# “Soft-decision” Decoding of Chinese Remainder Codes

Venkatesan Guruswami* Amit Sahai *† Madhu Sudan *‡

**Abstract**

Given $n$ relatively prime integers $p_1 < \cdots < p_n$ and an integer $k < n$, the Chinese Remainder Code, $\mathrm{CRT}_{p_1,\ldots,p_n;k}$, has as its message space $\mathcal{M} = \{0, \ldots, \prod_{i=1}^{k} p_i - 1\}$, and encodes a message $m \in \mathcal{M}$ as the vector $\langle m_1, \ldots, m_n \rangle$, where $m_i = m (\bmod\, p_i)$. The soft-decision decoding problem for the Chinese remainder code is given as input a vector of residues $\vec{r} = \langle r_1, \ldots, r_n \rangle$, a vector of weights $\langle w_1, \ldots, w_n \rangle$, and an agreement parameter $t$. The goal is to find all messages $m \in \mathcal{M}$ such that the weighted agreement between the encoding of $m$ and $\vec{r}$ (i.e., $\sum_i w_i$ summed over all $i$ such that $r_i = m (\bmod\, p_i)$) is at least $t$. Here we give a new algorithm for solving the soft-decision problem for the CRT code that works provided the agreement parameter $t$ is sufficiently large. We derive our algorithm by digging deeper into the algebra underlying the error-correcting algorithms and unveiling an “ideal”-theoretic view of decoding.

When all weights are equal to 1, we obtain the more commonly studied “list decoding” problem. List decoding algorithms for the Chinese Remainder Code were given recently by Goldreich, Ron, and Sudan [5], and improved by Boneh [1]. Their algorithms work for $t \geq \sqrt{2kn \log p_n / \log p_1}$ and $t \geq \sqrt{kn \log p_n / \log p_1}$, respectively. We improve upon the algorithms above by using our soft-decision decoding algorithm with a non-trivial choice of weights, and solve the list decoding problem provided $t \geq \sqrt{k(n + \varepsilon)}$, for arbitrarily small $\varepsilon > 0$.

## 1 Introduction

Given $n$ relatively prime integers $p_1 < \cdots < p_n$ and an integer $k < n$, the Chinese Remainder Code, $\mathrm{CRT}_{p_1,\ldots,p_n;k}$, has as its message space $\mathcal{M} = \{0, \ldots, \prod_{i=1}^{k} p_i - 1\}$, and encodes a message $m \in \mathcal{M}$ as the vector $\langle m_1, \ldots, m_n \rangle$, where $m_i = m (\bmod\, p_i)$. The Chinese Remainder Code (henceforth, CRT code), also referred to as the Redundant Residue Number System code, seems to have been studied for several years now in the literature in coding theory (see [15, 9], and the

*MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139. Email: {venkat,amits,madhu}@theory.lcs.mit.edu.

†Supported in part by a DOD NDSEG Fellowship.

‡Supported in part by an MIT-NEC Research Initiation Award, a Sloan Foundation Fellowship and NSF Career Award CCR-9875511.

references there in), and its redundancy property has been exploited often in theoretical computer science as well. Mandelbaum gave a decoding algorithm for this code, correcting $\frac{n-k}{2}$ errors.[1]

Recently, Goldreich, Ron, and Sudan [5] gave a "list decoding" algorithm for this code. Formally, the list decoding problem has as its inputs a vector $\langle p_1, \ldots, p_n \rangle$, an integer $k$ (specifying the CRT code), a vector $\langle r_1, \ldots, r_n \rangle$ and an agreement parameter $t$. The goal is to find a list of all messages $m \in \mathcal{M}$ such that $r_i = m(\bmod p_i)$ for at least $t$ choices of $i \in \{1, \ldots, n\}$. The notion of list decoding was proposed independently by Elias [3] and Wozencraft [17] as a relaxation to the usual notion of recovery from errors (which requires the output to be a single message). Informally, a list decoding algorithm offers a method of recovery from $n-t$ errors. For the case of the CRT code, the algorithm of [5] solved the list decoding problem in polynomial time provided $t \geq \sqrt{2kn\frac{\log p_n}{\log p_1}}$. If $p_n = O(p_1)$, and $k = o(n)$, then $t$ can be growing as $o(n)$ and this is far better than the results achievable via standard (not list) decoding. More recently, Boneh [1] reduced the requirement on $t$ by a factor of $\sqrt{2}$ to be able to correct from $\sqrt{kn\frac{\log p_n}{\log p_1}}$ agreements. Numerous applications are also now known for the CRT list decoding problem. Goldreich et al. [5] describe an application to computation of the permanent on random instances, Håstad and Näslund [8] use it in constructing hardcore predicates from some (specific) one-way functions, and Boneh [1] shows consequences to the task of finding smooth numbers in short intervals. While for all the applications, the original result of [5] would have sufficed (at least to derive qualitatively interesting results), they nevertheless motivate a closer look at the decoding algorithms (and if this yields an improvement in performance, so much the better).

