Chinese Remaindering with Errors

Oded Goldreich* Dana Ron† Madhu Sudan‡

October 7, 1998

Abstract

The Chinese Remainder Theorem states that a positive integer \( m \) is uniquely specified by its remainders 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 \( m \leq \prod_{i=1}^{k} p_i \) 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 \( (m \mod p_i) = r_i \) for at least \( t \) values of \( i \in \{1, \ldots, n\} \), provided \( t = \Omega(\sqrt{kn \log q_n / \log p_n}) \). We also give a simpler algorithm to decode from a smaller number of errors, i.e., when \( t > n - (n - k)\frac{\log q_n}{\log \log q_n} \). In such a case there is a unique integer which has such an agreement with the sequence of residues.

One consequence of our result is that it significantly strengthens 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 \#P = BPP. Previous results of this nature typically worked over a fixed prime moduli or assumed very small (though non-negligible) error probability (as opposed to small but non-negligible success probability).

* Department of Computer Science and Applied Math, Weizmann Institute of Sciences
† Department of Electrical Engineering — Systems, Tel Aviv University
‡ Laboratory for Computer Science, MIT
1 Introduction

The Chinese Remainder Theorem states that a positive integer $m$ is uniquely specified by its remainder modulo $k$ relatively prime integers $p_1, \ldots, p_n$, 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 very 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 $\{(m \mod p_1), \ldots, (m \mod p_n)\}$ and $\{(m' \mod p_1), \ldots, (m' \mod p_n)\}$ differ in at least $n - k$ 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 [17]. This representation was used to show the strength of probabilistic communication over deterministic communication protocols. 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. 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 permanent of $0/1$ matrices. In fact it is this application that motivates the main question studied here.

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 residues $\langle r_1, \ldots, r_n \rangle$ that are obtained from taking residues of an integer $m < \prod_{i=1}^{k} p_i$ modulo primes $p_1 < \cdots < p_n$, 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 $n$ 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 [13]. 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:

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 $(m \mod p_i) \neq r_i$ for at most $(n - k)\frac{\log p_n}{\log p_i}$ 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 $(m \mod p_i) = r_i$ for at least $\sqrt{2n(k + 2)\frac{\log p_n}{\log p_i}} + \frac{k+3}{2} + 2 \log n = \Theta(\sqrt{nk\frac{\log p_n}{\log p_i}})$ values of $i \in \{1, \ldots, n\}$. (Theorem 11.) (We comment that this list contains at most $\sqrt{2n/k}$ integers; cf., [13].)

\footnote{Unfortunately, it does not allow for easy inequality comparisons — which is presumably why it was not employed.}
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 [25, 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 [28], and Spielman [29]. 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 [26, 4, 24, 33]. In fact, an elegant unification of these results (see [25, 18, 8]) provides the inspiration for our algorithm. The second algorithm described above abstracts from the recent works on “list-decoding” algorithms [3, 30, 27, 15]. 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 even 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) [20] for the Unique Decoding task, and the approximate basis reduction algorithm (in varying dimension) [19] 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 primary motivation for studying the Chinese remainder representation of integers was to study the “random self-reducibility” property of the permanent [21].

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 [14, 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 \((p, M)\), 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 \((p, M)\), pick at random many primes \(p_1, \ldots, p_k\), 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 [11, 12, 30] imply that if for 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:

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 \(\text{P} \neq \text{P}^\# \Rightarrow \text{BPP}\).

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.

## 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 \([N]_M \in \mathbb{Z}_M\).

**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 \(K := \prod_{i=1}^k p_i\). The encoding of a message \(m \in \mathbb{Z}_K\), denoted \(E_{p_1, \ldots, p_n}(m)\), is the \(n\)-tuple \(([m]_{p_1}, \ldots, [m]_{p_n})\).

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. The distance properties of this code however are very similar to those of Reed-Solomon and BCH codes; and follow immediately from the Chinese Remainder Theorem.
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 $e \leq (n-k)\frac{\log p_i}{\log p_1 + \log p_n}$. 

