and $k < n$ an integer. The* Chinese Remainder Code *with basis $p_1, \ldots, p_n$ and rate k is defined for message space $\mathbb{Z}_K$, where $K \stackrel{\text{def}}{=} \prod_{i=1}^{k} p_i$. The encoding of a message $m \in \mathbb{Z}_K$, denoted $E_{p_1,\ldots,p_n}(m)$, is the n-tuple $\langle [m]_{p_1}, \ldots, [m]_{p_n} \rangle$.*

Thus the Chinese Remainder Code does not have a "fixed alphabet" (the alphabet depends on the coordinate position) and it is not linear in the usual sense (as the natural arithmetic here is done modulo p_i for the i'th coordinate). Distance of a code can however be defined as usual; i.e., the distance between two "words" of block length n is the number of coordinates on which they differ; and the distance of a code is the minimum distance between any pair of distinct codewords. The distance properties of this code are very similar to those of Reed-Solomon and BCH codes; and follow immediately from the Chinese Remainder Theorem:

Theorem 2 (Chinese Remainder Theorem — CRT) *If $q_1, \ldots, q_\ell$ are relatively prime positive integers and $r_1, \ldots, r_\ell$ are integers such that $r_i \in \mathbb{Z}_{q_i}$, then there exists a unique integer $r \in \mathbb{Z}_{\prod_{i=1}^{\ell} q_i}$ such that $[r]_{q_i} = r_i$. Furthermore, $r = \left[\sum_{i=1}^{\ell} c_i \cdot Q_i \cdot r_i\right]_Q$, where $Q = \prod_{j=1}^{\ell} q_j$, $Q_i = Q/q_i$, and c_i is the multiplicative inverse modulo q_i of Q_i.*

Corollary 3 *For any n relatively prime integers $p_1, \ldots, p_n$ and any integer $k < n$, the Chinese Remainder Code with basis $p_1, \ldots, p_n$ and rate k has distance $n - k + 1$. That is, for any two messages m_1, m_2, the code words $E_{p_1,\ldots,p_n}(m_1)$ and $E_{p_1,\ldots,p_n}(m_2)$ disagree on at least $n - k + 1$ coordinates.*

Thus if $p_1, \ldots, p_n$ are all $(1+o(1)) \cdot \log n$-bit primes, then the information rate and the distance of the Chinese Remainder Code are comparable with those of the Reed-Solomon code or the BCH code. For our purposes, it is more useful to consider a variant of the notions of block length, rate and distance as defined below.

Definition 4 (amplitude) *For a Chinese Remainder Code with basis $p_1, \ldots, p_n$ and rate k, the* amplitude of the encoding *is defined to be $N = \prod_{i=1}^{n} p_i$; the* amplitude of the message space *is defined to be $K = \prod_{i=1}^{k} p_i$. For vectors $\vec{v} = \langle v_1, \ldots, v_n \rangle$ and $\vec{w} = \langle w_1, \ldots, w_n \rangle \in \mathbb{Z}^n$ with $v_i, w_i \in \mathbb{Z}_{p_i}$, the* amplitude of the distance *between $\vec{v}$ and $\vec{w}$ is defined to be $\prod_{i: v_i \neq w_i} p_i$. The* amplitude of agreement *between $\vec{v}$ and $\vec{w}$ is defined to be $\prod_{i: v_i = w_i} p_i$. Notice that the product of the amplitudes of agreement and distance equals the amplitude of the encoding.*

It is easy to see that if the distance between $\vec{v}$ and $\vec{w}$ is d, and the amplitude of the distance between $\vec{v}$ and $\vec{w}$ is D; then $d \log p_1 \leq \log D \leq d \log p_n$. In case of traditional codes that are defined over fixed alphabets, i.e., $p_1 = p_2 = \cdots = p_n$, d is directly proportional to $\log D$ and hence there is no need to consider the latter separately. In our case, the latter parameter provides a more refined look at the performance of the algorithms. From the Chinese Remainder Theorem it follows immediately that the amplitude of distance between any two codewords is larger than N/K.

Our goal is to solve the following error-correction problems (for as large an error parameter as possible).

The Error-correction/List decoding Problem
Given: (1) *n relatively prime integers $p_1 < \cdots < p_n$ and rate parameter k specifying a Chinese Remainder Code*; (2) *n integers $r_1, \ldots, r_n$, with $r_i \in \mathbb{Z}_{p_i}$ and an error-parameter e.*
Task: *Find (all) message(s) $x \in \mathbb{Z}_K$, where $K = \prod_{i=1}^{k} p_i$, s.t. $[x]_{p_i} \neq r_i$ for at most e values of i.*

It follows from the distance of the Chinese Remainder Code that the answer is unique if $e < \frac{n-k}{2}$. In this case the problem corresponds to the traditional error-correction problem for error-correcting codes. If e is larger, then there may be more than one solution. We will expect the algorithm to return a list of all codewords x with at most e errors.

3 The Decoding Algorithm for Small Error

The first algorithm we present is a simple algorithm to recover from a small number of errors. The algorithm recovers from error of amplitude at most $\sqrt{N/K}$. Translating to classical measures this yields an error-correcting algorithm for $e \leq (n-k)\frac{\log p_1}{\log p_1 + \log p_n}$ (and in particular, if $p_n = p_1^{O(1)}$, then the algorithm can handle a constant fraction of errors).

The algorithm is described below formally. The inspiration for the algorithm comes from a general paradigm for decoding of many algebraic codes (see [31, 22, 10] or the full version of this paper [15]). Given a received word $\langle r_1, \ldots, r_n \rangle$ that is close to the encoding of (a unique) message m, the algorithm Unique-Decode tries to find two integers y and z such that $y \cdot m = z$. To this end it first reconstructs the integer $r \in \mathbb{Z}_N$ that corresponds to the received word $\langle r_1, \ldots, r_n \rangle$ (i.e., $[r]_{p_i} = r_i$, for every i). It then searches for integers y and z such that $y \cdot r \equiv z \pmod{N}$ (where $N = \prod_{i=1}^{n} p_i$), and both y and z are of bounded sizes. In the analysis of the algorithm we show that the equality (modulo N) between $r \cdot y$ and z together with the restrictions on the sizes of y and z implies that $y \cdot m$ is equal to z (over the integers). Furthermore, (as we show in the full version of this paper [15]), y has the following *error-detection* property: For every index i such that $r_i \neq [m]_{p_i}$, it holds that $[y]_{p_i} = 0$, and moreover, the message m can be reconstructed from the remaining r_i's Though we do not use this property explicitly in the algorithm described below (as well as in its analysis), it can be used to obtain a variant of the algorithm, (described in [15]), which is more clearly related to the general decoding paradigm.