One weakness common to all the known algorithmic results on CRT decoding is their poor(er) performance if the primes are varying significantly in size. This can cause the algorithm of Mandelbaum [11] to take exponential time, while it degrades the number of errors that the algorithms of Goldreich et al. [5], or Boneh [1] can correct. This weakness, in turn, highlights an eccentricity of the CRT code: Its alphabet size is not uniform, and so the "contribution" of an error is not independent of its location. Viewed differently, if the residue of a message $m$ is known correctly modulo a small prime, then this provides less information than if the residue of $m$ is known correctly modulo a large prime. The first coordinate of the code provides only $\log p_1$ bits of information about the message, while the last coordinate provides $\log p_n$ bits of information. However when we treat the code as a combinatorial object, all coordinates are declared to be equally important. The distortion in translating between the two measures of "importance" of the coordinates leads to a degradation in performance of the code and this explains the common occurrence of the quantity $\log p_n / \log p_1$.

At first glance, this loss in performance seems inevitable. After all we are distorting the natural weighting of the code and so the algorithmic results should suffer. However, a closer look reveals that this distortion has already been accounted for when estimating the distance of the code. It then follows that the code does have distance greater than $n-k$ in the uniform weighting; and thus it should be possible to correct $(n-k)/2$ errors unambiguously. Similarly, some standard results on the combinatorics of list decoding imply that the output size of the list decoding problem is bounded by a polynomial in $n$ if $t > \sqrt{nk}$. However, the algebra of known decoding algorithms defer to the natural weighting of the alphabets of the CRT code. To overcome this limitation, one

[1]Mandelbaum [11] does not give a precise bound on the running time of the algorithm. It is pointed out in [5] that the algorithm can have exponential running time for certain values of the $p_i$'s. It seems easier to modify the algorithm so as to correctly only $\frac{(n-k)\log p_1}{\log p_1 + \log p_n}$ errors in polynomial (in $n, \log p_n$) time (cf. [5]).

needs algorithms to decode the CRT code under the uniform weighting, or more generally, some arbitrary "user-specified" weighting, of the coordinates of the code.

**Our Results.** We first consider the combinatorial implications of the question of "reweighting" the coordinates of a code in a general setting (and not just the CRT code). Say we have a code $\mathcal{C}$ of $n$-letter strings, with its natural weight vector $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_n \rangle$, where $\alpha_i$ is a non-negative real representing the "natural importance" of the $i$-th coordinate of the code. (For the CRT code $\alpha_i = \log p_i$.) Say the code $\mathcal{C}$ has distance $D_{\vec{\alpha}}$ under this weighting (i.e. for any two codewords $x, y \in \mathcal{C}$, $\sum_{i|x_i \neq y_i} \alpha_i \geq D_{\vec{\alpha}}$). Now suppose we wish to impose our own weighting $\vec{\beta} = \langle \beta_1, \ldots, \beta_n \rangle$ on the alphabets (typically, our weighting would be the uniform one), and wish to study the code $\mathcal{C}$ under this weighting. We first prove some combinatorial results giving some lower bound on $t$, such that if the weight of agreement under the $\vec{\beta}$-weighting is at least $t$, then the size of the output of the list decoder is bounded by a polynomial in $n$. (See Theorem 1 and its Corollaries.)

Next we consider the task of recovering the list of all such codewords in polynomial time, for the CRT code. In general, there are few algorithms in the literature on coding theory where the natural weighting of the code (usually the uniform one) can be overcome by a "user-imposed" weighting; and this is exactly what we wish to do in this case. Two known exceptions to this are the Generalized Minimum Distance (GMD) decoding algorithm of Forney [4], and the weighted version of the Reed-Solomon decoding algorithm of Guruswami and Sudan [6]. These two algorithms form the starting points for our algorithmic results.

