Chinese Remaindering with Errors*

Oded Goldreich† Dana Ron‡ Madhu Sudan§

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 gives a number-theoretic construction of an "error-correcting code" that has been considered often in the past (see [41, 19, 35]). In this code a "message" (integer) $m < \prod_{i=1}^{k} p_i$ is encoded by the list of its residues modulu $p_1, \ldots, p_n$. By the Chinese Remainder Theorem, if a code-word is corrupted in $e < \frac{n-k}{2}$ coordinates, then there exists a unique integer m whose corresponding code-word differs from the corrupted word in at most e places. Furthermore, Mandelbaum [25, 26] shows how m can be recovered efficiently given the corrupted word, provided that the p_i's are very close to one another. To deal with arbitrary p_i's, we present a variant of his algorithm that runs in almost linear-time and recovers from $e < \frac{\log p_1}{\log p_1 + \log p_n} \cdot (n-k)$ errors. Our main contribution is an efficient decoding algorithm for the case in which the error e may be larger than $\frac{n-k}{2}$. 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 $t = \Omega(\sqrt{kn \frac{\log p_n}{\log p_1}})$. For $n \gg k$ (and $p_n \leq p_1^{O(1)}$), the fraction of error corrected by the algorithm is almost twice that corrected by the previous work. More significantly, the algorithm recovers the message even when the amount of agreement between the "recieved word" and the "codeword" is much smaller than the number of errors.

Keywords: List-decoding, Redundant Residue Number System (RRNS) Codes, Lattice reduction.

*An extended abstract of this paper appeared in *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 225-234, Atlanta, Georgia, 1-4 May 1999.

†Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel. oded@wisdom.weizmann.ac.il. Work done in part while visiting MIT, and partially supported by DARPA grant DABT63-96-C-0018.

‡Department of Electrical Engineering – Systems, Tel Aviv University, Ramat Aviv, Israel. danar@eng.tau.ac.il. Work done in part while visiting MIT, supported by an ONR Science Scholar Fellowship of the Bunting Institute.

§Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, USA. madhu@mit.edu. Research supported in part by a Sloan Foundation Fellowship, an MIT-NEC Research Initiation Grant and NSF Career Award CCR-9875511.

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 representation of m. Specifically, m can be recovered given any k of the n remainders.

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 [16]. This representation was used to show the strength of probabilistic communication over deterministic communication protocols (cf. [20, Exercise 3.6]). The representation allows for easy arithmetic — addition, subtraction, and multiplication — on large integers and was even proposed by Watson and Hastings [41] 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 [39].

In the context of error correction, this representation of integers yields a natural *error-correcting code* that has been studied extensively in the literature (see [35, 19] and the references therein). The central property that allows for the correction of error is that for any two integers $m, m' < \prod_{i=1}^k p_i$, the sequences $\{(m \bmod p_1), \ldots, (m \bmod p_n)\}$ and $\{(m' \bmod p_1), \ldots, (m' \bmod p_n)\}$ (which we view as *encodings* of m and m', respectively), differ in at least $n - k + 1$ coordinates. This difference (or distance) has the following implication. Consider a sequence of integers $\langle r_1, \ldots, r_n \rangle$ that are obtained by taking residues of an integer $m < \prod_{i=1}^k p_i$ modulo $p_1 < \cdots < p_n$, where e of the residues are *erroneous*. In other words, $\langle r_1, \ldots, r_n \rangle$ differs from the code-word representing m in e coordinates. Then the Chinese Remainder Theorem indicates that for $e < \frac{n-k}{2}$ there exists a *unique* code-word that differs from $\langle r_1, \ldots, r_n \rangle$ in at most e coordinates. Thus, for such bounded error, m is uniquely specified by the vector $\langle r_1, \ldots, r_n \rangle$.

The first algorithmic question that arises is whether, under the above conditions, it is possible to recover m in polynomial time (i.e., in time polynomial in n and $\log p_n$). This question was answered affirmatively by Mandelbaum [25, 26]. Mandelbaum gives an efficient algorithm for the above *Unique Decoding* problem when $e < \frac{n-k}{2}$, provided that the p_i's are very close. Specifically, his algorithm runs in time[2]

$$\mathrm{poly}\left(\sum_{i=1}^{n} \log p_i\right) \cdot \frac{\prod_{i=(n+k)/2}^{n} p_i}{\prod_{i=k}^{(n+k)/2} p_i}$$

Note that the ratio may be exponential in $n - k$, even if $p_n = p_1^{1+o(1)}$ (e.g., if $p_i \approx \frac{n+1}{n} \cdot p_{i-1}$ for all i's and $n = 3k$ then the ratio is approximately $\exp(n/9)$).