### 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 $e \leq (n-k)\frac{\log p_i}{\log p_1 + \log p_n}$.
The algorithm is described below formally. The intuition behind the algorithm comes from a general paradigm for decoding of many algebraic codes (see [25, 18, 8]). Given a received word \( \langle r_1, \ldots, r_n \rangle \) that is close to the encoding of (a unique) message \( m \), we try and detect the indices for which \( r_i \neq [m]_{p_i} \). We then reconstruct the message, using CRT, from those \( r_i \)'s which we believe are correct. The above detection is done using an integer \( y \) which we show satisfies \([y]_{p_i} = 0 \) whenever \( r_i \neq [m]_{p_i} \). By restricting \( y \) to be relatively small we ensure that \([y]_{p_i} \) does not equal \( 0 \) for many \( i \) satisfying \( r_i = [m]_{p_i} \) (so that CRT can in fact be applied). However to find this \( y \), we need some way to (describe and) exploit the fact there exists some small \( m \) s.t., for every \( i \), \([y]_{p_i} = 0 \) or \([m]_{p_i} = r_i \); or equivalently \([y]_{p_i} \cdot [m]_{p_i} \equiv [y]_{p_i} \cdot r_i \pmod{p_i} \). The final condition suggest that we may attempt to find \( z = y \cdot m \) such that \([z]_{p_i} \equiv [y]_{p_i} \cdot r_i \pmod{p_i} \). While ideally we would like to specify further that \( z \) is a multiple of \( y \), we relax this and simply use the fact that \( z \) is also small (since both \( y \) and \( m \) are small). This leads to the following algorithm:

\[\text{Unique-Decode}(p_1, \ldots, p_n, k, r_1, \ldots, r_n)\]

- Set \( K = \prod_{i=1}^{k} p_i \), \( N = \prod_{i=1}^{n} p_i \), and \( F = (K-1)E \), with \( E \) to be determined later.
- Let \( r \in \mathbb{Z}_N \) be s.t. \( r_i = [r]_{p_i} \).

1. Find integers \( y, z \) s.t.

\[
\begin{align*}
1 & \leq y \leq E \\
0 & \leq z \leq F \\
y \cdot r & \equiv z \pmod{N}
\end{align*}
\]

(1)

2. Let \( I \) def \( \{ i : [y]_{p_i} = 0 \} \). For every \( i \in I \), set \( x_i = r_i \).

3. Find \( x \in \mathbb{Z}_K \) s.t. \([x]_{p_i} = x_i \) for every \( i \in I \) (if such an \( x \) exists) and output it.

The above algorithm can be implemented in polynomial time in the bit sizes of \( p_1, \ldots, p_n \). The main realization is that Step 1 can be computed using an algorithm for integer programming in fixed number of variables, due to [20]. To see how to formulate our problem in this way, we let the final equality be expressed as \( y \cdot r = z + j \cdot N \). Our task thus reduces to computing \( y \) and \( j \) s.t \( 0 < y \leq E \) and \( 0 \leq y \cdot r - j \cdot N < F \). Step 2 is straightforward, while Step 3 is just an application of the Chinese Remainder Theorem (i.e., find \( x \in \mathbb{Z}^\prod_{i \in I} p_i \) via CRT and check if it is \( \cdot \) smaller than \( K \).

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 \( r \).

**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 \( N > E^2 \cdot K \) then Unique-Decode\((p_1, \ldots, p_n, k, r_1, \ldots, r_n)\) returns \( m \).

**Proof:** We prove the lemma using a sequence of claims.

**Claim 5.1** For \( r \) as in the lemma, there exist \( y, z \) satisfying Eq. (1).
Proof: \( \text{Let } y = \prod_{i \in [p]} p_i; \text{ (so that } y \text{ equals the amplitude of the distance between } \langle r_1, \ldots, r_n \rangle \text{ and } \langle [m]_p, \ldots, [m]_p \rangle \text{), and } z = y \cdot m. \text{ Then notice that } y \neq 0, \text{ and } y \leq E. \text{ Since } m < K, \text{ we have } z = m \cdot y \leq (K - 1) \cdot E. \text{ Finally, by CRT, the condition } y \cdot r \equiv z \pmod{N} \text{ holds since the condition holds modulo every } p_i: \) for any fixed \( i \in \{1, \ldots, n\}, \) either \( r_i = [m]_p \) or \( \langle y \rangle_{p_i} = 0. \text{ In either case, we have } z = ym \equiv yr \pmod{p_i}. \)

Claim 5.2 \( \text{Let } r \text{ and } m \text{ be as stated in the lemma, and } N \geq (K - 1) \cdot E^2. \text{ For any pair } (y, z) \text{ satisfying Eq. (1) it holds that } y \cdot m = z. \)

Proof: \( \text{For every } i \text{ s.t. } [m]_p = r_i, \text{ we have } m \cdot y \equiv r_i \cdot y \equiv y \cdot r \equiv z \pmod{p_i}. \)