Our first algorithmic result (Theorem 2) applies the GMD algorithm of Forney [4] to the task of decoding the CRT code under the uniform weighting. We show how to combine this result with the results of Mandelbaum [11, 12] and Goldreich et al. [5] to obtain the *first* polynomial time algorithm which decodes the CRT code up to half the minimum distance of the code (i.e., recovering from up to $(n-k)/2$ errors). We stress that no polynomial time algorithm was known for this task prior to our result, since the run time of Mandelbaum's algorithm [11, 12] to correct up to $(n-k)/2$ errors, was not always polynomial in $n, \log p_n$ (see [5] for a discussion). Our algorithm can actually recover from a number of errors which is less than *half the weighted minimum distance* for *any* set of positive weights imposed on the codeword positions. Technically, this part of the paper is simple – the main contribution of this part may be viewed as highlighting the role of GMD decoding in the task of decoding the CRT code.

Our second algorithmic result extends the weighted list decoding algorithm of Guruswami and Sudan [6] to the case of the CRT code. As a consequence we show how to solve the weighted list decoding problem for an arbitrary choice of the $\vec{\beta}$ vector, as long as the $\vec{\beta}$-weighted error matches the combinatorial bound of Theorem 1. This result is shown in Corollary 4 to Theorem 4. We then show how to choose the weight vector $\vec{\beta}$ (and this part turns out to be a non-trivial guess) so that we get a solution to the uniform list decoding problem for the CRT code, for $t \geq \sqrt{k(n+\varepsilon)}$, for a tolerance parameter $\varepsilon > 0$ as small as we seek. In fact, we can efficiently list decode as long as $t \geq \min\left\{\sqrt{k(n+\varepsilon)}, \sqrt{\left(\sum_{i=1}^{k} \log p_i\right)\left(\sum_{i=1}^{n} \frac{1}{\log p_i} + \varepsilon\right)}\right\}$.

Theorem 3 is the technical centerpoint of this paper. It is proven by creating an "ideal"-lic view of error-correcting codes and the decoding problem. This view captures all known algebraic codes, including Reed-Solomon codes and the more general algebraic-geometric codes, as well as number-theoretic codes such as the CRT code. Further, we present a decoding algorithm in the same framework that unifies the algebra of most of the known list decoding algorithms including those in [16, 14, 6, 5, 1], and most important for us, the weighted list decoding algorithm of [6]. The resulting abstraction reduces the decoding problem to a number of "elementary" algorithmic problems on the underlying ideals. In the case of the CRT code, these problems turn out to be well-solved problems on integer lattices, such as the problems of computing the sum and the intersection of given lattices, or finding short vectors in them, and thereby solves the weighted list decoding problem for the CRT code. The unified algebraic framework emerging from this study may be of independent interest.

**Organization.** We begin by describing combinatorial bounds on the "radius" up to which we are guaranteed to have a small number of codewords for a general code which has varying weights (and varying alphabet sizes) on its various coordinates. In Section 3 we describe and analyze a "soft-decision" algorithm for decoding CRT codes, and also prove our main algorithmic result (Theorem 3). Our algorithm is motivated and founded upon an ideal-theoretic view of existing decoding algorithms [6, 5, 1] for "redundant-residue codes" like the Reed-Solomon and Chinese Remainder codes, which we ferret out and describe as an Appendix (Appendix A). We then get specific results for interesting weightings of the coordinates by non-trivial choice of weights in the main algorithm.

# 2 Combinatorial Bounds

**Theorem 1** *Let $\mathcal{C}$ be a code of length $n$ with the $i$th symbol coming from an alphabet of size $q_i$. Let the distance $D_\alpha$ of the code be measured according to a weighting vector $\vec{\alpha}$ i.e., for any two distinct codewords $c_1, c_2$, $\sum_{i:c_{1i}\neq c_{2i}} \alpha_i \geq D_\alpha$ (assume each $\alpha_i \geq 1$ without loss of generality). For a weighting vector $\vec{\beta}$ and a received word $y$, define the ball $B_{\vec{\beta}}(y, W)$ to consist of all strings $z$ (in the space $[q_1] \times [q_2] \times \cdots \times [q_n]$) such that $\sum_{i:y_i \neq z_i} \beta_i \leq W$. Then, for all $y$, the ball $B_{\vec{\beta}}(y, E_\beta)$ has at most $\frac{D_\alpha}{\sum_i \alpha_i - D_\alpha}\Big(\sum_i \frac{\beta_i^2}{\alpha_i}\Big)\varepsilon^{-1}$ codewords from $\mathcal{C}$ provided:*

$$E_\beta \;\leq\; \sum_i (1-\frac{1}{q_i})\beta_i - \Big[\Big(\sum_i (1-\frac{1}{q_i})\frac{\beta_i^2}{\alpha_i} + \varepsilon\Big) \Big(\big(\sum_i (1-\frac{1}{q_i})\alpha_i\big) - D_\alpha\Big)\Big]^{1/2}.$$

*(All sums are for $1 \leq i \leq n$.)* □

We prove this bound by generalizing the method of [7], which was used to establish a similar bound in the special case where $\vec{\alpha}$ is the all $1^{\text{s}}$ vector, and $\vec{\beta} \in \{0,1\}^n$, and all $q_i$'s are equal. The details of the proof are quite technical, and may be found in the full version of the paper.

