Finding Smooth Integers in Short Intervals Using CRT Decoding

Dan Boneh

Computer Science Dept.
Stanford University

E-mail: dabo@cs.stanford.edu

We present a new algorithm for CRT list decoding. An instance of the CRT list decoding problem consists of integers $B, \langle p_1, \ldots, p_n \rangle$ and $\langle r_1, \ldots, r_n \rangle$, where $p_1 < p_2 < \cdots < p_n$ is a sequence of relatively prime integers. The CRT list decoding problem is to find all positive integers $x < B$ such that $x = r_i \bmod p_i$ for all but e values of $i \in \{1, \ldots, n\}$. Suppose $B = \prod_{i=1}^{k} p_i$ for some integer k. Goldreich, Ron, and Sudan recently gave several applications for this problem and presented the first efficient algorithm that works whenever e (approximately) satisfies $e < n - \sqrt{2kn \frac{\log p_n}{\log p_1}}$. Our new algorithm achieves the stronger bound $e < n - \sqrt{kn \frac{\log p_n}{\log p_1}}$ (approximately). The improvement is significant when k is relatively close to n, e.g. $k > n/3$. The bounds we obtain are similar to the bounds obtained by Guruswami and Sudan for Reed-Solomon list decoding. Hence, our algorithm reduces the gap between CRT list decoding and list decoding of Reed-Solomon codes. In addition, we give a new application for CRT list decoding: finding smooth integers in short intervals. Problems of this type come up in several algorithms for factoring large integers. We define and solve a generalized CRT list decoding problem and discuss how it might be used within the quadratic sieve factoring method.

---

1. INTRODUCTION

It is well known that given d pairs $(x_i, y_i) \in \mathbb{F}^2$, for which the x_i's are distinct, there is a unique polynomial $f \in \mathbb{F}[x]$ of degree at most $d-1$ such that $y_i = f(x_i)$ for all i (*polynomial interpolation*). Similarly, it is well known that given d pairs $(x_i, p_i) \in \mathbb{Z}^2$, for which the p_i's are relatively prime, there is a unique positive integer $X < \prod_{i=1}^{d} p_i$ such that $X = x_i \bmod p_i$ for all i (*Chinese Remainder Theorem*). These two facts about these Euclidean rings are analogous in many ways. Many algorithmic questions related to polynomial interpolation have an analogous algorithmic question related to the Chinese Remainder Theorem (CRT). Some examples are given in [10] and in Section 5.

In this paper we study this analogy for a classic polynomial interpolation problem known as decoding of Reed-Solomon codes. The problem reduces to the following question: given n pairs $(x_i, y_i) \in \mathbb{F}_q^2$, for which the x_i's are distinct, find all polynomials $f \in \mathbb{F}_q[x]$ of degree at most k such that $y_i = f(x_i)$ for all but e values of $i \in \{1, \ldots, n\}$. It is easy to see that when $e < (n-k)/2$ the solution is unique. The solution can be efficiently found using a classic algorithm due to Berlekamp and Massey (see [2, 19] for a description). Surprisingly, it is possible to decode beyond the Berlekamp-Massey bound, however the solution is no longer unique. In a recent seminal work Guruswami and Sudan [22, 12] show that as long as $e < n - \sqrt{kn}$ it is possible to efficiently recover a list of *all* polynomials f satisfying $y_i = f(x_i)$ for all but e values. This decoding problem is known as the *list decoding* problem for Reed-Solomon codes.

Considering the analogy between interpolation and CRT it is natural to study the problem analogous to Reed-Solomon decoding, namely CRT decoding [21, 10]. Let $p_1 < p_2 < \ldots < p_n$ be relatively prime integers. The CRT code is defined as the set of all codewords in $\mathbb{Z}^n$ of the form $\vec{x} = \langle x \bmod p_1, \ldots, x \bmod p_n \rangle$ where $x \in \{0, \ldots, B-1\}$. The code contains B codewords. The error correction property comes from the fact that two code words can match in at most k coordinates where $k = \lfloor \log B / \log p_1 \rfloor$. Indeed, if two codewords $\vec{x_1}, \vec{x_2}$ match on $k+1$ coordinates then $x_1 = x_2 \bmod M$ where $M \geq p_1^{k+1} > B$, but then $x_1 = x_2$. Decoding the CRT code reduces to the following list decoding problem:

CRT list decoding: An instance of the problem consists of $B, \langle p_1, \ldots, p_n \rangle$ and $\langle r_1, \ldots, r_n \rangle$. The values $0 < p_1 < p_2 < \ldots < p_n$ are relatively prime integers (not necessarily prime), and $B, r_1, \ldots, r_n \in \mathbb{Z}$. Find all positive integers $x < B$ such that $x = r_i \bmod p_i$ for all but e values of $i \in \{1, \ldots, n\}$ (here e is some predefined function of B, n, p_1 and p_n).

CRT list decoding has several applications beyond decoding the CRT code [10]. The question is, what is the maximal value of e for which CRT list decoding can be solved in polynomial time (in n and $\log p_n$)? We refer to [10] for a history of this problem. Recently Goldreich, Ron and Sudan [10] gave the first efficient algorithm for CRT list decoding. Their algorithm works as long as e (approximately) satisfies $e < n - \sqrt{2kn \frac{\log p_n}{\log p_1}}$. We refer to this expression as the GRS bound. When $\log p_n = (1+\epsilon)\log p_1$ for some small $\epsilon > 0$ the GRS bound is close to the Guruswami-Sudan bound for Reed-Solomon list decoding mentioned above. However, there is an extra factor of $\sqrt{2}$ in the GRS bound that does not appear in the Guruswami-Sudan bound (the $\sqrt{2}$ appears in an earlier work of Sudan [22]). Hence, although the decoding problems are analogous, there is a gap between our ability to decode Reed-Solomon codes and our ability to decode the CRT code. This gap is significant when the code rate $\epsilon = k/n$ satisfies $\epsilon > 1/3$.

Our first result is an improved CRT list decoding algorithm that reduces the gap between the polynomial and CRT decoding problems. We give a decoding algorithm that works whenever $e < n - \sqrt{kn \frac{\log p_n}{\log p_1}}$ (no $\sqrt{2}$). In Section 3 we give a new application for CRT list decoding: finding strongly smooth integers in short intervals. CRT list decoding gives an algorithm for this task that runs in time $(\log |I|)^{O(1)} \cdot s^{O(1)}$ where $|I|$ is the width of the interval and s is the smoothness bound. Our results also give strong bounds on the number of strongly smooth

integers in short intervals. Techniques for locating smooth integers are often used in algorithms for factoring large integers. In Section 4, Motivated by the quadratic sieve factoring algorithm, we define a generalized CRT list decoding problem and give an algorithm for solving it.
Throughout the paper, unless otherwise stated, all logarithms are base 2.