Thus, by CRT, \( y \cdot m \equiv z \pmod{T} \) where \( T = \prod_{i \in [p]} [m]_p = r_i \geq N/E \) is the amplitude of the agreement between \( \langle r_1, \ldots, r_n \rangle \) and \( \langle [m]_p, \ldots, [m]_p \rangle. \) But \( z \) and \( m \cdot y \) are both at most \((K - 1)E < N/E. \text{ Thus } z = m \cdot y. \)

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 \( y \cdot m = z. \) Since \( y \cdot r \equiv z \pmod{N}, \text{ it follows that for every } i \cdot y \cdot r \equiv y \cdot m \pmod{p_i}, \text{ and so for every } i \in I = \{j: y \neq 0 \pmod{p_j}\}, \text{ we have } r \equiv m \pmod{p_i}. \text{ Thus, } m \in \mathbb{Z}_K \text{ is a valid solution for the task in Step 3 (since } \forall i \in I, x_i = [r]_{p_i}. \)

It remains to show that \( m \) is the only possible solution. For this, let \( I = \{1, \ldots, n\} \setminus I \) (i.e., \( \forall i \in I, y \equiv 0 \pmod{p_i}. \)) By CRT, \( y \equiv 0 \pmod{\prod_{i \in I} p_i}, \text{ and since } y > 0 \text{ it follows that } y \geq \prod_{i \in I} p_i. \text{ Since } y \leq E, \text{ we have } \prod_{i \in I} p_i \geq N/E > K. \text{ Thus, again by CRT, the message } m \in \mathbb{Z}_K \text{ is the only solution to the system } \{x \equiv r \pmod{p_i}\}_{i \in I}. \)

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

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

Proof: \( \text{Using } N = \prod_{i=1}^{n} p_i, K = \prod_{i=k}^{n} p_i \text{ and } E \leq \prod_{i=n-\epsilon+1}^{n} p_i, \text{ Lemma 5 can be applied if } \prod_{i=1}^{n} p_i > (\prod_{i=1}^{k} p_i) \cdot (\prod_{i=n-\epsilon+1}^{n} p_i)^2, \text{ which is equivalent to } \prod_{i=k+1}^{n-\epsilon+1} p_i > \prod_{i=n-\epsilon+1}^{n} p_i. \text{ In turn this condition holds if } p_i^{n-\epsilon+k} > p_i^{n-\epsilon+1}. \text{ The theorem follows by taking logarithms of both sides. } \)

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 \( 1 - \frac{\log p_i}{2\epsilon \log p_1}. \) As \( \epsilon \to 0, \) this quantity approaches 1. This algorithm is inspired by the recent progress in list-decoding algorithms \([3, 30, 27, 15]\). Our algorithm and
analysis follow the same paradigm, though each step is different. A closer comparison is included at the end of this section.

Instead of describing the algorithm in terms of the amount of error, we will try to describe it in terms of the amount of agreement $t$ that it requires between the codeword and the received word. We use $T$ to denote the amplitude of agreement.

**Proof:** Consider the function

$$f: \mathbb{Z} \times \cdots \times \mathbb{Z} \rightarrow \mathbb{Z}$$

such that $f(c_0, \ldots, c_l) = \sum_{i=0}^{l} c_i r^i \bmod N$. Since the domain has larger cardinality than the range, there exist different $(d_0, \ldots, d_l)$ and $(e_0, \ldots, e_l)$ such that $f(d_0, \ldots, d_l) = f(e_0, \ldots, e_l)$. Setting $c_i = d_i - e_i$, we get $|c_i| < B_i$, $\sum_{i=0}^{l} c_i r^i = 0$, and $(c_0, \ldots, c_l) \neq 0$ as required.

**Lemma 7** For integers $r, N$ if $B_0, \ldots, B_l$ are positive integers such that $\prod_{i=0}^{l} B_i > N$, then there exist integers $c_0, \ldots, c_l$, such that $|c_i| < B_i$, $(c_0, \ldots, c_l) \neq 0$ and $\sum_{i=0}^{l} c_i r^i \equiv 0 \pmod{N}$.

**Proof:** Consider the function $f: \mathbb{Z} \times \cdots \times \mathbb{Z} \rightarrow \mathbb{Z}$ given by $f(c_0, \ldots, c_l) = \sum_{i=0}^{l} c_i r^i$. Since the domain has larger cardinality than the range, there exist different $(d_0, \ldots, d_l)$ and $(e_0, \ldots, e_l)$ such that $f(d_0, \ldots, d_l) = f(e_0, \ldots, e_l)$. Setting $c_i = d_i - e_i$, we get $|c_i| < B_i$, $\sum_{i=0}^{l} c_i r^i = 0$, and $(c_0, \ldots, c_l) \neq 0$ as required.