When $\vec{\beta}$ equals $\vec{\alpha}$ or is the all 1s vector, we can get the following Corollaries:

**Corollary 1** *When $\vec{\beta} = \vec{\alpha}$ in the above Theorem, then there are at most a polynomial (in $n, \sum_i \alpha_i, \varepsilon^{-1}$) many codewords in any ball $B_{\vec{\beta}}(y, E_\alpha)$ provided $E_{\vec{\alpha}} \leq \alpha_{\text{tot}} - \sqrt{(\alpha_{\text{tot}} + \varepsilon)(\alpha_{\text{tot}} - D_\alpha)}$, where $\alpha_{\text{tot}} = \sum_{i=1}^n \alpha_i$.*

**Corollary 2** *If $\beta_i = 1$ for all $i$, i.e., the distance of the received word from codewords is measured using the Hamming distance, then there are at most a polynomial in $n, 1/\varepsilon$ many codewords in a Hamming ball of radius $E$ provided $E \le n - \sqrt{(\alpha_{\mathrm{tot}} - D_\alpha)\Big(\sum_{i=1}^{n} \frac{1}{\alpha_i} + \varepsilon\Big)}$.*

We will see that for the case of CRT codes, we can essentially match the bounds of Theorem (1) (in the limit of large alphabet sizes) and Corollaries 1 and 2 algorithmically.

# 3 Algorithms for decoding CRT codes

In this section, we discuss efficient decoding algorithms for the CRT code. As stated above, we consider a sequence $p_1 < p_2 < \ldots < p_n$ of relatively prime integers and an integer $k < n$. Let $K = \prod_{i=1}^{k} p_i$; $N = \prod_{i=1}^{n} p_i$. We associate to each integer $m \in \{0, 1, \ldots, K-1\}$ the sequence $(m_1, m_2, \ldots, m_n)$, where $m_i = m \bmod p_i$. We will abuse notation and refer to both this sequence and $m$ as a *codeword*. We consider a *received word* to be a sequence $\langle r_1, r_2, \ldots, r_n\rangle$ of integers with $0 \le r_i < p_i$ for each $i$ from 1 to $n$. By the Chinese Remainder Theorem, each such sequence corresponds to a unique non-negative integer $r < N$. For a given sequence of weights $\vec{w} = \langle w_1, \ldots, w_n\rangle$, we say the *$\vec{w}$-weighted agreement* (or simple *weighted agreement* when the weighting we are referring to is clear) between a codeword $m < K$ and a received word $r < N$ is $\sum_i a_i w_i$, where $a_i = 1$ if $m_i = r_i$, and $a_i = 0$ otherwise.

In this section, we present two efficient decoding algorithms. For any sequence of positive weights $\vec{\beta}$, the first one efficiently (in near-quadratic time) recovers the *unique* codeword $m < K$ with highest $\vec{\beta}$-weighted Hamming agreement with a received word $r$, as long as there is a codeword whose $\vec{\beta}$-weighted Hamming distance from $r$ is less than half the $\vec{\beta}$-weighted minimum distance of the code. codeword modulo at least $(n+k)/2$ positions. This is accomplished by adapting the method of Forney, introduced for Reed-Solomon codes in [4], to CRT codes. Note that in particular this gives the first efficient algorithm to correct from $(n-k)/2$ errors (i.e., decode up to half the minimum distance) for the CRT code.

In the second (which is our main) decoding algorithm, the goal is to efficiently find a list of *all* codewords $m < K$ such that $m$ and the received word $r$ have sufficient weighted agreement. In particular, we are able to give an efficient list decoding algorithm which outputs all codewords $m < K$ which agree with $r$ modulo at least $\sqrt{k(n+\varepsilon)}$ positions (for any $\varepsilon$, with the running time of the algorithm depending polynomially in $1/\varepsilon$).