1.1. Lattices

Our results make use of algorithms for lattice basis reduction. We state a few basic results about lattices and refer to [18, Ch. 1] for an introduction. Let $u_1, \ldots, u_d \in \mathbb{Z}^d$ be linearly independent vectors. A lattice L spanned by $\langle u_1, \ldots, u_d \rangle$ is the set of all *integer* linear combinations of $u_1, \ldots, u_d$. The determinant of L is defined as the determinant of the $d \times d$ matrix whose rows are the basis vectors $u_1, \ldots, u_d$. We say that L is a lattice of dimension d.

A classic result due to Hermite shows that every lattice L of dimension d contains a short vector v whose L_2 norm satisfies $\|v\| < c\sqrt{d} \cdot \det(L)^{1/d}$. Here c is Hermite's constant which is known to be less than $1/\sqrt{e\pi}$. The celebrated LLL algorithm [17] is a constructive version of Hermite's theorem. It finds a short vector in L whose length is close to Hermite's bound.

THEOREM 1.1 (LLL). *Let L be a lattice spanned by $\langle u_1, \ldots, u_d \rangle$. The LLL algorithm, given $\langle u_1, \ldots, u_d \rangle$, produces a vector v satisfying: $\|v\| \le 2^{d/2} \det(L)^{1/d}$. The algorithm runs in time quartic in the size of its input.*

Next, we state a simple lemma that will be used throughout the paper. The lemma gives a sufficient condition for when a modular root of a polynomial is also a root of the same polynomial over the integers. We define the norm of a polynomial $w(x) = \sum_i c_i x^i$ as $\|w(x)\|^2 = \sum_i c_i^2$.

LEMMA 1.1. *Let S and B be positive integers and let $w(x) \in \mathbb{Z}[x]$ be a polynomial of degree $d-1$. Suppose that*
a. $w(x_0) = 0 \bmod S$ for some integer x_0 where $|x_0| < B$, and
b. $\|w(xB)\| < S/\sqrt{d}$.
Then $w(x_0) = 0$ holds over the integers.

Proof. Let $w(x) = c_0 + \ldots + c_{d-1}x^{d-1}$. Observe that

$$\begin{aligned} |w(x_0)| &= \left|\sum c_i x_0^i\right| = \left|\sum c_i B^i \left(\frac{x_0}{B}\right)^i\right| \le \\ &\le \sum \left|c_i B^i \left(\frac{x_0}{B}\right)^i\right| \le \sum \left|c_i B^i\right| \le \sqrt{d}\,\|w(xB)\| < S \end{aligned}$$

but since $w(x_0) \equiv 0$ modulo S we have that $w(x_0) = 0$. ■

We also state an important result due to Coppersmith [9] that will be used in Section 5.

THEOREM 1.2 (Coppersmith). *Let $N > 0$ be an integer and let $f(x) \in \mathbb{Z}_N[x]$ be a monic polynomial of degree d. Then there is a polynomial time algorithm to find all $x_0 \in \mathbb{Z}$ such that $f(x_0) = 0 \bmod N$ and $|x_0| < N^{1/d}$.*

Coppersmith's algorithm makes use of lattice basis reduction and the LLL algorithm. We refer to [14] for an alternate proof of Theorem 1.2.

2. IMPROVED CRT LIST DECODING

In this section we present the new algorithm for CRT list decoding. The algorithm uses a lattice that is dual to the one used by Goldreich et al. [10]. The improved bounds are obtained by increasing the dimension of this lattice. These extra dimensions are derived from a technique due to Coppersmith [9] (and used in the proof of Theorem 1.2), and from some recent applications [14, 7, 6]. The CRT decoding algorithm relies on the following notion of *amplitude*:

DEFINITION 2.1. Let $B, \langle p_1, \ldots, p_n \rangle$ and $\langle r_1, \ldots, r_n \rangle$ be an instance of the CRT list decoding problem. Set $P = \prod_{i=1}^n p_i$. Let R be an integer such that $R = r_i \bmod p_i$ for all $i = 1, \ldots, n$. For an integer m, we define the *amplitude* of m with respect to the given CRT list decoding problem as:

$$\mathrm{amp}(m) = \gcd(P, m - R)$$

Observe that if $p_1, \ldots, p_n$ are distinct primes then $\mathrm{amp}(m)$ is equal to the product of all the p_i's for which $m = r_i \bmod p_i$. This follows from the fact that p_i divides $\mathrm{amp}(m)$ if and only if $m = r_i \bmod p_i$. Hence, in this case, the amplitude measures the "weight" of all the p_i's for which $m = r_i \bmod p_i$. When the p_i's are not all prime the amplitude of m could be a bit higher than the product of the p_i's for which $m = r_i \bmod p_i$. For example, if q divides p_i the amplitude of m could contain the factor q even though $m \neq r_i \bmod p_i$. This fact will be used in the next section. We note that Goldreich et al. [10] define $\mathrm{amp}(m)$ as $\prod_{m=r_i\ (p_i)} p_i$ regardless of whether the p_i's are prime or not.
The main result of this section follows from the following theorem:

THEOREM 2.1. *Let $B, \langle p_1, \ldots, p_n \rangle$ and $\langle r_1, \ldots, r_n \rangle$ be an instance of the CRT list decoding problem. Define $P = \prod_{i=1}^n p_i$. Then, for any $d \geq 1 + \sqrt{\frac{\log P}{\log B}}$, there is an algorithm that outputs all positive integers $m < B$ for which*

$$amp(m) > P^{\epsilon} \quad where \quad \epsilon = \sqrt{\frac{\log 4B}{\log P}} + \frac{5}{4d}$$

The algorithm's running time is dominated by the time it takes to run LLL on a lattice of dimension d. When $d \geq 20 log P$ the algorithm outputs all positive $m < B$ for which $amp(m) > P^{\sqrt{\frac{\log 4B}{\log P}}}$.

