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 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 $m \leq \prod_{i=1}^{k} p_i$ 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 $(m \bmod p_i) = r_i$ 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 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 that 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 #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 $p_1, \ldots, p_k$, 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 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 $\{(m \bmod p_1), \ldots, (m \bmod p_n)\}$ and $\{(m' \bmod p_1), \ldots, (m' \bmod p_n)\}$ 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[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 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 $0 \le r_i < p_i$; and an integer k; find an integer $m < \prod_{i=1}^{k} p_i$ satisfying $(m \bmod p_i) \ne r_i$ 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 $0 \le r_i < p_i$; 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., [13].)

[1] 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 $p_1, \ldots, p_t$, 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 [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 $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$.

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 $K \stackrel{\text{def}}{=} \prod_{i=1}^{k} p_i$. 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. 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:

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, $r = [\sum_{i=1}^{n} c_i \cdot Q_i \cdot r_i]_Q$, 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 $n - k$. 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 $n - k$ 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 $e \leq (n-k)\frac{\log p_1}{\log p_1 + \log p_n}$.

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 $z \stackrel{\text{def}}{=} y \cdot m$ 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:

$\mathsf{Unique\text{-}Decode}(p_1, \ldots, p_n, k, r_1, \ldots, r_n)$.

- 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.
$$\left.\begin{array}{ccccc} 1 & \leq & y & \leq & E \\ 0 & \leq & z & \leq & F \\ \multicolumn{5}{c}{y \cdot r \equiv z \pmod{N}} \end{array}\right\} \qquad (1)$$
2. Let $I \stackrel{\text{def}}{=} \{i : [y]_{p_i} = 0\}$. 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 . 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* $\mathsf{Unique\text{-}Decode}(p_1, \ldots, p_n, k, r_1, \ldots, r_n)$ *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: 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 $y \leq E$. Since $m < K$, we have $z = m \cdot y \leq (K-1) \cdot E$. 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 $z = ym \equiv yr \pmod{p_i}$. ∎

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

Proof: For every i s.t. $[m]_{p_i} = r_i$, 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 $T = \prod_{\{i \mid [m]_{p_i} = r_i\}} p_i \geq N/E$ 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 and $m \cdot y$ are both at most $(K-1)E < N/E$. 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 $y \cdot r \equiv z \pmod N$, it follows that for every i $y \cdot r \equiv y \cdot m \pmod{p_i}$, and so for every $i \in I = \{j : y \not\equiv 0 \pmod{p_j}\}$, we have $r \equiv m \pmod{p_i}$. Thus, $m \in \mathbb{Z}_K$ 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 $\bar{I} = \{1, \ldots, n\} \setminus I$ (i.e., $\forall i \in I,\ y \equiv 0 \pmod{p_i}$). By CRT, $y \equiv 0 \pmod{\prod_{i \in \bar{I}} p_i}$, and since $y > 0$ it follows that $y \geq \prod_{i \in \bar{I}} p_i$. Since $y \leq E$, we have $\prod_{i \in \bar{I}} p_i \geq N/E > K$. Thus, again by CRT, the message $m \in \mathbb{Z}_K$ 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 Unique-Decode$(p_1, \ldots, p_n, k, r_1, \ldots, r_n)$ *solves the error-correction problem in polynomial time for values of any value of the error parameter up to $(n-k)\frac{\log p_1}{\log p_1 + \log p_n}$, 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 $E \leq \prod_{i=n-e+1}^{n} p_i$, Lemma 5 can be applied if $\prod_{i=1}^{n} p_i > (\prod_{i=1}^{k} p_i) \cdot (\prod_{i=n-e+1}^{n} p_i)^2$, which is equivalent to $\prod_{i=k+1}^{n-e} p_i > \prod_{i=n-e+1}^{n} p_i$. In turn this condition holds if $p_1^{n-(e+k)} > p_n^e$. 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, 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.

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 $F = 2^{(\ell+2)/2} \cdot \sqrt{\ell+2} \cdot N^{1/(\ell+1)} \cdot K^{(\ell+1)/2}$, with ℓ to be determined shortly.