## 3.1 GMD decoding for CRT codes

For integers $k, n$ and relatively prime integers $p_1 < p_2 < \cdots < p_n$, and any integer $j$, $1 \le j \le n$, Goldreich, Ron, and Sudan [5] gave a near-linear time algorithm to compute the unique integer $m$, if any, that satisfies

$$\sum_{i=1}^{j} a_i \log p_i > \frac{1}{2}\left(\sum_{i=1}^{j} \log p_i + \sum_{i=1}^{k} \log p_i\right) \tag{1}$$

where $a_i$ is defined in the usual way: $a_i = 1$ if $m = r_i (\bmod p_i)$ and $a_i = 0$ otherwise. Note that the above algorithm decodes up to half the minimum $\vec{w}$-weighted distance ($\log(N/K)$) for the "natural" weighting $w_i = \log p_i$ of the CRT code. Using this algorithm as the "basic algorithm" and running a GMD style algorithm similar to Forney [4], we are able to perform such a decoding for any choice of weights $\vec{\beta} = \langle \beta_1, \beta_2, \ldots, \beta_n \rangle$.

To prove this we show a more general result. Suppose we have an arbitrary code $\mathcal{C}$ of block-length $n$. We show how to use a decoding algorithm designed for *any* weighting $\vec{\alpha}$ to produce one that works for the *desired* weighting $\vec{\beta}$. Define $A_\alpha = \sum_{i=1}^n \alpha_i - D_\alpha$ where $D_\alpha$ is $\vec{\alpha}$-weighted distance of the code, so that $A_\alpha$ is the maximum $\vec{\alpha}$-weighted agreement between two distinct codewords of $\mathcal{C}$; $A_\beta$ is defined similarly. We are now ready to state and prove the main result of this section:

**Theorem 2** *Let $\vec{\alpha}, \vec{\beta} \in \mathbb{R}_+^n$ be positive real vectors such that $\frac{\beta_1}{\alpha_1} \geq \frac{\beta_2}{\alpha_2} \geq \cdots \geq \frac{\beta_n}{\alpha_n}$. Suppose we have a polynomial time algorithm $\mathsf{Alg}_\alpha$ that given a received word $\vec{r} = \langle r_1, \ldots, r_n \rangle$ and an index $j$ ($1 \leq j \leq n$), can find the unique codeword $C$, if any, whose $\vec{\alpha}$-weighted agreement with $\vec{r}$ in the first $j$ codeword positions is more than $\frac{1}{2}(\sum_{i=1}^j \alpha_i + A_\alpha)$. Then, for any vector of positive reals $\vec{\beta} = \langle \beta_1, \ldots, \beta_n \rangle$, there is a polynomial time algorithm $\mathsf{Alg}_\beta$ that given a received word $\langle r_1, \ldots, r_n \rangle$, outputs the unique codeword, if any, whose $\vec{\beta}$-weighted agreement with $\vec{r}$ is at least $\frac{1}{2}\Big(\sum_{i=1}^n \beta_i + A_\beta + \beta_{\max}\Big)$, and moreover the run-time of $\mathsf{Alg}_\beta$ is at most $O(n)$ times that of $\mathsf{Alg}_\alpha$.*

**Corollary 3** *For the CRT code with parameters $(n, k; p_1, p_2, \ldots, p_n)$, for any received word $\vec{r} = (r_1, r_2, \ldots, r_n)$, there is a polynomial time (in fact near-quadratic time) algorithm to find the unique codeword $m = (m_1, m_2, \ldots, m_n)$, if any, that agrees with $\vec{r}$ in at least $\frac{n+k}{2}$ positions.*

**Proof:** By Equation (1) we have a near-linear time decoding algorithm for the weighting $\alpha_i = \log p_i$ and $A_\alpha = \log K$ (where $K = p_1 p_2 \cdots p_k$). By Theorem 2 applied to $\vec{\beta}$ being the all-ones vector, we have $A_\beta = k$ and thus we can find the unique codeword $m$ that agrees with $\vec{r}$ in at least $(n+k+1)/2$ places. For any constant $c$, we can also correct $c$ additional errors by simply erasing $c$ symbols for all $\binom{n}{c}$ possible choices of $c$ positions and then running the above decoding algorithm. In particular, this implies that we can find the unique codeword with agreement at least $(n+k)/2$ with $\vec{r}$ in polynomial time. $\square$

**Proof of Theorem 2:** Recall that the codeword positions $i$ are ordered so that $\frac{\beta_1}{\alpha_1} \geq \frac{\beta_2}{\alpha_2} \geq \cdots \geq \frac{\beta_n}{\alpha_n}$. Define

$$\tilde{A}_\beta \stackrel{\text{def}}{=} \max_{\substack{\mathbf{x} \in [0,1]^n \\ \sum \alpha_i x_i \leq A_\alpha}} \left\{ \sum_{i=1}^n \beta_i x_i \right\} . \tag{2}$$