We present and analyze a variant of Mandelbaum's algorithm. This variant can be implemented in almost-linear time, and solves the unique decoding problem when $e \le \frac{\log p_1}{\log p_1 + \log p_n} \cdot (n - k)$. Note that for $p_n = p_1^{1+o(1)}$, we have $\frac{\log p_1}{\log p_1 + \log p_n} \approx \frac{1}{2+o(1)}$, and for $p_i \approx \frac{n+1}{n} \cdot p_{i-1}$ we have $\frac{\log p_1}{\log p_1 + \log p_n} \approx \frac{1}{2} - \frac{1}{4 \log p_1}$.

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

[2] Mandelbaum's paper does not provide an explicit analysis of the running-time. The poly$(\cdot)$ factor is actually almost linear.

The main focus of our work is the study of the decoding problem for larger error. In this case there is no unique code-word that differs in at most e coordinates from the received word $\langle r_1, \ldots, r_n \rangle$. However, as long as $e \leq n - \sqrt{nk}$, 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 [12]. In this paper we present an efficient algorithm for recovering this list in polynomial time. More precisely, the algorithm solves the following task:

> 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., [12].)

Coding theory context. The better 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 [29, 18, 8].

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 [34], and Spielman [36]. 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.

By extending the work of Mandelbaum [25, 26] (see also [3]), our work contributes to broadening the study of efficiently decodable error-correcting codes, to include a number-theoretic code. 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 unique-decoding algorithm abstracts from a large collection of (unique) error-correcting algorithms for algebraic codes [30, 4, 27, 42, 5]. In fact, an elegant unification of these results (see [29, 18, 8] or the technical report version of this paper [11]) provides the inspiration for our algorithm. The list-decoding algorithm abstracts from the recent works on list-decoding algorithms for algebraic codes [1, 37, 33, 13]. 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 the Chinese Remainder 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 the continued-fraction method for the Unique Decoding task, and the approximate basis reduction algorithm [21] for the List Decoding task. Our final algorithms achieve decoding capabilities comparable to those in algebraic cases. In particular, the list-decoding algorithm recovers from $n - o(n)$ errors, provided $p_n = p_1^{O(1)}$ and $k = o(n)$.

Applications. As mentioned earlier the Chinese Remainder code is used naturally in many places. We suspect the decoding algorithm could be of use in some of these applications. At this point we are aware of some scenarios in which the decoding algorithm(s) for the Chinese Remainder code are of use. In Section 5, we describe the applicability of the Chinese Remainder code and the decoding algorithm in "secret sharing" as an alternate to Shamir's scheme [32]. (The latter used the properties of the Reed-Solomon code). In the technical report version of this paper [11], an application to the "average-case complexity of computing the permanent of random matrices" is also given. One more case where the list-decoding algorithm for the CRT code has been utilized is in a result of Håstad and Näslund [14] who utilize the list-decoding algorithm for the CRT code (as part of a more involved algorithm) to show that all the individual bits of $ax + b \bmod p$ are "secure" given a, b, p and $f(x)$ for any "one-way function" f.

Organization of this paper. In Section 2 we define the Chinese Remainder Code. In Section 3 we describe the unique decoding algorithm for the Chinese Remainder Code (in case of small error), and in 4 we give the list-decoding algorithm (in case of large error). In Section 5 we describe the 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 is very similar to Mandelbaum's algorithm [26], but is presented and analyzed slightly differently. In particular, we simplify the description of the algorithm by invoking a powerful algorithm for integer programming in small dimensions, due to Lenstra [22]. 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 [29, 18, 8] or the technical report version of this paper [11]). 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 technical report version of this paper [11]), 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 [11]), 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.
$$\left.\begin{array}{ccccc} 1 & \leq & y & \leq & E \\ 0 & \leq & z & < & N/E \\ \multicolumn{5}{l}{y \cdot r \equiv z \pmod{N}} \end{array}\right\} \qquad (1)$$
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 Lenstra [22]. 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 the technical report version of this paper [11] we also show how the method of continued fractions leads to a nearly linear time algorithm for this task.