Unique-Decode($p_1, \ldots, p_n, k, r_1, \ldots, r_n$).

Set $K = \prod_{i=1}^{k} p_i$, $N = \prod_{i=1}^{n} p_i$, and let E be an integer to be determined later.

Let $r \in \mathbb{Z}_N$ be s.t. $r_i = [r]_{p_i}$ (as defined by CRT).

1. Find integers y, z s.t.

$$\left.\begin{array}{rcl} 1 & \leq\ y\ \leq & E \\ 0 & \leq\ z\ < & N/E \\ & y \cdot r \equiv z \pmod{N} & \end{array}\right\} \qquad (1)$$

2. Output z/y if it is an integer.

The above algorithm can be implemented in polynomial time in the bit sizes of $p_1, \ldots, p_n$. Step 2 is straightforward. The main realization is that Step 1 can be computed using an algorithm for integer programming in fixed number of variables, due to [25]. To see how to formulate our problem in this way, we let the final equality be expressed as $y \cdot r = z + x \cdot N$. Our task thus reduces to computing y and x s.t $0 < y \leq E$ and $0 \leq y \cdot r - x \cdot N < N/E$. In Section 6.1 we show how this task can actually be performed in nearly linear time (using the "continued fractions method").

We now analyze the performance of this algorithm. We first describe it in terms of the amplitude of the distance between the message m and the received word r.

Lemma 5 *If r is such that for some $m \in \mathbb{Z}_K$ the amplitude of the distance between $\langle r_1, \ldots, r_n \rangle$ and $\langle [m]_{p_1}, \ldots, [m]_{p_n} \rangle$ is at most E, and $E < \sqrt{N/(K-1)}$, then* Unique-Decode($p_1, \ldots, p_n, k, r_1, \ldots, r_n$) *returns m.*

We prove the lemma using the following two claims.

Claim 5.1 *Under the premises of Lemma 5 there exist y, z satisfying Eq.* (1).

Claim 5.2 *Under the premises of Lemma 5, for any pair (y, z) satisfying Eq.* (1) *it holds that $y \cdot m = z$.*

We prove the two claim momentarily, and first show how Lemma 5 follows from the claims.

Proof of Lemma 5: By Claim 5.1, Step 1 of the algorithm always returns a pair (y, z) satisfying Eq. (1). By Claim 5.2, any pair (y, z) that may be the outcome of Step 1 satisfies $y \cdot m = z$. Thus $z/y = m$ is an integer and the output of the algorithm is m. ▮

We now prove Claims 5.1 and 5.2.

Proof of Claim 5.1: Let $y = \prod_{\{i | r_i \neq [m]_{p_i}\}} p_i$ (so that y equals the amplitude of the distance between $\langle r_1, \ldots, r_n \rangle$ and $\langle [m]_{p_1}, \ldots, [m]_{p_n} \rangle$), and $z = y \cdot m$. Then notice that $y \neq 0$, and $y \leq E$, and so the first item of Eq. (1) holds. Since $m \leq K-1$, we have $z = m \cdot y \leq (K-1) \cdot E$. Using $E < N/((K-1)E)$ (so that $(K-1) \cdot E < N/E$), and since $z \geq 0$, the second item of Eq. (1) also holds. Finally, by CRT, the condition $y \cdot r \equiv z \pmod{N}$ holds since the condition holds modulo every p_i: For any fixed $i \in \{1, \ldots, n\}$, either $r_i = [m]_{p_i}$ or $[y]_{p_i} = 0$. In either case, we have $z = ym \equiv yr \pmod{p_i}$. ▮

Proof of Claim 5.2: For every i s.t. $[m]_{p_i} = r_i$, we have

$$y \cdot m \ \equiv\ y \cdot [m]_{p_i} \ \equiv\ y \cdot r_i \ \equiv y\ \cdot r \ \equiv\ z \pmod{p_i}\,.$$

Thus, by CRT, $y \cdot m \equiv z \pmod{T}$ where $T = \prod_{\{i\ |\ [m]_{p_i} = r_i\}} p_i \geq N/E$ is the amplitude of the agreement between $\langle r_1, \ldots, r_n \rangle$ and $\langle [m]_{p_1}, \ldots, [m]_{p_n} \rangle$. But $z < N/E$ and $m \cdot y \leq (K-1)E < N/E$. Thus $z = m \cdot y$. ▮

As an immediate consequence of Lemma 5, and the observation relating amplitudes of distance to classical distance, we get the following theorem.

Theorem 6 Unique-Decode($p_1, \ldots, p_n, k, r_1, \ldots, r_n$) *solves the error-correction problem in polynomial time for any value of the error parameter $e \leq (n-k)\frac{\log p_1}{\log p_1 + \log p_n}$, with the setting $E = \prod_{i=n-e+1}^{n} p_i$.*

Proof: Using $N = \prod_{i=1}^{n} p_i$, $K = \prod_{i=1}^{k} p_i$ and $E = \prod_{i=n-e+1}^{n} p_i$, Lemma 5 can be applied if $E^2 \le N/K$ (as $N/K < N/(K-1)$). Namely, it suffices that $(\prod_{i=n-e+1}^{n} p_i)^2 \le \prod_{i=k+1}^{n} p_i$, which is equivalent to $\prod_{i=n-e+1}^{n} p_i \le \prod_{i=k+1}^{n-e} p_i$. In turn this condition holds if $p_n^e \le p_1^{n-k-e}$. The theorem follows by taking logarithms of both sides. ▮

4 Decoding for Large Error

In this section we will describe an algorithm that recovers from possibly many more errors than described in the previous section. In particular, if we fix $k = \epsilon n$ and let $n \to \infty$, the fraction of errors that can be corrected goes to $1 - \sqrt{2\epsilon \frac{\log p_n}{\log p_1}}$. As $\epsilon \to 0$, this quantity approaches 1. This algorithm is inspired by the recent progress in list-decoding algorithms [3, 38, 35, 18]. Our algorithm and analysis follow the same paradigm, though each step is different.