**Lemma 8** (Algorithm for Step 1.) $c_i$'s as required in Step 1 of List-Decoding exists 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 $\ell$). For $j \in \{0, \ldots, \ell+1\}$, the $j$th 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 \( a = \sum_{i=0}^{\ell} c_i \cdot v_i + d \cdot w \), for integers \( c_0, \ldots, c_\ell \) and \( d \). Explicitly the \( j \)th coordinate of \( a \) is given by:

\[
(a)_j = \begin{cases} 
  c_j \cdot K^j & 0 \leq j \leq \ell \\
  M \cdot (\sum_{i=0}^{\ell} c_i r^i + d N) & \text{if } j = \ell + 1.
\end{cases}
\]

Using Lemma 7 with \( B_i = N^{1/(\ell+1)} \cdot K^{(\ell+1)/2-i} \), we observe that this lattice 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 \( B_i \cdot K^i = N^{1/(\ell+1)} \cdot K^{(\ell+1)/2} \). Thus, the \( L_2 \)-norm of this vector is at most \( \sqrt{\ell + 2} \cdot N^{1/(\ell+1)} \cdot K^{(\ell+1)/2} \). By using the “approximate shortest vector” algorithm of [19], we find, in polynomial time, a vector of \( L_2 \)-norm at most \( F = 2(\ell + 2)/\sqrt{\ell + 2} \cdot N^{1/(\ell+1)} \cdot K^{(\ell+1)/2} \). For sufficiently large \( M \) (any \( M > F \) will do), all vectors with \( L_2 \)-norm at most \( F \) have a last coordinate identical to 0, and thus yield a sequence of \( c_i \)'s satisfying \( \sum_{j=0}^{\ell} c_j r^j \equiv 0 \pmod{N} \) and \( |c_i \cdot K^i| \leq F \). This sequence is as required in Step 1.

Now we move on to Step 2 of List-Decode. We argue next any solution to the list-decoding problem is a root of the polynomial whose coefficients are given by any solution to Step 1.

**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, \( \sum_{j=0}^{\ell} c_j m^j \) is small in absolute value:

\[
\left| \sum_{j=0}^{\ell} c_j m^j \right| \leq (\ell + 1) \cdot \max_j \{|c_j m^j|\} \\
\leq (\ell + 1) \cdot \max_j \{|c_j K^j|\} \\
\leq (\ell + 1) \cdot F.
\]

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 \( P = \prod_{(i | [m]_{p_i})} p_i \). 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 \( P > 2(\ell + 1)F \) 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 lemma describes the performance in terms of amplitude (for any choice of \( \ell \)).

**Proposition 10** For any choice of the parameter \( \ell \), List-Decode \((p_1, \ldots, p_n, k, r_1, \ldots, r_n)\) produces a list of up to \( \ell \) 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 \( \tilde{r} \) is at least \( 2(\ell + 2)^3/2^2(\ell + 2)^2 N^{1/(\ell+1)} K^{(\ell+1)/2} \).
Proof: By Lemma 8, $c_i$'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 $\sum c_j x^j$, and thus is included in the output.

The following theorem is obtained by optimizing the choice of the parameter $\ell$ in the above proposition. See appendix for a proof.

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

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

Comparison with [3, 30] Our algorithm List-Decode is similar to those of [3, 30] 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 7) is similar to an analogous step in [30]; 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 [3, 30] - again our proofs are different since our criteria are different.

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 $\mathbb{Z}_{2^s}$, 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 time complexity of solving this distributional problem.

**Lemma 12** ([2] following [22]; cf., [7]) Suppose there exists a probabilistic polynomial time algorithm $\mathcal{A}'$ and a polynomial $r : \mathbb{Z} \to \mathbb{Z}$ such that on input $M$, an $s \times s$ matrix of 2s-bit integer elements, $\mathcal{A}'(M)$ 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 $\mathcal{A}'$). Then $\mathbb{P}^{\#P} = \mathbb{BPP}$.

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 $s$,

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

Then there exists a randomized poly$(s/\epsilon(s))$-time algorithm $A'$ that on input an $s \times s$ matrix $M$ with entries from $\mathbb{Z}_{2^s}$, 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 $n = \text{poly}(s/\epsilon(s))$, $n' = O(s/\epsilon(s)^2)$
- 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 $[\text{perm}(M)]_{p_i}$ */
  
  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 $[\text{perm}(M + jR)]_{p_i}$ */
    
    Let $v_j = A(M + jR, p_i)$;
  
  * Reconstruct a list of all degree $s$ univariate polynomials $\{f_1, \ldots, f_{n'}\}$ that satisfy $f_i(j) = v_j$ for at least an $\epsilon(s)/16$ fraction of the $v_j$’s.
  