Note that under the condition $\mathbf{x} \in \{0,1\}^n$, the above would just define $A_\beta$; we relax the condition to $\mathbf{x} \in [0,1]^n$ in the above to define $\tilde{A}_\beta$. Clearly $A_\beta \leq \tilde{A}_\beta < A_\beta + \beta_{\max}$. We will present an algorithm to find the unique codeword $C$, if any, that satisfies

$$\sum_{i=1}^n a_i \beta_i > \frac{1}{2}\Big(\sum_{i=1}^n \beta_i + \tilde{A}_\beta\Big) \tag{3}$$

(where $a_i = 1$ if $C_i = r_i$ and 0 otherwise), and this will imply the claimed result (since $\tilde{A}_\beta < A_\beta + \beta_{\max}$). We now assume such a $C$ exists, for, otherwise, there is nothing to prove.

The algorithm $\mathsf{Alg}_\beta$ will simply run $\mathsf{Alg}_\alpha$ for all values of $j$, $1 \le j \le n$, and pick the closest codeword among the (at most $n$) codewords which the runs of $\mathsf{Alg}_\alpha$ returns. If this algorithm fails to find the codeword $C$ that satisfies Condition (3), then we must have, by the hypothesis of the Theorem, for every $j$, $1 \le j \le n$,

$$2\sum_{i=1}^{j} a_i\alpha_i \le \sum_{i=1}^{j} \alpha_i + A_\alpha \ . \tag{4}$$

Let $\tilde{\mathbf{x}} = \langle 1\ 1\ \cdots\ 1\ \varepsilon\ 0\ \cdots\ 0\rangle$ be a vector such that $\sum_{i=1}^{n} \alpha_i\tilde{x}_i = A_\alpha$ (here $0 \le \varepsilon < 1$). Denote by $\ell$ the last position where $\tilde{x}_i = 1$ (i.e., $\tilde{x}_\ell = 1$ and $\tilde{x}_{\ell+1} = \varepsilon$). By our definition (2) $\tilde{A}_\beta \ge \sum \beta_i\tilde{x}_i$ (in fact by the ordering of the codeword positions it is easy to see that $\tilde{A}_\beta = \sum \beta_i\tilde{x}_i$ though we will not need this). Now for $j \ge \ell+1$, $A_\alpha = \sum_{i=1}^{n} \alpha_i\tilde{x}_i = \sum_{i=1}^{j} \alpha_i\tilde{x}_i$. Also, for $1 \le j \le \ell$, we have the obvious inequality $\sum_{i=1}^{j} a_i\alpha_i \le \sum_{i=1}^{j} \alpha_i = \sum_{i=1}^{j} \alpha_i\tilde{x}_i$. Combining these with Equation (4) we obtain the following uniform condition that holds for all $j$, $1 \le j \le n$:

$$2\sum_{i=1}^{j} a_i\alpha_i \le \sum_{i=1}^{j} \alpha_i + \sum_{i=1}^{j} \alpha_i\tilde{x}_i \ . \tag{5}$$

Multiplying the $j^{\text{th}}$ inequality above by the non-negative quantity $(\frac{\beta_j}{\alpha_j} - \frac{\beta_{j+1}}{\alpha_{j+1}})$ for $1 \le j \le n$ (define $\beta_{n+1} = 0$ and $\alpha_{n+1} = 1$), and adding the resulting inequalities, we get

$$2\sum_{i=1}^{n} a_i\beta_i \le \sum_{i=1}^{n} \beta_i + \sum_{i=1}^{n} \beta_i\tilde{x}_i \le \sum_{i=1}^{n} \beta_i + \tilde{A}_\beta \ ,$$

which contradicts Condition (3). Thus the codeword $C$ that satisfies (3), if any, will indeed be output by the algorithm $\mathsf{Alg}_\beta$. $\square$

## 3.2 The Weighted List Decoding Algorithm

Our goal in this section is to efficiently find a list of all codewords $m < K$ such that $m$ and the received word $r$ have sufficient weighted agreement. We note that a simple transformation makes it equivalent for us to find integers $m$ where $|m| \le K/2$, with sufficient agreement with a received word $(r_1, \ldots, r_n)$.