The algorithm List-Decode can be viewed as a generalization of Unique-Decode. In both algorithms, given the received word $\langle r_1, \ldots, r_n \rangle$, the algorithm first finds, using CRT, an integer $r \in \mathbb{Z}_N$ corresponding to the received word (i.e., $[r]_{p_i} = r$ for every i). In Unique-Decode the algorithm then attempts to find integers y and z (restricted in size), such that $y \cdot r \equiv z \pmod{N}$, and outputs z/y. In other words, the algorithm searches for integers y, z satisfying $y \cdot r - z \equiv 0 \pmod{N}$, and outputs the (unique) root of the (degree-1) polynomial $y \cdot x - z$. In List-Decode, the algorithm instead searches for a *sequence* of integers $c_0, \ldots, c_\ell$ (of certain bounded sizes), such that $\sum_i c_i r^i \equiv \pmod{N}$ and outputs *all* roots of the polynomial $\sum_i c_i x^i$. As we show subsequently, the increase in the degree of the polynomial that the algorithm searches for (together with the particular restrictions on the sizes of its coefficients) allows us to decode for much larger error.

List-Decode($p_1, \ldots, p_n, k, r_1, \ldots, r_n$).

Set $N = \prod_{i=1}^{n} p_i$; $K = \prod_{i=1}^{k} p_i$; and $F = 2^{\frac{\ell+2}{2}} \cdot \sqrt{\ell+2} \cdot N^{\frac{1}{\ell+1}} \cdot K^{\frac{\ell+1}{2}}$, with ℓ to be determined shortly.

Let $r \in \mathbb{Z}_N$ s.t. $[r]_{p_i} = r_i$ for every i (as defined by CRT).

1. Find integers $c_0, \ldots, c_\ell$ satisfying

$$\left.\begin{array}{ll} \forall 0 \le i \le l & |c_i| \le \frac{F}{K^i} \\ \text{s.t.} & \sum_{i=0}^{\ell} c_i r^i = 0 (\text{mod } N) \\ & \langle c_0, \ldots, c_\ell \rangle \ne \vec{0} \end{array}\right\} \qquad (2)$$

2. Output all roots of the integer polynomial $C(x) = \sum_{i=0}^{\ell} c_i x^i$.

The running time of Step 2 above is bounded by a polynomial in $n, \ell, \log N$ and $\log F$ (one can use LLL's algorithm for factoring polynomials over the integers if required, though faster algorithms exist for this simpler task of "root-finding"). We need to show how to implement Step 1. Mainly the idea is to set up a lattice whose short vectors correspond to small values of the coefficients c_i's. We show first that very small vectors of this form exist; and then use the basis reduction algorithm of LLL to find short (but not shortest) vectors in this lattice; and this will suffice for Step 1.

Lemma 7 (Algorithm for Step 1.) *c_i's as required in Step 1 of* List-Decode *exist and can be found in polynomial time.*

Proof: We set up an $\ell+2$-dimensional integer lattice using basis vectors $v_0, \ldots, v_\ell$ and w described next. Let M be a very large integer (to be determined later as a function of N and ℓ). For $j \in \{0, \ldots, \ell+1\}$, the jth coordinate of the vector v_i, denoted $(v_i)_j$ is given by:

$$(v_i)_j = \begin{cases} K^i & \text{if } j = i \\ M \cdot r^i & \text{if } j = \ell+1 \\ 0 & \text{otherwise.} \end{cases}$$

The vector w is zero everywhere except in the last coordinate where $(w)_{\ell+1} = M \cdot N$.

A generic vector in this lattice is of the form $u = \sum_{i=0}^{\ell} c_i v_i + dw$, for integers $c_0, \ldots, c_\ell$ and d. Explicitly the jth coordinate of u is given by:

$$(u)_j = \begin{cases} c_j K^j & 0 \le j \le \ell \\ M \cdot (\sum_{i=0}^{\ell} c_i r^i + dN) & \text{if } j = \ell+1. \end{cases}$$

We are interested in showing that this lattice contains "short" vectors whose last coordinate equals 0, and every other coordinate has absolute value at most F (thus satisfying Eq. (2)). Furthermore, we would like to show that such vectors can be found efficiently. To his end, we first prove the following technical lemma.

Lemma 8 *For integers r, N if $B_0, \ldots, B_\ell$ are positive integers such that $\prod_{i=0}^{\ell} B_i > N$, then there exist integers $c_0, \ldots, c_\ell$, such that $|c_i| < B_i$, $\langle c_0, \ldots, c_\ell \rangle \ne \vec{0}$ and $\sum_{i=0}^{\ell} c_i r^i \equiv 0 \pmod{N}$.*

Proof: Consider the function $f : \mathbb{Z}_{B_0} \times \cdots \times \mathbb{Z}_{B_\ell} \to \mathbb{Z}_N$ given by $f(c_0, \ldots, c_\ell) = [\sum_{i=0}^{\ell} c_i r^i]_N$. Since the domain has larger cardinality than the range, there exist different $\langle d_0, \ldots, d_\ell \rangle$ and $\langle e_0, \ldots, e_\ell \rangle$ s.t. $f(d_0, \ldots, d_\ell) = f(e_0, \ldots, e_\ell)$. Setting $c_i = d_i - e_i$, we get $|c_i| < B_i$, $\sum_i c_i r^i = 0$, and $\langle c_0, \ldots, c_\ell \rangle \ne \vec{0}$ as required. ▮

Using Lemma 8 with $B_i = N^{\frac{1}{\ell+1}} \cdot K^{\frac{\ell+1}{2} - i}$, we observe that the lattice defined above has a (short) non-zero vector (where the c_i's are as guaranteed by the lemma and $d = -\sum_{i=0}^{\ell} c_i r^i / N$) with the last coordinate identically 0, and each other coordinate has absolute value at most $B_i \cdot K^i = N^{\frac{1}{\ell+1}} \cdot K^{\frac{\ell+1}{2}}$. Thus, the L_2-norm of this vector is at most $\sqrt{\ell+2} \cdot N^{\frac{1}{\ell+1}} \cdot K^{\frac{\ell+1}{2}}$. By using the "approximate shortest vector" algorithm of [24], we find, in polynomial time, a vector of L_2-norm at most $F = 2^{\frac{\ell+2}{2}} \cdot \sqrt{\ell+2} \cdot N^{\frac{1}{\ell+1}} \cdot K^{\frac{\ell+1}{2}}$. For sufficiently large M (any $M > F$ will do), all "short" vectors (i.e., with L_2-norm at most F) have a last coordinate identical to 0, and thus yield a sequence of c_i's satisfying $\sum_i c_i r^i \equiv 0 \pmod{N}$ and $|c_i \cdot K^i| \leq F$. This sequence is as required in Step 1. $\blacksquare$

Now we move on to Step 2 of List-Decode. We argue next that any solution to the list-decoding problem is a root of the polynomial whose coefficients are given by *any* solution to Step 1. Instead of performing the analysis in terms of the amount of error in the received word, we do so in terms of the amount of agreement with some message.