Proof. The outline of the algorithm is similar to the algorithm in [10]. The algorithm works in three steps:
Step 1. Construct a polynomial $w(x)$ over the integers so that all required m are roots of $w(x)$.
Step 2. Find all integer roots of $w(x)$.

$$
\begin{array}{r}
\\
g_0(xB): \\ g_1(xB): \\ g_2(xB): \\ g_3(xB): \\ h_0(xB): \\ h_1(xB): \\ h_2(xB):
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} 1 & x & x^2 & x^3 & x^4 & x^5 & x^6 \end{array} \\
\begin{bmatrix}
P^4 & & & & & & \\
-RP^3 & P^3B & & & & & \\
R^2P^2 & -2RP^2B & P^2B^2 & & & & \\
-R^3P & 3R^2PB & -3RPB^2 & PB^3 & & & \\
R^4 & -4R^3B & 6R^2B^2 & -4RB^3 & B^4 & & \\
 & R^4B & -4R^3B^2 & 6R^2B^3 & -4RB^4 & B^5 & \\
 & & R^4B^2 & -4R^3B^3 & 6R^2B^4 & -4RB^5 & B^6
\end{bmatrix}
\end{array}
$$

FIG. 1. Lattice obtained when $a = 4$ and $a' = 3$

Step 3. Output all integer roots m for which $\text{amp}(m)$ satisfies the required bound.

The bulk of the work is done in Step 1. Step 2 can be done efficiently using a number of root finding algorithms in $\mathbb{R}$ (for example, see [8, Ch. 3] or [17]). Our improved bounds are due to a new algorithm for Step 1.

Let $R \in [0, P]$ be an integer such that $R = r_i \bmod p_i$ for all $i = 1, \dots, n$. Such an R exists by the Chinese Remainder Theorem. Let a and a' be two integer parameters that will be determined later. We define the following two families of polynomials in $\mathbb{Z}[x]$:

$$
\begin{aligned}
g_i(x) &= P^{a-i} \cdot (x-R)^i \qquad &&\text{for } i = 0, \dots, a-1 \\
h_i(x) &= (x-R)^a \cdot x^i \qquad &&\text{for } i = 0, \dots, a'-1
\end{aligned}
$$

We first explain the significance of these two families. Let $0 < m < B$ be an integer for which $\text{amp}(m)$ satisfies the bound of Theorem 2.1. Set $M = \text{amp}(m)$. By definition of R we know that $m - R = 0 \bmod M$ (since $\text{amp}(m)$ divides $m - R$). Consequently, by definition of g_i and h_i we have that

$$
\begin{aligned}
g_i(m) = 0 \pmod{M^a} \qquad &\text{for all } i = 0, \dots, a-1 \text{ and,} \\
h_i(m) = 0 \pmod{M^a} \qquad &\text{for all } i = 0, \dots, a'-1
\end{aligned}
$$

Note that if $w(x)$ is an integer linear combination of the g_i's and h_i's then $w(m) = 0 \bmod M^a$. Furthermore, recall that by Lemma 1.1, if the polynomial $w(xB)$ has low norm (i.e. norm less than $M^a/\sqrt{deg(w)}$) then m will be a root of $w(x)$ over the integers.

The observations above show that to find the required polynomial $w(x)$ needed in Step 1 it is sufficient to find an integer linear combination of the polynomials $g_i(xB)$ and $h_i(xB)$ that has low norm. To do so, we construct a lattice L of dimension $d = a + a'$. The lattice is spanned by the coefficient vectors of the polynomials $g_i(xB)$ for $i = 0, \dots, a$ and $h_i(xB)$ for $i = 0, \dots, a'$. For example, taking $a = 4$ and $a' = 3$ the resulting lattice L of dimension 7 is spanned by the rows of the matrix in Figure 1.

We find a short vector in L using the LLL algorithm (Fact 1.1). The algorithm will produce a short vector which we view as the coefficients vector of a polynomial

$w(xB)$. By Fact 1.1 we know that this polynomial $w(xB)$ satisfies $\|w(xB)\| < 2^{d/2}\det(L)^{1/d}$. Hence, when

$$2^{d/2}\det(L)^{1/d} < M^a/\sqrt{d}$$

we are guaranteed that LLL will produce a polynomial $w(xB)$ whose norm is less than $M^a/\sqrt{d}$. Then by Lemma 1.1, $w(x)$ must have m as a root over the integers. Rearranging terms in the above inequality we obtain the equivalent bound:

$$M > \det(L)^{1/ad} \cdot \gamma(d)^{1/a} \quad \text{where} \quad \gamma(d) = 2^{d/2}\sqrt{d} \tag{1}$$

It remains to compute the determinant of L. Since the given basis of L forms a triangular matrix, the determinant of L is simply the product of the elements on the diagonal. Hence, $\det(L) = P^{a(a+1)/2}B^{d(d-1)/2}$. Plugging $\det(L)$ into the expression above leads to:

$$M > P^{(a+1)/2d}B^{(d-1)/2a} \cdot \gamma(d)^{1/a} = P^{\frac{1}{2d}} \cdot P^{\frac{a}{2d}+\frac{1}{a}\left(\frac{\log B}{\log P}\frac{d-1}{2}+\frac{\log\gamma(d)}{\log P}\right)}$$

We are free to choose $a \in \{1,\ldots,d\}$ so as to minimize the right hand side. For simplicity, set $C = 1/2d$ and $D = \frac{\log B}{\log P}\frac{d-1}{2} + \frac{\log\gamma(d)}{\log P}$. Then we want an integer $a \in \{1,\ldots,d\}$ that minimizes $P^{C+aC+D/a}$. The optimal value is achieved at

$$a_{opt} = \sqrt{D/C} = d \cdot \sqrt{\frac{(1-\frac{1}{d})\log B}{\log P} + \frac{2\log\gamma(d)}{d\log P}} \tag{2}$$

Let $[a_{opt}]$ be the closest integer to a_{opt}. Since $d \geq 1+\sqrt{\frac{\log P}{\log B}}$ we know that $a_{opt} \geq 1$. Hence,

$$[a_{opt}] \leq a_{opt} + \tfrac{1}{2} \quad \text{and} \quad \frac{1}{[a_{opt}]} \leq \frac{1}{a_{opt}-\frac{1}{2}} \leq \frac{1}{a_{opt}} + \frac{1}{a_{opt}^2}$$

Therefore,

$$C + [a_{opt}]C + \frac{D}{[a_{opt}]} \leq C + (a_{opt}+\frac{1}{2})C + (\frac{1}{a_{opt}} + \frac{1}{a_{opt}^2})D = 2\sqrt{CD} + \frac{5}{2}C$$

Plugging in the values for C, D we see that Condition (1) is satisfied as long as $M > P^\epsilon$ where

$$\epsilon = \sqrt{\frac{\log B}{\log P}\left(1-\frac{1}{d}\right) + \frac{2\log\gamma(d)}{d\log P}} + \frac{5}{4d}$$

Observe that $\frac{2\log\gamma(d)}{d} < 1.75$ for all $d > 0$. Furthermore, we can simplfy the bound by dropping the $1-\frac{1}{d}$ factor. This shows that Condition (1) holds whenever:

$$M > P^\epsilon \quad \text{where} \quad \epsilon = \sqrt{\frac{1.75+\log B}{\log P}} + \frac{5}{4d} \tag{3}$$

which is slightly stronger than the bound given in Theorem 2.1.