Let $r \in \mathbb{Z}_N$ s.t. $[r]_{p_i} = r_i$ for every i.

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

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

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 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. We will do so after the following lemma. 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 *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. ▌

Lemma 8 (Algorithm for Step 1.) *c_i's as required in Step 1 of* List-Decode *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 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:

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

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 $F = 2^{(\ell+2)/2} \cdot \sqrt{\ell+2} \cdot N^{1/(\ell+1)} \cdot K^{(\ell+1)/2}$. 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 $\sum_i c_i r^i \equiv 0 \pmod{N}$ 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 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, $\sum_j c_j m^j$ is small in absolute value:

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

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 lemma 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 $2(\ell+2)^{3/2} 2^{(\ell+2)/2} N^{1/(\ell+1)} K^{(\ell+1)/2}$.*

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 $\sum_j c_j x^j$, and thus is included in the output. ▮

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

Theorem 11 List-Decode$(p_1, \ldots, p_n, k, r_1, \ldots, r_n)$ *with parameter* $\ell = \left\lceil \sqrt{\frac{2n \log p_n}{k \log p_1}} - 1 \right\rceil$ *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 $\epsilon \to 0$.

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 $\mathbb{Z}_{2^{2s}}$, 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 [22]; cf., [7]) *Suppose there exists a probabilistic polynomial time algorithm* 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,* $A'(M)$ *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 $\epsilon : \mathbb{Z} \to [0,1]$ 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 $\mathrm{poly}(\epsilon(s))$ (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 $x \leq s!2^{s^2}$ 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 $\forall i,\ p_i \geq 2^{s/2}$), 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 $t' > \sqrt{2sn'}$.) 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 module a prime). In particular this implies that

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

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

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

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

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

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

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

(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 $\mathsf{Mod\text{-}Perm}(M,p')$ returns the correct value of $[\mathrm{perm}(M)]_{p'}$ is at least $\frac{\epsilon(s)}{8}\cdot\frac{\epsilon(s)}{16}\cdot\frac{1}{\ell'}$ – 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 terms is the probability of selecting the correct index h. As $\ell' \leq 2\cdot(\epsilon(s)/16)^{-1}$ (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 Mod-Perm that $\mathsf{Mod\text{-}Perm}(M,p_i) = [\mathrm{perm}(M)]_{p_i}$, 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 $\mathrm{perm}(M)$. ∎

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,*

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

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

References

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, pages 99-108, Philadelphia, Pennsylvania, 22-24 May 1996.

[2] A. Amir, R. Beigel, and W. Gasarch. Cheatable, P-terse, and P-superterse sets, manuscript, Dec. 1998.

[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. Preliminary version in *FOCS*, 1992.

[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. *SICOMP*, Vol. 13, pages 850–864, 1984. Preliminary version in *23rd FOCS*, 1982.

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

[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 Laboratory of Electronics, MIT, 1957.

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

[11] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing, pages 32-42, New Orleans, Louisiana, 6-8 May 1991.

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

[13] 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.

[14] S. Goldwasser and S. Micali. Probabilistic Encryption. *JCSS*, Vol. 28, No. 2, pages 270–299, 1984. Preliminary version in *14th STOC*, 1982.

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

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

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

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

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

[21] 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.

[22] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *JACM*, Vol. 39, No. 4, pages 859–868, 1992. Preliminary version in *31st FOCS*, 1990.

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

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

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

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

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

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

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

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

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

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

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

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 $f_1 \stackrel{\text{def}}{=} (\ell+2)^{3/2}$ and $f_2 \stackrel{\text{def}}{=} 2^{(\ell+2)/2}$, and applying Proposition 10, it suffices to show that

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

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

$$p_1^{t_1} \geq f_1 \tag{3}$$

$$p_1^{t_2} \geq f_2 \tag{4}$$

$$\text{and } \prod_{i=1}^{t_3} p_i \geq \left(\prod_{i=1}^{n} p_i \right)^{1/(\ell+1)} \cdot \left(\prod_{i=1}^{k} p_i \right)^{(\ell+1)/2} \tag{5}$$

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

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

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

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

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

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

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

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

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

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