Lemma 9 *If r is such that for some $m \in \mathbb{Z}_K$ the amplitude of the agreement between $\langle r_1, \ldots, r_n \rangle$ and $\langle [m]_{p_1}, \ldots, [m]_{p_n} \rangle$ is greater than $2(\ell+1)F$, and $c_0, \ldots, c_\ell$ are integers satisfying Eq. (2), then $\sum_{j=0}^{\ell} c_j m^j = 0$ (i.e., m is a root of the polynomial $C(x)$).*

Proof: We first observe that since the c_j's are small, $\sum_j c_j m^j$ is small in absolute value:

$$\begin{aligned} \left| \sum_{j=0}^{\ell} c_j m^j \right| &\leq (\ell+1) \cdot \max_j \{|c_j m^j|\} \\ &\leq (\ell+1) \cdot \max_j \{|c_j K^j|\} \\ &\leq (\ell+1) \cdot F. \end{aligned}$$

Now we observe that for i such that $[m]_{p_i} = r_i$ it holds that

$$\sum_{j=0}^{\ell} c_j m^j \equiv \sum_{j=0}^{\ell} c_j [m]_{p_i}^j \equiv \sum_{j=0}^{\ell} c_j r_i^j \equiv \sum_{j=0}^{\ell} c_j r^j \equiv 0 \pmod{p_i}.$$

Define $P = \prod_{\{i | r_i = [m]_{p_i}\}} p_i$. By CRT, $\sum_{j=0}^{\ell} c_j m^j \equiv 0 \pmod{P}$. Since the sum $\sum_{j=0}^{\ell} c_j m^j$ has absolute value at most $(\ell+1)F$, the hypothesis $P > 2 \cdot (\ell+1)F$ implies that the sum is identically zero as required. $\blacksquare$

As an immediate consequence of the last two lemmas, we get a proof of the correctness of List-Decode. The following proposition describes the performance in terms of amplitude (for any choice of ℓ).

Proposition 10 *For any choice of the parameter ℓ,* List-Decode$(p_1, \ldots, p_n, k, r_1, \ldots, r_n)$ *produces a list of up to ℓ integers which includes all messages $m \in \mathbb{Z}_K$ such that the amplitude of agreement between $\langle [m]_{p_1}, \ldots, [m]_{p_n} \rangle$ and $\vec{r}$ is at least $2(\ell+2)^{3/2} 2^{\frac{\ell+2}{2}} N^{\frac{1}{\ell+1}} K^{\frac{\ell+1}{2}}$.*

Proof: By Lemma 7, c_i's satisfying Eq. (2) exist and are found in Step 1. By Lemma 9, any m as in the lemma is a root of the polynomial $\sum_j c_j x^j$, and thus is included in the output. $\blacksquare$

The following theorem is obtained by optimizing the choice of the parameter ℓ in the above proposition. See the full version of this paper [15] for its proof.

Theorem 11 List-Decode$(p_1, \ldots, p_n, k, r_1, \ldots, r_n)$ *with parameter $\ell = \left\lceil \sqrt{\frac{2n \log p_n}{k \log p_1}} - 1 \right\rceil$ solves the error-correction problem in polynomial time, for $e < n - \sqrt{2(k+3)n \frac{\log p_n}{\log p_1}} - \frac{k+6}{2}$.*

Remark: If $k/n = \epsilon$, then the above theorem indicates that approximately $1 - \sqrt{2 \cdot \left(\frac{\log p_n}{\log p_1}\right) \cdot \epsilon} - \epsilon/2$ fraction of errors can be corrected. In particular this fraction approaches 1 as $\epsilon \longrightarrow 0$.

5 The Permanent of Random Matrices

In this section we show that computing the permanent of a random matrix modulo a random prime is very hard. The distribution of matrices and primes we consider is the following:

$\mathcal{D}$ is an ensemble of distributions $\{\mathcal{D}_s\}$ where $\mathcal{D}_s$ consists of pairs (T, p) where T is an $s \times s$ matrix whose entries are chosen uniformly and independently from $\mathbb{Z}_{2^{2s}}$, and p is a prime chosen uniformly from $\mathbb{Z}_{2^s}$.

The distributional problem we consider is: Given a randomly chosen pair (T, p) from $\mathcal{D}_s$, compute the permanent of T modulo p. We show that no polynomial time algorithm is likely to have inverse polynomial probability of solving this distributional problem.

Lemma 12 ([2] following [28]; cf., [8]) *Suppose there exists a probabilistic polynomial time algorithm A' and a polynomial $r : \mathbb{Z} \rightarrow \mathbb{Z}$ such that on input M, an $s \times s$ matrix of $2s$-bit integer elements, $A'(M)$ outputs a list of $r(s)$ integers such that the permanent of M is included in this list (with probability at least, say, $\frac{1}{2}$ over the internal coin tosses of A'). Then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$.*

We complement this lemma with an algorithm that utilizes a subroutine for computing the permanent on random instances, and uses it to compute a list of values of the permanent on worst-case instances.

Lemma 13 *Suppose there exists a polynomial time algorithm A and a function $\epsilon : \mathbb{Z} \rightarrow [0, 1]$ such that for every positive integer s,*

$$\Pr_{(T,p) \in \mathcal{D}_s} [A(T, p) = [\mathrm{perm}(T)]_p] \geq \epsilon(s).$$

Then there exists a randomized $\mathrm{poly}(s/\epsilon(s))$-time algorithm A' that on input an $s \times s$ matrix M with entries from $\mathbb{Z}_{2^{2s}}$, outputs a list of at most $O(1/\epsilon(s)^4)$ integers, which includes the permanent of M with high probability.