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 $\mathsf{Unique\text{-}Decode}(p_1, \ldots, p_n, k, r_1, \ldots, r_n)$ *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 $E^2 \leq N/K$ (as $N/K < N/(K-1)$). Namely, it suffices that $(\prod_{i=n-e+1}^{n} p_i)^2 \leq \prod_{i=k+1}^{n} p_i$, which is equivalent to $\prod_{i=n-e+1}^{n} p_i \leq \prod_{i=k+1}^{n-e} p_i$. In turn this condition holds if $p_n^e \leq p_1^{n-k-e}$. 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 [1, 37, 33, 13]. 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 $\sum_i c_i r^i \equiv 0 \pmod N$ 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

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

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 easily bounded by a polynomial in $n, \ell, \log N$ and $\log F$. For example, one can use the algorithm for factoring polynomials over the integers due to Lenstra, Lenstra and Lovasz [21] (LLL). Faster algorithms are also known for the simpler task of "root-finding". (In particular, one may use classical results on the bound of real roots to isolate an interval which includes all the real roots of the given polynomial. Then one may perform binary search to approximate the real roots, using the method of Sturm sequences to determine the number of real roots in any interval. For details on the bounds of roots and the method of Sturm sequences, c.f. Mishra [28, Sections 8.3 and 8.4]. Finally since we are only interested in integer roots, an additive approximation of say 1/3 suffices to determine them. Based on these ideas it is possible to obtain an algorithm with running time $O(l^3(\log F)^3)$ to list all integer roots of a degree l polynomial with coefficients in the range $[-F, +F]$. Faster implementations may be possible, but we are unaware of any published work.)

The more involved task is Step 1 and we will show how to implement it in polynomial time. 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 [21] 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 $|c_i| < B_i$, $\langle c_0, \ldots, c_\ell \rangle \neq \vec{0}$ 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 $\langle c_0, \ldots, c_\ell \rangle \neq \vec{0}$ 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 [21], 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 $|c_i \cdot K^i| \le F$. This sequence is as required in Step 1. ▮

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. ▌

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. ▌

The following theorem is obtained by optimizing the choice of the parameter ℓ in the above proposition.

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, if $\frac{\log p_n}{\log p_1}$ is fixed (say, $p_n \leq p_1^2$) and $\epsilon \to 0$, then this fraction approaches 1.

Proof: Suppose we want to find all codewords which agree with $\langle r_1,\ldots,r_n\rangle$ on t coordinates. Setting $f_1 \stackrel{\text{def}}{=} (\ell+2)^{3/2}$ and $f_2 \stackrel{\text{def}}{=} 2^{\frac{\ell+2}{2}}$, and applying Proposition 10, it suffices to show that

$$\prod_{i=1}^{t} p_i \geq f_1 \cdot f_2 \cdot \left(\prod_{i=1}^{n} p_i\right)^{\frac{1}{\ell+1}} \cdot \left(\prod_{i=1}^{k} p_i\right)^{\frac{\ell+1}{2}}$$

Setting $t = t_1 + t_2 + t_3$, we will find t_1, t_2, t_3 s.t.

$$p_1^{t_1} \geq f_1 \tag{3}$$
$$p_1^{t_2} \geq f_2 \tag{4}$$
$$\text{and } \prod_{i=1}^{t_3} p_i \geq \left(\prod_{i=1}^{n} p_i\right)^{\frac{1}{\ell+1}} \cdot \left(\prod_{i=1}^{k} p_i\right)^{\frac{\ell+1}{2}} \tag{5}$$

We start with an analysis of the last inequality. For this we need

$$\left(\prod_{i=1}^{t_3} p_i\right)^{1-\frac{1}{\ell+1}} \geq \left(\prod_{i=t_3+1}^{n} p_i\right)^{\frac{1}{\ell+1}} \cdot \left(\prod_{i=1}^{k} p_i\right)^{\frac{\ell+1}{2}}$$

Let $q = (\prod_{i=1}^{k} p_i)^{1/k}$. Then $q \geq p_1$ and $(\prod_{i=1}^{t_3} p_i) \geq q^{t_3}$, provided $t_3 \geq k$. Thus it suffices to show

$$q^{t_3\ell/(\ell+1)} \geq p_n^{(n-t_3)/(\ell+1)} \cdot q^{k\cdot\frac{\ell+1}{2}} \tag{6}$$

Fact: Eq. (6) holds if $t_3 \geq \frac{k(\ell+1)}{2} + \frac{n \log p_n}{(\ell+1)\log p_1}$ and $\ell \geq 1$.