Our algorithm follows the ideal-based framework presented in Appendix A. Following [5], the basic idea will be to find an integer polynomial $c(x)$ (based on the received word $r$) with the property that all codewords that have sufficient weighted agreement with the received word are *roots* of the polynomial $c(x)$ over the integers. Then, by factoring $c(x)$ and extracting all factors of the form $(x - m)$ for $|m| \le K/2$ where $m$ has sufficient weighted agreement, we could recover all sufficiently similar codewords. We are able to construct such a polynomial by pursuing two objectives, which are in turn adaptations of the objectives of [16] in the context of Reed-Solomon codes:

1. To ensure that the polynomial $c(x)$ has the property that for any integer $m$ such that $|m| \le K/2$, if $m = r_i \bmod p_i$, then $c(m) \equiv 0 \bmod M_i$, for some sequence of moduli $M_i$. By the Chinese Remainder Theorem, this in turn implies that for any $m$ with $|m| \le K/2$, we have that $c(m) \equiv 0 \bmod \left(\prod_i M_i^{a_i}\right)$, where $a_i = 1$ if $m_i = r_i$, and $a_i = 0$ otherwise.

2. To ensure that the coefficients of $c(x) = \sum_{j=0}^{\ell} c_j x^j$ are sufficiently small. In particular, for some integer $G$, ensure that $|c_j| \le G/(K/2)^j$ for all $j$. This in turn implies that if $\ell$ is the degree of $c(x)$, for any $m$ with $m \le K/2$, we have that $|c(m)| < \sum_{j=0}^{\ell} |c_j|(K/2)^j \le (\ell+1)\cdot G$.

By combining Objectives 1 and 2, we see that for any integer $m$ such that $|m| \le K/2$ with sufficient agreement so that $\prod_i M_i^{a_i} > (\ell+1)\cdot G$, we have that $m$ is a root of $c(x)$, not only modulo some number, but over the integers too, as we desired. Note that the decoding condition is equivalent to $\sum_i a_i \log M_i > \log((\ell+1)\cdot G)$.

We show how to achieve these objectives for $M_i = p_i^{z_i}$, for arbitrary non-negative integer sequences $z_i$, yielding a weighted decoding condition similar to the one in [6] for Reed-Solomon codes.

## 3.3 The Main Theorem

We now state and prove our main algorithmic result.

**Theorem 3** *For a CRT code with the above parameters, given a received word* $r = (r_1, r_2, \ldots, r_n)$ *with* $0 \le r_i < p_i$*, and any non-negative integers* $\ell$ *and* $z_i$ *for* $1 \le i \le n$*, we can find in time polynomial in* $n, \log N, \ell$ *and* $\sum_i z_i$*, a list of all codewords* $m$ *that satisfy*

$$\sum_{i=1}^{n} a_i z_i \log p_i \;>\; \log(\ell+1) + \frac{\ell}{2}\log K + \\ + \frac{1}{\ell+1}\sum_{i=1}^{n}\binom{z_i+1}{2}\log p_i, \tag{6}$$

*where* $a_i = 1$ *if* $m_i = r_i$ *and* $a_i = 0$ *otherwise.*

**Proof:** In light of the preceding discussion, our basic objective will be, given some sequence of integers $z_i$, to find a polynomial $c(x)$ such that for all $i$, $1 \le i \le n$, the following Condition holds:

(*) For all integers $m$ such that $|m| \le K/2$, we have that $m \equiv r_i \bmod p_i$ implies $c(m) \equiv 0 \bmod p_i^{z_i}$.

For a fixed $i$, consider the $(z_i+1)$ polynomials $\left\{p_i^a (x-r_i)^{(z_i-a)}\right\}_{a=0}^{z_i}$. These certainly satisfy Condition (*). Let $I_i^{z_i}$ be the ideal in the polynomial ring $\mathbb{Z}[x]$ generated by these $(z_i+1)$ polynomials.[2] In other words, $I_i^{z_i}$ is the closure of this set of polynomials under addition, and multiplication by any polynomial. It is immediate that all polynomials in $I_i^{z_i}$ satisfy Condition (*). We now establish that there must be polynomials with small coefficients that lie in the intersection[3] ideal $I = \bigcap_{i=1}^{n} I_i^{z_i}$, and thus satisfy Condition (*) for all $i$:

[2] We note that the exponentiation notation actually makes sense here, see Appendix A.

[3] Note that this is also the product ideal, but we will refer to it as the intersection ideal in this text to retain intuition. See Appendix A for discussion.