Proof: Assume, w.l.o.g, that when given a pair (T, p), algorithm A first reduces each entry of T modulo p. Our algorithm for reconstructing the permanent of any s-by-s matrix, M, is given below:

Algorithm Perm(M).

```
• Parameters $n = \mathrm{poly}(s/\epsilon(s))$, $n' = O(s/\epsilon(s)^2)$
• Uniformly select $n$ random primes $p_1, \ldots, p_n$ in the interval $[2^{s/2}, 2^s]$.
• For $i = 1$ to $n$ do    /* try to obtain $[\mathrm{perm}(M)]_{p_i}$ */
  Subroutine Mod-Perm$(M, p_i)$.
    – Uniformly select an $s \times s$ random matrix $R$ with entries from $\mathbb{Z}_{p_i}$.
    – For $j = 1$ to $n'$ do    /* try to obtain $[\mathrm{perm}(M + jR)]_{p_i}$ */
          Let $v_j = A(M + j \cdot R, p_i)$;
    – Reconstruct a list of all degree $s$ univariate polynomials $\{f_1, \ldots, f_{\ell'}\}$ that satisfy $f_h(j) = v_j$ for at least an $\epsilon(s)/16$ fraction of the $v_j$'s.
    – Uniformly select a random $h \in \{1, \ldots, \ell'\}$ and set $r_i = f_h(0)$.
          /* with probability poly$(\epsilon(s))$ (taken over the choice of $p_i$ and the internal coins of Mod-Perm), we will have $r_i = [\mathrm{perm}(M)]_{p_i}$ */
• Reconstruct a list of all integers $x \le s!2^{s^2}$ such that $[x]_{p_i} = r_i$ for at least $t = O(\epsilon(s)^4) \cdot n$ of the $i$'s, and output this list. Namely, apply List-Decode with parameters $p_1, \ldots, p_n$, $k = 6s$ (as $K = s!2^{s^2} < 2^{3s^2}$ and $\forall i,\ p_i \ge 2^{s/2}$), and $r_1, \ldots, r_n$.
```

The polynomial reconstruction step may be performed using the algorithm of [38], which requires $n' \ge 2s \cdot (\epsilon(s)/16)^{-2}$. (To recover polynomials of degree s from a list of values at n' places, the algorithm requires the agreement t' to satisfy $t' > \sqrt{2sn'}$.) The reconstruction of integers satisfying the Chinese Remainder Property uses Theorem 11 and works when $n = \Omega(s/\epsilon(s)^8)$. (Here to recover all sequences with agreement t out of n places, the algorithm requires $t = \Omega(\sqrt{kn}) = \Omega(\sqrt{sn})$.)

Let P_s denote the set of primes in the interval $[2^{s/2}, 2^s]$. Let $\mathcal{D}'_s$ be the distribution over pairs (T', p') where p' is chosen uniformly in P_s (rather than among the primes in $\mathbb{Z}_{2^s}$, as defined by $\mathcal{D}_s$), and then T' is chosen uniformly from the set of $s \times s$ matrices with entries from $\mathbb{Z}_{p'}$ (rather than by reducing modulo p' a matrix with entries chosen independently and uniformly in $\mathbb{Z}_{2^{2s}}$). We notice that the statistical difference between the two distributions is at most $O\left(\frac{2^{s/2}/(s/2)}{2^s/s}\right) + s^2 \cdot \frac{2^s}{2^{2s}}$, which is negligible (where the first term comes from the probability that in D_s a prime smaller than $2^{s/2}$ is selected, and the second from uneven wrap-around in the reduction modulo a prime). In particular this implies that

$$\Pr_{(T',p') \in \mathcal{D}'_s} \left[A(T', p') = [\mathrm{perm}(T')]_{p'} \right] \ge \frac{\epsilon(s)}{2} .$$

Say that a prime p' (from P_s) is *good* if

$$\Pr_{T' \in \mathbb{Z}_{p'}^{s \times s}} \left[A(T', p') = [\mathrm{perm}(T')]_{p'}] \right] \ge \frac{\epsilon(s)}{4} .$$

A simple counting argument shows that at least $\epsilon(s)/4$ fraction of the primes in P_s are good.

For any fixed good prime p', and for any $j \in \{1, \ldots, n'\}$, we thus have that

$$\Pr_{R \in \mathbb{Z}_{p'}^{s \times s}} \left[A(M + jR, p') = [\mathrm{perm}(M + jR)]_{p'} \right] \ge \frac{\epsilon(s)}{4}$$

(recall that we assume that when given a pair (T, p), algorithm A first reduces each entry of T modulo p). Say that a matrix R is *compatible* with p' if

$$\Pr \left[\left| \left\{ j : \; A(M + jR, p') = [\mathrm{perm}(M + jR)]_{p'} \right\} \right| > \frac{\epsilon(s)}{16} n' \right]$$
$$> \frac{\epsilon(s)}{16} ,$$

(where the probability here is taken only over the coin flips of A). It is not hard to verify that the probability that a random R is compatible with p' is at least $\epsilon(s)/8$. It follows that for any good p',

$$\Pr \left[\text{Mod-Perm}(M, p') \; = \; [\mathrm{perm}(M)]_{p'} \right] \ge \frac{\epsilon(s)}{8} \cdot \frac{\epsilon(s)}{16} \cdot \frac{1}{\ell'}$$

where the first term $(\epsilon(s)/8)$ is the probability that R is compatible with p'; the second $(\epsilon(s)/16)$ is the probability that A returns the correct output for at least $\epsilon(s)/16$ fraction of the j's (so that the polynomial reconstruction can work), conditioned on R being compatible; and the third term $(1/\ell')$ is the probability of selecting the correct index h. As $\ell' \le 2 \cdot (\epsilon(s)/16)^{-1}$ (cf., [38]), the above probability is $\Omega(\epsilon(s)^3)$.