Proof: By the hypothesis, $t_3 \geq k$ and so

$$\begin{aligned}
\left(t_3 - \frac{k(\ell+1)}{2}\right)\log p_1 &\geq \frac{n}{\ell+1}\log p_n \\
\Rightarrow \left(\frac{\ell}{\ell+1}t_3 - \frac{k(\ell+1)}{2}\right)\log q &\geq \frac{n-t_3}{\ell+1}\log p_n \\
\Rightarrow \frac{\ell t_3}{\ell+1}\log q &\geq \frac{n-t_3}{\ell+1}\log p_n + \frac{k(\ell+1)}{2}\log q
\end{aligned}$$

and Eq. (6) follows. □

Setting $\ell + 1 = \left\lceil\sqrt{\frac{2n\log p_n}{k\log p_1}}\right\rceil$, shows that $t_3 = \sqrt{2kn\frac{\log p_n}{\log p_1}} + \frac{k}{2}$ suffices to achieve Eq. (5). This setting of ℓ also implies that to satisfy Eq. (4), which is equivalent to $t_2 \geq \frac{\ell+2}{2\log p_1}$, it suffices to set $t_2 = \sqrt{\frac{2n\log p_n}{k\log p_1}} + \frac{3}{2}$. Finally, to satisfy Eq. (3), which is equivalent to $t_1 \geq \frac{3\log(\ell+2)}{2\log p_1}$, it suffices to set $t_1 = \frac{\log(2n\log p_n/k\log p_1)}{\log p_1}$, which is smaller than $2\log t_2/\log p_1 \ll t_2$.

Thus we find that it suffices to have

$$\begin{aligned}
t &= 2\sqrt{\frac{2n\log p_n}{k\log p_1}} + 3 + \sqrt{2kn\frac{\log p_n}{\log p_1}} + \frac{k}{2} \\
&= \left(1 + \frac{2}{k}\right)\cdot\sqrt{2kn\frac{\log p_n}{\log p_1}} + \frac{k+6}{2} \\
&< \sqrt{1 + 3\cdot\frac{2}{k}}\cdot\sqrt{2kn\frac{\log p_n}{\log p_1}} + \frac{k+6}{2} \\
&= \sqrt{2(k+3)n\frac{\log p_n}{\log p_1}} + \frac{k+6}{2}
\end{aligned}$$

Setting $e < n - t$ yields the theorem. ∎

Comparison with [1, 37] Our algorithm List-Decode is similar to those of [1, 37] in the basic steps. In their case also, they first find a polynomial "explaining" the corrupted word and then factor it to retrieve a list of messages. However the specifics are quite different: They look for a bivariate polynomial explanation; their criterion is to find a non-zero polynomial of low degree; they find it by solving a linear system; and then employ a bivariate factorization step. We look for a univariate polynomial explanation; our criterion is the size of the coefficients; we find it by (essentially) solving Diophantine systems; and finally employ univariate factorization. Similarly our analysis follows the same steps. The existence proof (Lemma 8) is similar to an analogous step in [37]; though our proof here appears to be more general than his proof. In particular, the pigeonhole argument could also be applied to his case achieving analogous results. Finally, Lemma 9 is also analogous in spirit to similar lemmas in [1, 37] - again our proofs are different since our criteria are different.

5 Secret Sharing based on CRT

The CRT code has been proposed as a alternate method for implementing secret-sharing [7, 2]. The scheme is based on the CRT-code, analogously to the way Shamir's secret-sharing scheme [32] is based on Reed-Solomon codes. The existence of decoding algorithms for the CRT code enhances the power of such a secret-sharing scheme. In this section, we formally analyze the secrecy preserved by such a scheme. First we recall the notion of a secret sharing scheme.

Loosely speaking, a secret sharing scheme is a method for a (honest) dealer to distribute its secret among n parties, giving each party a share of the secret, so that the following two conflicting goals are achieved

1. any large enough collection of parties can efficiently recover the secret; but
2. no small coalition of parties can obtain any information about the secret.