To summarize, when $M > P^\epsilon$ any short vector in L corresponds to a polynomial $w(x)$ with the following property: if $0 < m < B$ and $\mathrm{amp}(m) > M$ then m is a root of $w(x)$ over the integers. Consequently, the complete algorithm for Step 1 is as follows:
Step 1a. Given an integer $d > 3$ create the $d \times d$ matrix L whose rows are the coefficients of the polynomials $g_0(xB), \dots, g_{a-1}(xB)$ and $h_0(xB), \dots, h_{d-a-1}(xB)$ where a is the closest integer to the value defined in equation (2).

Step 1b. Run the LLL algorithm on the d-dimensional lattice spanned by the rows of the matrix L. Let $\bar{v} \in \mathbb{Z}^d$ be the resulting short vector in L.

Step 1c. View $\bar{v}$ as the coefficients of a $d-1$ degree polynomial $w(xB)$. Output $w(x)$ as the required polynomial.

We then use a root finding algorithm in $\mathbb{R}$ to find all integer roots of $w(x)$ and output those roots m that satisfy $\mathrm{amp}(m) > P^\epsilon$.

To complete the proof it remains to evaluate the bound of (3) for $d \geq 20 \log P$. Observe that:

$$\sqrt{1.75 + \log B} = \sqrt{\log(4B) - \frac{1}{4}} \leq \sqrt{\log(4B)} - \frac{1/8}{\sqrt{\log(4B)}}$$

Hence, when $d \geq 10\sqrt{\log P \log 4B}$ we obtain that the bound on ϵ in (3) satisfies:

$$\sqrt{\frac{1.75 + \log B}{\log P}} + \frac{5}{4d} \leq \sqrt{\frac{\log(4B)}{\log P}} - \frac{1/8}{\sqrt{\log P \log(4B)}} + \frac{5}{4d} \leq \sqrt{\frac{\log(4B)}{\log P}}$$

Finally, since $P > B \geq 1$ we know that $d \geq 10\sqrt{\log P \log 4B}$ holds whenever $d > 20 \log P$. ■

Improved CRT list decoding is an immediate corollary of the previous theorem.

COROLLARY 2.1. *Let $B, \langle p_1, \dots, p_n \rangle$ and $\langle r_1, \dots, r_n \rangle$ be an instance of the CRT list decoding problem. We wish to find all positive integers $m < B$ satisfying $m = r_i \bmod p_i$ for all but e elements $i \in \{1, \dots, n\}$. Then there is a polynomial time algorithm for this task as long as*

$$e \leq n - \epsilon n \frac{\log p_n}{\log p_1} \quad \text{where} \quad \epsilon = \sqrt{\frac{\log 4B}{\log P}}$$

Proof. Let $P = \prod_{i=1}^n p_i$. By Theorem 2.1 there is a polynomial time algorithm to find all integers $m \in [1, B]$ such that $\mathrm{amp}(m) > P^\epsilon$. If $m = r_i \bmod p_i$ for all but e locations then $\mathrm{amp}(m) \geq \prod_{i=1}^{n-e} p_i > p_1^{n-e}$. Hence, if $p_1^{n-e} \geq P^\epsilon$ the algorithm will find all the required m. Since $P \leq (p_n)^n$ it follows that if $p_1^{n-e} \geq (p_n)^{n\epsilon}$ the algorithm will find all the required m. Solving for e we obtain the required bound. ■

Corollary 2.1 is the main result of this section regarding CRT list decoding. To compare this result to the bounds in [10] we re-derive the corollary using the notation in [10]. Namely, we set $B = \prod_{i=1}^k p_i$ for some integer $k > 0$.

COROLLARY 2.2. *Let $B, \langle p_1, \ldots, p_n \rangle$ and $\langle r_1, \ldots, r_n \rangle$ be an instance of the CRT list decoding problem. Suppose $B = \prod_{i=1}^{k} p_i$ for some k. Then there is a polynomial time algorithm that finds all positive integers $m < B$ satisfying $m = r_i \bmod p_i$ for all but e elements $i \in \{1, \ldots, n\}$, as long as*

$$e \leq n - \sqrt{kn \frac{\log p_n}{\log p_1}}$$

Proof. Set $B' = \frac{1}{4}B = \frac{1}{4}\prod_{i=1}^{k} p_i$. We first show how to find all required integers m in $[0, B']$. Set $P = \prod_{i=1}^{n} p_i$ and $\epsilon = \sqrt{\frac{\log 4B'}{\log P}}$. As in the proof of Corollary 2.1 we know that CRT list decoding can be solved efficiently as long as $\prod_{i=1}^{n-e} p_i > P^{\epsilon}$. Taking logarithms of both sides leads to:

$$\sum_{i=1}^{n-e} \log p_i > \sqrt{(\log P)(\log 4B')}$$

In other words, since $4B' = \prod_{i=1}^{k} p_i$, we can decode as long as

$$\sum_{i=1}^{n-e} \log p_i > \sqrt{(\sum_{i=1}^{n} \log p_i)(\sum_{i=1}^{k} \log p_i)}$$

Since $p_1 < p_2 < \ldots < p_n$ this condition is satisfied whenever (see Appendix A):

$$e \leq n - \sqrt{kn \frac{\log p_n}{\log p_1}}$$

Hence, by Theorem 2.1 there is a polynomial time algorithm to find all required m in the interval $[0, B']$. We now show how to find all required m in $[0, B]$. The algorithm for solving the problem in $[0, B']$ enables us to solve the problem in any interval $[U, V]$ of length B'. To see this observe that an integer $m \in [U, V]$ satisfies $m = r_i \bmod p_i$ for all but e elements $i \in \{1, \ldots, n\}$ if and only if m is a solution to the following instance of the CRT list decoding problem: $B', \langle p_1, \ldots, p_n \rangle$ and $\langle r_1 - U, \ldots, r_n - U \rangle$ with the same value of e. Hence, the complexity of CRT list decoding is invariant under tranlation of the codeword interval. We can now find all required m in the interval $[0, B]$ by running the algorithm of Theorem 2.1 to find all such m in the four sub-intervals $[0, \frac{1}{4}B]$, $[\frac{1}{4}B, \frac{1}{2}B]$, $[\frac{1}{2}B, \frac{3}{4}B]$, $[\frac{3}{4}B, B]$ each of length $B' = B/4$. ■

Corollary 2.2 improves on the bound of [10] and shrinks the gap between Reed-Solomon decoding [12] and CRT decoding.