Recall that the probability that each p_i (uniformly selected in P_s) is good is at least $\epsilon(s)/4$. Hence, the probability, taken over the choice of p_i and the random coin flips of Mod-Perm that Mod-Perm$(M, p_i) = [\mathrm{perm}(M)]_{p_i}$, is $\Omega(\epsilon(s)^4)$. Finally, since the success events of the various i's are independent, by applying a Chernoff bound, we get that with high probability, the number of p_i's for which $r_i = [\mathrm{perm}(M)]_{p_i}$ is at least $\Omega(\epsilon(s)^4) \cdot n$. In this case List-Decode will succeed in reconstructing a list that includes perm(M). ▌

By combining Lemma 12 and Lemma 13 we get

Theorem 14 *Suppose there exists a polynomial time algorithm A and a positive polynomial function $q : \mathbb{Z} \to \mathbb{Z}$ such that for every positive s,*

$$\Pr_{(T,p) \in \mathcal{D}_s} [A(T, p) = [\mathrm{perm}(T)]_p] \ge \frac{1}{q(s)}$$

Then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$.

Remark 15 *A quick examination of the proof shows that the theorem continues to hold if the distribution $\mathcal{D}_s$ is altered so that the primes are chosen uniformly from $\mathbb{Z}_{f(s)}$, and the entries of the matrix are chosen uniformly from $\mathbb{Z}_{f^2(s)}$, where f is any super-polynomial function. For $f(s) > 2^{4s^2}$, there exists a simpler argument which does not use the CRT decoding algorithm (see [15]).*

6 Improvements and Applications

6.1 Nearly linear time algorithms for the CRT Code

In this section we review some well-known results which yield fast algorithms for tasks associated with the CRT code. In particular, there exist nearly linear time algorithms for encoding and for decoding with $(n-k)\frac{\log p_1}{\log p_1+\log p_n}$ errors. The following theorem summarizes these results.

Theorem 16 *For relatively prime integers $p_1, \ldots, p_n$, let $b = \sum_{i=1}^{n}(1+\lfloor \log_2 p_i \rfloor)$. Then the following tasks can be performed in time $O(b \log^c b)$ for some constant c:*

1. Encoding: *Given $k \leq n$ and $m < \prod_{i=1}^{k} p_i$, compute $([m]_{p_1}, \ldots, [m]_{p_n})$.*
2. Decoding without errors: *Given $k \leq n$ and $(r_1, \ldots, r_n)$, $r_i \in \mathbb{Z}_{p_i}$, compute $m < \prod_{i=1}^{k} p_i$ such that $[m]_{p_i} = r_i$ for every $i \in \{1, \ldots, n\}$, in case such m exists.*
3. Decoding with errors: *Given $k \leq n$ and $(r_1, \ldots, r_n)$, $r_i \in \mathbb{Z}_{p_i}$, compute $m < \prod_{i=1}^{k} p_i$ such that $[m]_{p_i} \neq r_i$ for at most $(n-k)\frac{\log p_1}{\log p_1+\log p_n}$ values of $i \in \{1, \ldots, n\}$, in case such m exists.*

Parts (1) and (2) of Theorem 16 follow immediately from the fact that the Chinese remainder representation can be computed and inverted in nearly linear time (cf. [7, Theorems 4.5.3 and 4.5.8]). These results in turn follow from nearly linear time algorithms due to Schonhage and Strassen [33] for multiplying and dividing two integers. (These algorithms are combined with a binary-tree structure in which the residues modulo individual p_i's are associated with the leaves and the residue modulo $\prod_{i=1}^{n} p_i$ is associated with the root.) So we just need to prove Part (3); that is, we show that the algorithm Unique-Decode can be implemented in nearly linear time.

A nearly linear time implementation of Step 2 (i.e., computing z/y) follows from the nearly linear time algorithm for integer division of Schonhage and Strassen [33] and from the fact that both z and y are at most b-bits long. Thus, we focus on Step 1. In this step we wish to compute y and z subject to the Eqn. (1). Equivalently, given N, E and r, we wish to find integers x, y such that

$$1 \leq y \leq E, \ x \geq 0, \text{ and } 0 \leq y \cdot r - x \cdot N < N/E. \qquad (3)$$

In turn the above can be rewritten as:

$$1 \leq y \leq E, \ x \geq 0, \text{ and } \frac{x}{y} < \frac{r}{N} < \frac{x}{y} + \frac{1}{y \cdot E}. \qquad (4)$$

Setting $\alpha = \frac{r}{N}$, the above problem is that of approximating a rational α from *below* by another rational number $\frac{x}{y}$ with denominator no larger than E. (In particular the approximation should be within an additive factor of less than $\frac{1}{y \cdot E}$.) This will be done using the "continued fractions method", and specifically algorithms due to Knuth [21].

We briefly introduce some notation and summarize known results regarding continued fractions. We follow the description in Lovasz [27, pages 9–12]. Given a positive real α, consider the sequence $a_0, a_1, \ldots$, defined as follows: $\alpha_0 = \alpha$ and $a_0 = \lfloor \alpha_0 \rfloor$. For $i = 0, 1, \ldots$, if $\alpha_i = a_i$ then the sequence terminates, else we define $\alpha_{i+1} = \frac{1}{\alpha_i - a_i}$ and $a_{i+1} = \lfloor \alpha_{i+1} \rfloor$. Let $\mathrm{CF}(\alpha)$ denote the sequence $(a_0, a_1, \ldots)$. It is well known that this sequence has finite length if and only if α is rational. Furthermore, for every finite sequence $(a_0, \ldots, a_l)$ of integers $a_i \geq 1$, there exists a unique rational number α such that $\mathrm{CF}(\alpha) = (a_0, a_1, \ldots, a_l)$. We use $\mathrm{CF}^{-1}(a_0, \ldots, a_l)$ to denote this α. Turning to algorithmics, we recall that the function CF can be computed and inverted in nearly linear time [21]: That is, if α is given as the ratio of two n-bit integers, then $\mathrm{CF}(\alpha)$ can be computed in time $O(n \log^{O(1)} n)$. Conversely, given a sequence of integers $(a_0, \ldots, a_l)$ with bit lengths summing to n, a pair of integers p, q such that $p/q = \mathrm{CF}^{-1}(a_0, \ldots, a_l)$ can also be computed in time $O(n \log^{O(1)} n)$.

The properties of the continued fraction representation that are of interest to us are the following. For rational α, let $(a_0, a_1, a_2, \ldots, a_l) = \mathrm{CF}(\alpha)$. For $0 \leq i \leq l$, let $\frac{g_i}{h_i} = \mathrm{CF}^{-1}(a_0, \ldots, a_i)$. Then the following facts hold (see [27, pages 9–12] for proofs):

(CF1) The h_i's are monotonically increasing.