  * Uniformly select a random $h \in \{1, \ldots, n'\}$ 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 $r_i = [\text{perm}(M)]_{p_i}$ */
  
  - Reconstruct a list of all integers $x \leq s!2^{2^s}$ 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 $K = s!2^{2^s} < 2^s2^s$ and $\forall i$, $p_i \geq 2^{s/2}$), and $r_1, \ldots, r_n$.

The polynomial reconstruction step may be performed using the algorithm of [30], which requires $n' \geq 2s \cdot (\epsilon(s)/16)^2$. (To recover polynomials of degree $s$ from a list of values at $n'$ places, the algorithm requires the agreement $t'$ to satisfy $t' > \sqrt{2s/\epsilon(s)^4}$.) 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 $t = \Omega(\sqrt{k/n}) = \Omega(\sqrt{s/n})$.)

Let $P_s$ denote the set of primes in the interval $[2^{s/2}, 2^s]$. Let $D'_s$ 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 $D_s$), 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 $\mathbb{Z}_{2^s}$). We notice that the statistical difference between the two distributions is at most $O\left(\frac{2^{s/2}/s}{2^{s/2}}\right) + s^2 \cdot \frac{2^s}{2^{s/2}}$, 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 module a prime). In particular this implies that

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

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

$$\Pr_{T \in \mathbb{Z}_p^\times} [A(T, p') = \text{perm}(T)_{p'}] \geq \frac{\epsilon(s)}{4}.$$ 

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

$$\Pr_{R \in \mathbb{Z}_p^\times} [A(M + jR, p') = \text{perm}(M + jR)_{p'}] \geq \frac{\epsilon(s)}{4}.$$ 

Say that a matrix $R$ is compatible with $p'$ if

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

(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 $p'$, the probability that $\text{Mod-Perm}(M, p')$ returns the correct value of $\text{perm}(M)_{p'}$ is at least \( \frac{\epsilon(s)}{8} \cdot \frac{\epsilon(s)}{16} \cdot \frac{1}{s} \) — the first term is of the probability that $R$ is compatible with $p'$; the second 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 is the probability of selecting the correct index $h$. As $\ell' \leq 2 : (\epsilon(s)/16)^{-1}$ (cf., [30]), 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 $\text{Mod-Perm}$ that $\text{Mod-Perm}(M, p_i) = \text{perm}(M)_{p_i}$, 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 $r_i = \text{perm}(M)_{p_i}$ is at least $\Omega(\epsilon(s)^4) \cdot n$. In this case List-Decode will succeed in reconstructing a list that includes $\text{perm}(M)$.

By combining Lemma 12 and Lemma 13 we get

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

$$\Pr_{(T, p) \in \mathcal{D}_s} [A(T, p) = \text{perm}(T)_{p}] \geq \frac{1}{q(s)}.$$ 

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

\[11\]
References


Appendix

Proof of Theorem 11: Suppose we want to find all codewords which agree with $\langle r_1, \ldots, r_n \rangle$ on $t$ coordinates. Setting $f_1 = (\ell + 2)^{3/2}$ and $f_2 = (\ell + 2)^{\ell+2}/2$, and applying Proposition 10, it suffices to show that

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

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

$$t_1 \geq f_1$$
$$t_2 \geq f_2$$
$$\prod_{i=1}^{t_3} p_i \geq \left( \prod_{i=t_3+1}^{n} p_i \right)^{1/(\ell+1)} \cdot \left( \prod_{i=1}^{k} p_i \right)^{(\ell+1)/2}$$

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

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

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

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

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

and Eq. (6) follows. □

Setting $\ell+1 = \left\lceil \frac{2n \log p_n}{k \log p_1} \right\rceil$, shows that $t_3 = \sqrt{2kn \log p_n} + \frac{k}{2}$ suffices to achieve Eq. (5). This setting of $\ell$ 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

$t_1 = \frac{3 \log(\ell+2)}{2 \log p_1}$, which is smaller than $2 \log t_2/\log p_1 \ll t_2$

Thus we find that it suffices to have

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

\[= \sqrt{2(k + 3)n \log p_n + \frac{k + 6}{2}} \]

Setting \( e < n - t \) yields the theorem. \( \blacksquare \)