3. FINDING SMOOTH INTEGERS IN SHORT INTERVALS

We show that CRT decoding is closely related to the problem of finding strongly smooth integers in short intervals. In fact, CRT decoding enables us to solve a more general problem: finding all integers in a short interval that have a large smooth factor. In addition, our algorithm yields bounds on the maximal number of integers in a short interval that have a large smooth factor. We begin by defining two notions of smoothness.

DEFINITION 3.1. Let $s > 0$ be an integer.
- An integer N is s-smooth if N has no prime divisors greater than s.
- An integer N is *strongly s-smooth* if N is s-smooth and in addition, if p is a prime divisor of N then the multiplicity of p as a divisor of N is at most $\frac{\log s}{\log p}$. In other words, p^m does not divide N for any m for which $p^m > s$.

Throughout this section we view s as a fixed given value. We let $q_1, \ldots, q_n$ be the set of all primes less than s. We denote by n the number of such primes and assume the set is sorted, i.e. $q_1 = 2$, $q_2 = 3$, $q_3 = 5$, etc. We set $S_0 = \prod_{i=1}^{n} q_i$.

Next, we briefly give the intuition linking CRT decoding with the problem of finding smooth numbers in short intervals. Let $I = [U, V]$ be an interval where $V < 2U$. We denote by $|I|$ the length of the interval, i.e. $|I| = V - U$. Consider the following instance of the CRT list decoding problem: $|I|, \langle q_1, \ldots, q_n\rangle$ and $\langle -U, \ldots, -U\rangle$. We want all positive integers $m < |I|$ such that $m = -U \bmod q_i$ for at least $\log U$ values of $i \in \{1, \ldots, n\}$ (recall that all logarithms are base 2). Assuming the parameters satisfy the conditions of Theorem 2.1, the decoding algorithm will produce integers $0 < m < |I|$ such that $U + m = 0 \bmod q_i$ for at least $\log U$ values of i. Hence, $U + m \in I$ is divisible by at least $\log U$ distinct primes less than s. Since $U + m \leq V < 2U$ it follows that $U + m$ must be s-smooth. In fact, $U + m$ is strongly s-smooth since each prime q_i divides $U + m$ at most once.

The above discussion shows that CRT list decoding can be used to find smooth integers in a given short interval. The problem is that the algorithm will ignore smooth integers $N \in I$ for which some small primes divide N multiple times. To obtain an algorithm for finding all strongly s-smooth integers in a given interval we use Theorem 2.1 directly. For the remainder of the section we set

$$S = \prod_{i=1}^{n} q_i^{a_i} \qquad \text{where} \qquad a_i = \left\lfloor \frac{\log s}{\log q_i} \right\rfloor \tag{4}$$

Note that all strongly s-smooth integers are factors of S.

The following theorem gives an algorithm for finding all integers $N \in [U, V]$ for which N has a large strongly smooth factor. If V is sufficiently small, the algorithm outputs all strongly s-smooth integers in $[U, V]$.

THEOREM 3.1. *Let $I = [U, V]$, let $s > 0$ and $T > 0$ be integers. We wish to find all integers $N \in I$ for which $\gcd(N, S) > T$. For any integer $d \geq 1 + \sqrt{\frac{\log S}{\log |I|}}$ there is a polynomial time algorithm (in d and the size of $\langle U, V, S, T\rangle$) that does so whenever*

$$T > S^{\epsilon} \qquad \textit{where} \qquad \epsilon = \sqrt{\frac{\log 4|I|}{\log S}} + \frac{5}{4d}$$

Solving for $|I|$ and simplifying we get: $|I| < \frac{1}{4} T^{\frac{\log T}{\log S} - \frac{2.5}{d}}$.
In addition, if $V < 2T$ and $|I|$ is sufficiently small so that $T > S^{\epsilon}$ then the algorithm will output all strongly s-smooth integers in $[U, V]$.

Proof. Let $|I|, \langle q_1^{a_1}, \ldots, q_n^{a_n}\rangle$, and $\langle -U, \ldots, -U\rangle$ be an instance of the CRT list decoding problem. Recall that the amplitude of $m < |I|$ with respect to this instance is defined as $\text{amp}(m) = \gcd(U + m, S)$. The first part of the theorem now follows directly from Theorem 2.1. The algorithm described in the proof of the theorem will output all m such that $\gcd(U + m, S) > S^{\epsilon}$ and in particular will output all m for which $\gcd(U + m, S) > T$. Hence, if $T > S^{\epsilon}$ the algorithm will output all $N \in I$ for which $\gcd(N, S) > T$. Solving for $|I|$ gives the required bound.

Now, suppose $V < 2T$. If $N \in I$ then $N < 2T$. Then if $\gcd(N, S) > T$ we must have $\gcd(N, S) = N$, i.e. $N|S$. Hence, N is strongly s-smooth. It follows that when $V < 2T$ and $|I|$ satisfies the required bound the algorithm will output all strongly s-smooth integers in I. ■

Example. Take $s = 1000$. Then $\lfloor \log S \rfloor = 1428$. Take $T = 2^{500}$ and use $d = 50$. Then in any interval I of width $|I| = 2^{148}$ we can find all integers N satisfying $\gcd(N, S) > T$. Previously it was not possible to scan intervals of this size. Furthermore, in the interval $[2T - 2^{148}, 2T]$ we can find all strongly s-smooth integers. The running time is dominated by the time it takes to run LLL on a lattice of relatively small dimension, e.g. dimension 50, whose entries are on the order of S^{50}. This can be done within several hours on a modern computer.

We note that a *random* interval I satisfying the bounds of Theorem 3.1 is unlikely to contain integers with sufficiently large smooth factors. However, the theorem shows that given an interval containing such elements it is possible to efficiently locate all of them.

Alternate approaches. We note that alternate approaches for finding s-smooth integers in $[U, V]$ do not work as well. The most common approach is called *sieving*. It is essentially a clever exhaustive search technique. Sieving is exponential in the *size* of U and V, i.e. exponential in $\log U + \log V$. Another approach uses algorithms for subset sum. Recall that if $N \in [U, V]$ is s-smooth then $N = \prod_{i=1}^{n} q_i^{c_i}$. Hence, it suffices to find a vector $\vec{c} = (c_1, \ldots, c_n)$ such that

$$c_1 \log q_1 + \ldots + c_n \log q_n \in [\log U, \log V]$$

This is a subset sum problem. The difficulty with this approach is that n is $\Omega(s/logs)$. Consequently, the dimension of the lattice used to solve the problem is exponential in the size of the input ($\log s$). In contrast, the algorithm of Theorem 3.1 can use a lattice of fixed dimension, e.g. $d = 50$.