(CF2) The g_i's and h_i's satisfy $g_{i+1} \cdot h_i - g_i \cdot h_{i+1} = (-1)^i = \mathrm{sgn}\left(\alpha - \frac{g_i}{h_i}\right)$. In particular it follows that $|\alpha - \frac{g_i}{h_i}| \leq \frac{1}{h_i \cdot h_{i+1}}$ for every i.

(CF3) For any integer E, let k be the largest index such that $h_k \leq E$. Let $j = \left\lfloor \frac{E-h_{k-1}}{h_k} \right\rfloor$, $\beta \stackrel{\text{def}}{=} \frac{g_k}{h_k}$, and $\beta' \stackrel{\text{def}}{=} \frac{g_{k-1}+j \cdot g_k}{h_{k-1}+j \cdot h_k}\}$. Then,

1. the number α lies between $\beta_1 = \min\{\beta, \beta'\}$ and $\beta_2 = \max\{\beta, \beta'\}$ (each being a rational with denominator at most E); and
2. every rational lying strictly between β_1 and β_2 has a denominator strictly larger than E.

It follows that β_1 is the largest rational less than α with denominator at most E.

We show that β_1 as in (CF3) necessarily satisfy Eqn. (4), in case some y', x' satisfying this equation do exist. Furthermore, we show that in case $\alpha = r/N$ and r, N are given, β_1 can be found in almost linear time. This yields the algorithm we were looking for. We comment that β_1 is the *best rational lower bound on $\alpha = \frac{r}{N}$ with denominator bounded by E*. That is, y, x satisfy $\frac{x}{y} \leq \alpha$, $y \leq E$ and every rational between $\frac{x}{y}$ and α has denominator greater than E.

Proposition 17 *Let E be an integer and α be a number so that*

$$1 \leq y \leq E, \ x \geq 0, \text{ and } \frac{x}{y} \leq \alpha < \frac{x}{y} + \frac{1}{y \cdot E} \qquad (5)$$

has a solution. Then the rational β_1 as in (CF3) is a solution. Furthermore, given b-bit integers r, N, E, and setting $\alpha = r/N$, it is possible to compute the rational β_1 in time $O(b \log^c b)$.

By the premise of Part 3 of Theorem 16 (concerning the existence of m as desired), and Claim 5.1, we know that there exists a solution to Eqn. (5). Part 3 of Theorem 16 follows using the same arguments as in the proof of Theorem 6. The proof of Proposition 17 is given in the full version of this paper [15].

6.2 Secret Sharing based on CRT

We present a new scheme for secret sharing. The scheme is based on the CRT-code, analogously to the way Shamir's secret-sharing scheme [34] is based on Reed-Solomon codes.

Recall that in Shamir's scheme, for parameters $t < n$ and $q > n$, one is given a secret $s \in \mathrm{GF}(q)$ and shares it among n parties by uniformly selecting a degree t polynomial, p, over $\mathrm{GF}(q)$ with free term s, and handing $p(i)$ to the i^{th} party. Clearly, any $t+1$ parties can recover the secret (by interpolation), whereas no set of t parties obtains any information about the secret. In abstract terms, Shamir's scheme consists of selecting a random codeword among those of a certain "label", and giving each party a block of bits in the codeword. We can do the same in case of the CRT code, and our secret sharing scheme follows.

Construction 18 (The CRT secret-sharing scheme):

parameters: *$t < n$ and primes $p_0 < p_2 < p_1 < \cdots < p_n$.*

sharing: *To share a secret $a_0 \overset{\mathrm{def}}{=} s \in \mathrm{GF}(p_0)$ one does the following*

1. *uniformly selects $a_1 \in \mathrm{GF}(p_1)$,..., $a_t \in \mathrm{GF}(p_t)$;*
2. *finds $x \in \mathbb{Z}_{\prod_{i=0}^{t} p_i}$ so that $x \equiv a_i \pmod{p_i}$, for $i = 0, 1, ..., t$;*
3. *sets the i^{th} share to be $x \bmod p_i$, for $i = 1, ..., n$.*

reconstructing: *Given any $t+1$ shares, $s_{i_1}, ..., s_{i_{t+1}}$, corresponding to parties $i_1, ..., i_{t+1}$, one reconstructs the secret as follows*

1. *finds $y \in \mathbb{Z}_{\prod_{j=1}^{t+1} p_{i_j}}$ so that $y \equiv s_{i_j} \pmod{p_{i_j}}$, for $j = 1, ..., t, t+1$.*
2. *recover the secret to be $(y \bmod p_0)$.*

We first show that the reconstruction indeed works. Consider x and y as computed in Step (2) of the Sharing procedure and Step (1) of the Reconstruction procedure, respectively. Clearly, $y \equiv x \pmod{p_{i_j}}$, for $j = 1, ..., t, t+1$. Viewing x and y as non-negative integers, we have $x < \prod_{i=0}^{t} p_i < \prod_{j=1}^{t+1} p_{i_j}$ and $x = y$. Thus, $y \equiv x \pmod{p_i}$ for every $i = 0, 1, ..., t$, and $y \equiv s \pmod{p_0}$ follows. On the other hand, the first t shares yield no information about the secret. As for other sets of upto t shares, here some information about the secret is leaked, but we can upper bound its amount.

Proposition 19 *Let $s, s' \in \mathrm{GF}(p_0)$, let $r_1, ..., r_t$ be chosen as in Step (1) of the sharing Sharing, and let $X(s)$ (resp., $X(s')$) denote the value computed in Step (2). Then, for every set $I \subset [n]$ of indices, the statistical difference between $(X(s) \bmod \prod_{i \in I} p_i)$ and $(X(s') \bmod \prod_{i \in I} p_i)$ is at most*

$$2 \cdot \frac{\prod_{i \in I} p_i}{\prod_{i=1}^{t} p_i}$$

Thus, in general, security is provided only for $|I| \leq t-1$ (rather than for $|I| \leq t$ as in case of Shamir's shceme). An advised choice of parameters is to have p_i's be of the same magnitude and large enough so that $1/p_i$ is negiligible in the security parameter.