The collection of parties in item (1) is thought of as a collection of available honest parties, whereas the coalition in item (2) refers to a coalition of dishonest parties. Typically, a secret sharing scheme is defined in terms of two thresholds, t_1 and t_2 (s.t., $t_1 > t_2$), corresponding to the above two items. Below we relax item (2), introduce a parameter bounding the information leakage, and require that the information leaked about the secret to any subset of up-to t_2 parties is bounded by this parameter.[3] We mention that secret sharing is one of the most widely used tools in Cryptography.

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 12 (The CRT secret-sharing scheme):

[3] The actual formulation will be slightly different: We will bound the statistical difference between t_2 shares generated for any two possible secrets.

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

sharing: *To share a secret $a_0 \stackrel{\rm def}{=} s \in \mathrm{GF}(p_0)$ 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 $i^{\rm th}$ 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 13 *Let $s, s' \in \mathrm{GF}(p_0)$, let $r_1, ..., r_t$ be chosen as in Step (1) of the Sharing procedure, 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 $|I| \le t-1$ (rather than for $|I| \le t$ as in case of Shamir's scheme). 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 negligible 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, $p \stackrel{\rm def}{=} p_0$, $K \stackrel{\rm def}{=} \prod_{i=1}^{t} p_i$, $R(s) \stackrel{\rm def}{=} X(s)$, 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 [8] to our attention. We would like to thank Valentine Kabanets for pointing out an error in the earlier version of this paper. We would like to thank Michael Quisquater for the pointer to [2], and Amin Shokrollahi for the pointers to [25, 26, 3]. We would like to thank an anonymous reviewer for numerous pointers including [7, 19, 25, 35, 41].

References

[1] S. Ar, R. Lipton, R. Rubinfeld and M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):488-511, 1999. Preliminary version in *FOCS*, 1992.

[2] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29(2):208-210, 1983.

[3] F. Barsie and P. Maestrini. Error detection and correction by product codes in residue number systems. *IEEE Transactions on Information Theory*, 24(5):640-643, 1978.

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

[7] D. E. R. Denning. *Cryptography and Data Security.* Addison-Wesley, Reading, Massachusetts, 1983.

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

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

[10] P. Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37(1):5-12, 1991.

[11] O. Goldreich, D. Ron and M. Sudan. Chinese Remaindering with Errors. TR98-062, available from *ECCC*, at `http://www.eccc.uni-trier.de/eccc/`, 1998.

[12] O. Goldreich, R. Rubinfeld and M. Sudan. Learning polynomials with queries: The highly noisy case. *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 294–303, Milwaukee, Wisconsin, 23-25 October 1995. Revised version available from *ECCC*, 1998.

[13] V. Guruswami and M. Sudan. Improved decoding for Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6): 1757-1767, September 1999. Preliminary version in *FOCS* 1998.

[14] J. Håstad and M. Näslund. The Security of all RSA and Discrete Log bits. *Manuscript*, February 1999.

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

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

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

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

[19] H. Krishna, B. Krishna, K.-Y. Lin, and J.-D. Sun. *Computational Number Theory and Digital Signal Processing: Fast Algorithms and Error Control Techniques.* CRC Press Inc., Boca Raton, Florida, 1994.

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

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

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

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

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

[25] D. M. Mandelbaum. On a class of arithmetic codes and a decoding algorithm. *IEEE Transactions on Information Theory*, 22(1):85-88, 1976.

[26] D. M. Mandelbaum. Further results on decoding arithmetic residue codes, *IEEE Transactions on Information Theory*, 24(5):643-644, 1978.

[27] J. L. Massey. Shift register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.

[28] B. Mishra. *Algorithmic Algebra.* Springer-Verlag, New York, 1993.

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

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

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

[32] A. Shamir. How to Share a Secret. *Communications of the ACM*, Vol. 22, Nov. 1979, pages 612–613.

[33] M. A. Shokrollahi and H. Wasserman. Decoding algebraic-geometric codes beyond the error-correction bound. *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 241-248, Dallas, Texas, 23-26 May, 1998.

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

[35] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor. *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing.* IEEE Press, New York, 1986.

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

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

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

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

[40] A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 92-109, El Paso, Texas, 4-6 May 1997.

[41] R. W. Watson and C. W. Hastings. Self-checked computation using residue arithmetic. *Proceedings of the IEEE*, vol. 54, pp. 1920-1931, December 1966. (Also available as [35, pp. 325-336].)

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