3.1. Bounding the number of integers with large smooth factors

Recall that the algorithm described in the proof of Theorem 2.1 finds solutions to the CRT list decoding problem by constructing a polynomial having all solutions as its roots. The degree of this polynomial bounds the number of solutions. This enables us to obtain a bound on the number of integers in $[U, V]$ with a large strongly s-smooth factor.

THEOREM 3.2. *Let $I = [U, V]$. Let $s > 0$ and $T > 0$ be integers. Then the number of integers $N \in I$ such that* $\gcd(N, S) > T$ *is at most*

$$d_{\max} = max\left[\frac{\frac{5}{4}\log S}{\log T - \sqrt{\log(4|I|)\log S}}\ ,\ 1 + \sqrt{\frac{\log S}{\log |I|}}\ \right]$$

Proof. Let $A \subseteq I$ be the set of integers in $N \in I$ such that $\gcd(N, S) > T$. We bound the size of A. The proof of Theorem 2.1 shows that, for any $d \geq 1 + \sqrt{\frac{\log S}{\log |I|}}$, when

$$T > S^{\epsilon} \qquad \text{where} \qquad \epsilon = \sqrt{\frac{\log 4|I|}{\log S}} + \frac{5}{4d}$$

all integers $N \in A$ are roots of a single polynomial $w(x)$ of degree $d - 1$. Therefore, there are at most $d - 1$ such numbers. The smallest value of d for which the above inequality holds is an upper bound on the number of elements in A.

Taking logarithms of both sides of the bound on T leads to:

$$\log T > \sqrt{\log 4|I| \log S} + \frac{5 \log S}{4d}$$

Solving for d we obtain the required bound. ■

Example. Take $s = 1000$ and $T = 2^{500}$. Then $\lfloor \log S \rfloor = 1428$. Then in any interval I of width $|I| = 2^{100}$ there can be at most 15 integers N with $\gcd(N, S) > 2^{500}$. In particular, in any interval of width 2^{100} contained in $[T, 2T]$ there are at most 15 strongly s-smooth numbers.

Providing good bounds on the number of smooth integers in short intervals is a long standing open problem [15]. Most bounds on the density of smooth integers make use of analytic tools. In contrast, our bounds are derived by purely algebraic (and algorithmic) means.

4. GENERALIZED CRT LIST DECODING

Common integer factoring algorithms such as the quadratic sieve [15] and the number field sieve [16] work by searching for smooth integers. However, rather than searching for smooth integers in a given interval, these algorithms search for integers $x \in [-B, B]$ such that $f(x)$ is s-smooth. Here $f(x)$ is some low degree polynomial and B and s are some predefined parameters.

For example, to factor an integer N the quadratic sieve algorithm uses monic quadratic polynomials of the form:

$$f_a(x) = (x + \lfloor\sqrt{aN}\rfloor)^2 - aN$$

for some small values of a. During the sieving phase the goal is to scan the interval $[-B, B]$ and find integers $x \in [-B, B]$ such that $f_a(x)$ is s-smooth. This is done using a technique called sieving whose running time is exponential in $\log B$. See [15] for a full description of the quadratic sieve.

Motivated by the quadratic sieve we consider the following problem: given a polynomial $f(x)$ find all integers $x \in [-B, B]$ such that $f(x)$ is s-smooth. We show how to solve a slightly more general problem which we call generalized CRT list decoding.

Generalized CRT list decoding: An instance of the problem consists of $B, f, \langle p_1, \ldots, p_n \rangle$ and T. The values $p_1, \ldots, p_n$ are relatively prime integers (not necessarily prime). $f(x) \in \mathbb{Z}[x]$ is a monic polynomial, and B, T are positive integers. Set $P = \prod_{i=1}^{n} p_i$. Find all integers $|x| \leq B$ such that $\gcd(f(x), P) > T$.

The CRT decoding problem considered in Theorem 2.1 is a special case of the above. Simply set $f(x)$ to be the linear polynomial $f(x) = x - R$ where R is defined as in Definition 2.1.

Let S be the product of powers of small primes as defined in equation (4). That is, $S = \prod_{i=1}^{n} q_i^{a_i}$. A solution to the generalized CRT list decoding problem will enable us to find all integers $x \in [-B, B]$ such that $\gcd(f(x), S) > T$. In other words, we will find all $x \in [-B, B]$ such that $f(x)$ has a large strongly s-smooth factor. Note that whenever the smooth factor is as large as $f(x)$ we know that $f(x)$ must be s-smooth.

THEOREM 4.1. *Let $B, f, \langle p_1, \ldots, p_n \rangle$ and T be an instance of the generalized CRT list decoding problem. Let δ be the degree of $f(x)$ and set $P = \prod_{i=1}^{n} p_i$. Then, for any $d > 1 + \sqrt{\frac{\delta \log P}{\log B}}$, there is an algorithm that outputs all integers $|x| < B$ for which* $\gcd(f(x), P) > P^{\epsilon}$ *where*

$$\epsilon = \sqrt{\frac{\delta \log 4B}{\log P}} + \frac{5\delta}{4d}$$

The algorithm's running time is dominated by the time it takes to run LLL on a lattice of dimension d.

Proof. The proof is along the same line as the proof of Theorem 2.1. We construct a polynomial $w(x) \in \mathbb{Z}[x]$ of degree $d-1$ so that all the required x are roots of $w(x)$. We define the following two families of polynomials:

$$\begin{aligned} g_{i,j}(x) &= P^{a-i} \cdot f(x)^i \cdot x^j \quad \text{for} \quad i = 0, \ldots, a-1 \text{ and} \\ & \qquad\qquad\qquad\qquad\qquad\quad j = 0, \ldots, \delta - 1 \\ h_i(x) &= f(x)^a \cdot x^i \qquad\qquad \text{for} \quad i = 0, \ldots, a' - 1 \end{aligned}$$

The values of a, a' will be determined later. Let $|x| < B$ be an integer and set $M_x = \gcd(f(x), P)$. By definition of $g_{i,j}(x)$ and $h_i(x)$ we know that x is a root of both families of these polynomials modulo M_x^a. As in Theorem 2.1 our goal is to find an integer linear combination $w(x)$ of these polynomials such that $\|w(xB)\|$ is small (i.e. less than $M_x^a / \sqrt{\deg(w)}$). We form a lattice L spanned by the coefficient vectors of $g_{i,j}(xB)$ and $h_i(xB)$ and look for a short vector in L using LLL. We view the resulting vector as the coefficients vector of a polynomial $w(xB)$. The dimension of L is $d = a\delta + a'$. As in Theorem 2.1 (Equation (1)), the polynomial