Proof: Let us further generalize the claim and consider, for two integers K, M each relatively prime to p, the randomized process $R : \mathbb{Z}_p \mapsto \mathbb{Z}_{pK}$ which maps each $s \in \mathbb{Z}_p$ to a uniformly selected member of $\{r \in \mathbb{Z}_{pK} : r \equiv s \pmod{p}\}$. We are interested in the statistical difference between $(R(s) \bmod M)$ and $(R(s') \bmod M)$, for the worst possible pair $s, s' \in \mathbb{Z}_p$. (In our case, $p \overset{\mathrm{def}}{=} p_0$, $K \overset{\mathrm{def}}{=} \prod_{i=1}^{t} p_i$, $R(s) \overset{\mathrm{def}}{=} X(s)$, and $M \equiv \prod_{i \in I} p_i$.)

Clearly $R(s) \equiv s + r \cdot p$, where r is uniformly chosen in $\mathbb{Z}_K$ (and same for $R(s')$). So,

$$[R(s)]_M \equiv [s]_M + [r]_M \cdot [p]_M \pmod{M}$$

The point is that $[r]_M$ is the only randomness in the r.h.s., and that multiplying by $[p]_M$ is a permutation over $\mathbb{Z}_M$ (since p is relatively prime to M). Thus, if $[r]_M$ is uniformly distributed over $\mathbb{Z}_M$ then $[R(s)]_M$ and $[R(s')]_M$ are identically distributed. In general, the statistical difference between the latter is bounded by twice the statistical difference of $[r]_M$ (where r is uniformly chosen in $\mathbb{Z}_K$) from the uniform distribution on $\mathbb{Z}_M$. In case M divides K the statistical difference is zero, and otherwise it is $(K \bmod M)/K$ which is bounded above by M/K. The claim follows. ▮

Acknowledgments

We would like to thank Venkatesan Guruswami for bringing the work of [10] to our attention. We would like to thank Valentine Kabanets for pointing out an error in the earlier version of this paper.

References

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). *STOC* 1996.

[2] A. Amir, R. Beigel, and W. Gasarch. Cheatable, P-terse, and P-superterse sets, manuscript, Dec. 1989.

[3] S. Ar, R. Lipton, R. Rubinfeld and M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):488-511, 1999.

[4] E. R. Berlekamp. *Algebraic Coding Theory.* McGraw Hill, New York, 1968.

[5] E. R. Berlekamp. Bounded Distance +1 Soft-Decision Reed-Solomon Decoding. *IEEE Transactions on Information Theory*, 42(3):704-720, 1996.

[6] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Computing*, Vol. 13, pages 850–864, 1984.

[7] A. Borodin and I. Munro. *The Computational Complexity of Algebraic and Numeric Problems.* American Elsevier Publishing Company, New York, 1975.

[8] J. Cai and L. A. Hemachandra. A note on enumerative counting. *Information Processing Letters*, 38(4):215-219, 31 May 1991.

[9] J. Cai, A. Pavan, and D. Sivakumar. On the Hardness of Permanent. *STAACS*, 1999.

[10] I. M. Duursma. *Decoding codes from curves and cyclic codes.* Ph.D. Thesis, Eindhoven, 1993.

[11] P. Elias. List decoding for noisy channels. Technical Report 335, Research Lab. of Electronics, MIT, 1957.

[12] P. Elias. Error-correcting codes for list decoding. *IEEE Trans. on Information Theory*, 37:5-12. 1991.

[13] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. *STOC* pages 32-42, New Orleans, Louisiana, 6-8 May 1991.

[14] P. Gemmell and M. Sudan. Highly resilient correctors for multivariate polynomials. *Information Processing Letters*, 43(4):169-174, 1992.

[15] O. Goldreich, D. Ron and M. Sudan. Chinese Remaindering with Errors. Available from *ECCC*, 1998.

[16] O. Goldreich, R. Rubinfeld and M. Sudan. Learning polynomials with queries: The highly noisy case. *36th FOCS*, pages 294–303, 1995. Revised version available from *ECCC*, 1998.

[17] S. Goldwasser and S. Micali. Probabilistic Encryption. *JCSS*, Vol. 28, No. 2, pages 270–299, 1984.

[18] V. Guruswami and M. Sudan. Improved decoding for Reed-Solomon and algebraic-geometric codes. *FOCS* 1998.

[19] E. Kaltofen. Polynomial factorization 1987–1991. *LATIN '92*, I. Simon (Ed.) Springer LNCS, v. 583:294-313, 1992.

[20] R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. Technical report TR-31-81, Aiken Computation Laboratory, Harvard University, 1981.

[21] D.E. Knuth. The analysis of algorithms. *Actes du Congres International des Mathematiciens*, Tome 3, 269-274, 1970.

[22] R. Kotter. A unified description of an error locating procedure for linear codes. *Proceedings of Algebraic and Combinatorial Coding Theory*, Voneshta Voda, Bulgaria, 1992.

[23] E. Kushilevtitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[24] A. K. Lenstra, H. W. Lenstra and L. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[25] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8: 538–548, 1983.

[26] R. J. Lipton. New directions in testing. *Distributed Computing and Cryptography*, J. Feigenbaum and M. Merritt (ed.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematics Society, 2:191–202, 1991.

[27] L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity.* SIAM Publications, 1986.

[28] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *JACM*, Vol. 39, No. 4, pages 859–868, 1992.

[29] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, Amsterdam, 1981.

[30] J. L. Massey. Shift register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15:122–127, 1969.

[31] R. Pellikaan. On decoding linear codes by error correcting pairs. *Eindhoven University of Technology*, preprint, 1988.

[32] W. W. Peterson. Encoding and error-correction procedures for Bose-Chaudhuri codes. IRE Transactions on Information Theory, IT-60:459-470, 1960.

[33] A. Schonhage and V. Strassen. Schnelle multiplikation grosser zahlen. *Computing*, 7:281-292, 1971.

[34] A. Shamir. How to Share a Secret. *CACM*, Vol. 22, Nov. 1979, pages 612–613.

[35] M. A. Shokrollahi and H. Wasserman. Decoding algebraic-geometric codes beyond the error-correction bound. *STOC*, 1998.

[36] M. Sipser and D. A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

[37] D. A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1732, 1996.

[38] M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180-193, 1997.

[39] J. H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, New York, 1982.

[40] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189-201, April 1979.

[41] A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. *STOC*, 1997.

[42] L. Welch and E. R. Berlekamp. Error correction of algebraic block codes. *US Patent* Number 4,633,470, issued December 1986.