$w(x)$ will have as roots all integers $|x| < B$ for which:

$$M_x > \det(L)^{1/ad} \cdot \gamma(d)^{1/a} \qquad \text{where} \qquad \gamma(d) = 2^{d/2}\sqrt{d}$$

Since the given basis of L forms a triangular matrix its determinant is given by the product of the entries on the diagonal. Hence,

$$\det(L) = P^{\delta a(a+1)/2}\, B^{d(d-1)/2}$$

Plugging $\det(L)$ into the above bound shows that $w(x)$ will have as roots all integers $|x| < B$ for which:

$$M_x > P^{\delta(a+1)/2d}\, B^{(d-1)/2a} \cdot \gamma(d)^{1/a}$$

We are free to choose $a \in \{1, \ldots, d/\delta\}$ so as to minimize the right hand side. The optimal value is:

$$a_{opt} = \frac{d}{\delta} \cdot \sqrt{\frac{\delta \log B}{\log P} \cdot \left(1 - \frac{1}{d}\right) + \frac{2\delta \log \gamma(d)}{d \log P}}$$

Rounding a_{opt} to the closest integer and using the fact that $\frac{2\log\gamma(d)}{d} < 2$ we get:

$$M_x > P^{\epsilon} \quad \text{where} \quad \epsilon = \sqrt{\frac{\delta \log 4B}{\log P}} + \frac{5\delta}{4d}$$

Hence, the polynomial $w(x) \in \mathbb{Z}[x]$ will have as roots all integers $|x| < B$ for which $\gcd(f(x), P) > P^{\epsilon}$ as required. ■

A possible application to the quadratic sieve: Suppose we are trying to factor a 10,000 bit integer N. This is currently beyond our reach, but it serves to illustrate how the algorithm works. Take $s = 1000$. Then $\lfloor \log S \rfloor = 1428$. As in the quadratic sieve, we build a polynomial

$$f_a(x) = (x + \lfloor\sqrt{aN}\rfloor)^2 - aN$$

for some small a. Observe that for relatively small values of x (i.e. $|x| < 2^{500}$) we have that $|f(x)| < 2^{5600}$, much smaller than N. Using Theorem 4.1 (with $d = 50$) we can find all $x \in [-2^{500}, 2^{500}]$ such that $\gcd(f(x), S) > 2^{1268}$. Hence, we are able to find all $f_a(x)$ that have a large smooth part, if they exist. We were able to scan an interval of width 2^{500}; far more than is possible using sieving.

There are two problems in using this technique within the quadratic sieve. First, random intervals of the length for which Theorem 4.1 applies will not contain sufficiently smooth integers. In other words, the current bounds on generalized CRT list decoding are insufficient for improving the basic quadratic sieve. Second, current factoring algorithms need a fully smooth integer – just having a large smooth factor is insufficient. Nevertheless, we hope future factoring algorithms will be able to take advantage of this ability to scan large intervals for smooth numbers.

5. NOISY CRT AND NOISY INTERPOLATION

We conclude by considering two analogous problems that come up in the context of program correction [1] and oblivious transfer [20]. These problems serve to further illustrate the algorithmic similarity between CRT and polynomial interpolation.

– **Problem 1.** Input: n pairs $(x_i, S_i)_{i=1,\dots,n}$ where $x_i \in \mathbb{Z}$ and $S_i \subseteq \mathbb{Z}$.
  All x_i's are distinct.
  Problem: find all $f(x) \in \mathbb{Z}[x]$ of degree at most d so that $f(x_i) \in S_i$ for all i.
– **Problem 2.** Input: n pairs $(p_i, S_i)_{i=1,\dots,n}$ where $p_i \in \mathbb{Z}$ and $S_i \subseteq \mathbb{Z}$.
  The p_i's are relatively prime.
  Problem: find all $\alpha \in \mathbb{Z}$ with $|\alpha| < 2^d$ so that $\alpha \bmod p_i \in S_i$ for all i.

In both problems we let $m = max|S_i|$.

Problem 2 comes up in the context of counting points on elliptic curves over finite fields [3, Ch. VII.9]. Problem 1 is closely related to a problem studied by Ar et al. [1].

Problem 1 can be solved using Guruswami and Sudan's algorithm [12] as long as $m < n/d$. We show that a similar bound holds for Problem 2. In fact, we show that Problem 2 can be solved directly using Coppersmith's algorithm (Theorem 1.2) as long as $m < (n/d)\log p_1$.

THEOREM 5.1. *Let $(p_i, S_i)_{i=1,\dots,n}$ be an instance of Problem 2. Whenever $m < (n/d)\log p_1$ there is a polynomial time algorithm that outputs all the required $\alpha \in \mathbb{Z}$.*

Proof. Define the polynomials $g_i(x) = \sum_{j=0}^{m} g_{i,j} x^j \in \mathbb{Z}[x]$ by:

$$g_i(x) = \prod_{a \in S_i} (x - a) \qquad \text{for } i = 1, \dots, n$$

The degree of these polynomials is at most m. We know that $g_i(\alpha) = 0 \bmod p_i$ for all i. Next, we build a polynomial $h(x) \in \mathbb{Z}[x]$ of degree m satisfying $h \equiv g_i \bmod p_i$ for all $i = 1, \dots, n$. To construct this $h(x) = \sum_{j=0}^{m} h_j x^j$ we construct its coefficients one by one. To construct the coefficient $h_j \in \mathbb{Z}$ simply use the CRT to find an integer h_j satisfying

$$h_j = g_{i,j} \bmod p_i \qquad \text{for } i = 1, \dots, n$$

The resulting polynomial $h(x)$ will satisfy $h = g_i \bmod p_i$ for all i. Let $P = \prod_{i=1}^{n} p_i$. Since $h(\alpha) = 0 \bmod p_i$ for all $i = 1, \dots, n$ we know that the polynomial $h(x)$ satisfies:

$$h(\alpha) = 0 \pmod{P} \qquad \text{and} \qquad |\alpha| < 2^d$$

The degree of h is m and it is a monic polynomial. Using Coppersmith's theorem (Theorem 1.2) we can find all roots of $h(x)$ that are less than $P^{1/m}$. Hence, when $P^{1/m}$ is greater than 2^d Coppersmith's algorithm will find all solutions to problem 2. Therefore, since $P > (p_1)^n$ we can solve problem 2 as long as $m < (n/d)\log p_1$. ■

We note that Bleichenbacher and Nguyen [4] recently discovered (heuristic) polynomial time algorithms for problems 1 and 2 that achieve better bounds than above.

Their algorithms give provable bounds for random instances of the problems. We also note that an earlier version of this paper [5] proposed an incorrect algrorithm for Problem 1. Fixing the algorithm reduces it to the Guruswami-Sudan method [12] as applied to Problem 1. We therefore omit the proposal from this writeup.

6. CONCLUSIONS

We gave a new algorithm for CRT list decoding. The algorithm solves the problem whenever

$$e < n - \sqrt{kn\frac{\log p_n}{\log p_1}}$$

When p_1 and p_n are close, e.g. $\log p_n = \log p_1 + o(1)$ the algorithm corrects as many errors as Guruswami-Sudan's algorithm for Reed-Solomon list decoding. Thus, the algorithm reduces the existing gap between these analogous problems. We note that recently Guruswami, Sahai, and Sudan [13] were able to eliminate the $\frac{\log p_n}{\log p_1}$ factor and obtain an algorithm that works whenever $e < n - \sqrt{k(n+\epsilon)}$ for arbitrarily small $\epsilon > 0$.

In the second half of the paper we showed that CRT list decoding can be used to find all integers in a given interval that have a large strongly smooth factor. The algorithm's running time is polynomial in both $\log |I|$ and the smoothness bound s. The algorithm also gives bounds on the number of such integers in a given interval. Finally we presented an algorithm for a generalized CRT list decoding problem that is motivated by the quadratic sieve.

ACKNOWLEDGMENTS

The author thanks Phong Nguyen and Dan Bernstein for several helpful discussions. This work was done while the author was supported by NSF contract #CCR-9732754 and a grant from the Packard foundation.

REFERENCES

1. S. Ar, R. Lipton, R. Rubinfeld, M. Sudan, "Reconstructing algebraic functions from mixed data", SIAM J. of Computing, vol. 28, No. 2, pp. 488-511, 1999.
2. E. Berlekamp, "Algebraic coding theory", McGraw Hill, 1968.
3. I. Blake, G. Seroussi, N. Smart, "Elliptic curves in cryptography", Cambridge University Press, 1999.
4. D. Bleichenbacher, P. Nguyen, "Noisy Polynomial Interpolation and Noisy Chinese Remaindering", Proc. of Eurocrypt '2000.
5. D. Boneh, "Finding smooth integers using CRT decoding", Proc. of STOC '2000, pp. 265–272, 2000.
6. D. Boneh, G. Durfee, "Cryptanalysis of RSA with private key $d < N^{0.292}$", Proc. of Eurocrypt '98, pp. 1–11, 1998.
7. D. Boneh, G. Durfee, N. Howgrave-Graham, "Factoring $N = p^r q$ for large r", Proc. of Crypto '99, pp. 326–337, 1999.
8. H. Cohen, "A course in computational algebraic number theory", Springer-Verlag, 1993.
9. D. Coppersmith, "Small solutions to polynomial equations, and low exponent RSA vulnerabilities", J. of Cryptology, Vol. 10, pp. 233–260, 1997.
10. O. Goldreich, D. Ron, M. Sudan, "Chinese remaindering with errors", Proc. of STOC '99, pp. 225–234, 1999.
Updated version available from *ECCC*.

11. O. Goldreich, R. Rubinfeld, M. Sudan, "Learning polynomials with queries: the highly noisy case", in proc. FOCS 95, pp. 294–303, 1995.
12. V. Guruswami, M. Sudan, "Improved decoding for Reed-Solomon and algebraic geometric codes", IEEE Tran. on Info. Theory, vol. 45, no. 6, pp 1757–1767, 1999.
13. V. Guruswami, A. Sahai, M. Sudan, "Soft decision decoding of chinese remainder codes", in proc. FOCS 2000.
14. N. Howgrave-Graham, "Finding small roots of univariate modular equations revisited", Proc. of Cryptography and Coding, LNCS 1355, Springer-Verlag, 1997, pp. 131–142.
15. A. Lenstra, H.W. Lenstra Jr., "Algorithms in Number Theory", in Handbook of Theoretical Computer Science (Volume A: Algorithms and Complexity), ch. 12, pp. 673–715, 1990.
16. A. Lenstra, H.W. Lenstra Jr., "The development of the number field sieve", Lecture Notes in Mathematics, Vol. 1554, Springer-Verlag, 1994.
17. A. Lenstra, H.W. Lenstra Jr., and L. Lovasz, "Factoring polynomial with rational coefficients", *Mathematiche Annalen*, 261:515–534, 1982.
18. L. Lovasz, "An algorithmic theory of numbers, graphs and convexity", SIAM lecture series, Vol. 50, 1986.
19. F. MacWilliams, N. Slaon, "The theory of error correcting codes", North-Holland, 1981.
20. M. Naor, B. Pinkas, "Oblivious Transfer and Polynomial Evaluation", in proc. STOC 1999, pp. 245–254, 1999.
21. M. Soderstrand, W. Jenkins, G. Jullien, F. Taylor, "Residue number system arithmetic: modern applications in digital signal processing", IEEE Press, 1986.
22. M. Sudan, "Decoding of Reed-Solomon codes beyond the error-correction bound", J. of complexity, vol. 13, no. 1, pp. 180–193, 1997.

APPENDIX A

We prove a simple fact used in the proof of Corollary 2.2.

LEMMA 6.1. *Let $0 < x_1 < x_2 < \ldots < x_n$ be n real numbers. Then for any $0 < e, k < n$, whenever $n - e \geq \sqrt{kn\frac{x_n}{x_1}}$ we have that*

$$\sum_{i=1}^{n-e} x_i \geq \sqrt{\sum_{i=1}^{k} x_i \sum_{i=1}^{n} x_i}$$

Proof. Observe that

$$\left(\sum_{i=1}^{n-e} x_i\right)^2 \geq [(n-e)x_1] \cdot \sum_{i=1}^{n-e} x_i = (n-e)^2 x_1 \cdot \frac{1}{n-e} \sum_{i=1}^{n-e} x_i$$

Since $n - e \geq k$ and the x_i's are in increasing order we know that $\frac{1}{n-e}\sum_{i=1}^{n-e} x_i \geq \frac{1}{k}\sum_{i=1}^{k} x_i$. Hence,

$$\left(\sum_{i=1}^{n-e} x_i\right)^2 \geq (n-e)^2 x_1 \cdot \frac{1}{k} \sum_{i=1}^{k} x_i$$

Using the fact that $n - e \geq \sqrt{kn\frac{x_n}{x_1}}$ we obtain

$$\left(\sum_{i=1}^{n-e} x_i\right)^2 \geq kn\frac{x_n}{x_1} x_1 \cdot \frac{1}{k} \sum_{i=1}^{k} x_i = nx_n \cdot \sum_{i=1}^{k} x_i \geq \sum_{i=1}^{n} x_i \sum_{i=1}^{k} x_i$$

The lemma now follows. ■