**Lemma 1** *For any positive integers $\ell$ and $F$, if*

$$F > \left(\prod_{i=1}^{n} p_i^{\binom{z_i+1}{2}}\right)^{\frac{1}{\ell+1}} \cdot (K/2)^{\ell/2}$$

*then there exists some degree $\leq \ell$ integer non-zero polynomial $c(x) = \sum_{j=0}^{\ell} c_j x^j$ such that $|c_j| < F/(K/2)^j$ for all $j$ from $0$ to $\ell$, and $c \in I$.*

**Proof:** For any fixed $i$, we will count how many possible residues any integer polynomial $c(x)$ can have modulo $I_i^{z_i}$. Consider the following sequence of polynomials: Let $c^0(x)$ be the remainder when $c(x)$ is divided by $(x - r_i)^{z_i}$ (using the standard polynomial division algorithm). Because $(x - r_i)^{z_i}$ is monic, $c^0(x)$ has degree at most $z_i - 1$. Now add or subtract $p_i(x - r_i)^{z_i - 1}$ as many times as necessary from $c^0(x)$ in order to force the coefficient of $x^{z_i-1}$ to be non-negative and less than $p_i$; let the result be $c^1(x)$. Continue this process, obtaining $c^a(x)$ by adding or subtracting $p_i^a(x - r_i)^{z_i-a}$ as many times as necessary from $c^{a-1}(x)$ in order to force the coefficient of $x^{z_i-a}$ to be non-negative and less than $p_i^a$. We stop at $c^{z_i}(x)$, which we will call the *canonical residue* of $c(x)$ modulo $I_i^{z_i}$. The canonical residue is a degree $(z_i - 1)$ integer polynomial such that for all $a$ from $0$ to $z_i$, the coefficient of $x^a$ is non-negative and less than $p_i^{z_i-a}$. There are thus $\prod_{a=1}^{z_i} p_i^a = p_i^{\binom{z_i+1}{2}}$ such polynomials. We associate a canonical residue to $c(x)$ for each ideal $I_i^{z_i}$, yielding $\prod_{i=1}^{n} p_i^{\binom{z_i+1}{2}}$ possible sets of canonical residues.

Now we consider the space of degree $\leq \ell$ integer polynomials $c(x) = \sum_{j=0}^{\ell} c_j x^j$ such that $0 \leq c_j < F \cdot (K/2)^{-j}$ for all $j$ from $0$ to $\ell$. There are $F^{\ell+1} \cdot (K/2)^{-\binom{\ell+1}{2}}$ such polynomials. If $F^{\ell+1} \cdot (K/2)^{-\binom{\ell+1}{2}} > \prod_{i=1}^{n} p_i^{\binom{z_i+1}{2}}$, then there must either be a non-zero polynomial $c(x)$ all of whose canonical residues vanish, in which case $c(x) \in I$; or there must be two distinct such polynomials $c(x)$ and $c'(x)$ whose canonical residues are identical modulo every $I_i^{z_i}$. In this case, the polynomial $c(x) - c'(x)$ is in the intersection $I$ and satisfies the condition of the Lemma. $\Box$*(Lemma 1)*

To establish our algorithmic result, we need only show how to find such a polynomial $c(x)$ efficiently. Note that if we consider the intersection ideal $I$ restricted to polynomials of degree at most $\ell$, this can be seen as an integer lattice $L$ of dimension $(\ell + 1)$. Finding a suitable polynomial with small coefficients can therefore be seen exactly as finding a short vector in this lattice. This can be accomplished using lattice basis reduction algorithms such as LLL, provided we can construct a basis for this lattice. We stress that it is *not* necessary to explicitly write down the basis; all that we need is to be able to efficiently compute a basis. We now demonstrate how to do this.

Explicit bases for the individual lattices $L_i$ corresponding to the polynomials of degree at most $\ell$ in each $I_i^{z_i}$ are easily obtained by considering the generating polynomials for $I_i^{z_i}$ restricted to polynomials of degree at most $\ell$: Let $\tilde{z}_i = \min\{z_i, \ell\}$. The first $\tilde{z}_i + 1$ vectors in our basis correspond to the generating polynomials $\{p_i^{(z_i-a)}(x - r_i)^a : 0 \leq a \leq z_i\}$ from the ideal $I_i^{z_i}$. For example, corresponding to $p_i^{z_i-2}(x - r_i)^2$, we add the vector $(r_i^2 \cdot p_i^{z_i-2}, \quad -2r_i \cdot p_i^{z_i-2}, \quad p_i^{z_i-2}, \quad 0, \quad \ldots, \quad 0)$. If $\ell > z_i$, then we also add vectors corresponding to the polynomials $\{x^a \cdot (x - r_i)^{z_i}\}_{a=1}^{\ell-z_i}$. Let

$M^{(i)}$ be the $(\ell+1)$ by $(\ell+1)$ matrix whose rows are the vectors from this basis. We observe that the integer linear combinations of these vectors correspond exactly to the set of polynomials in the ideal $I_i^{z_i}$ of degree at most $\ell$:

**Lemma 2** *The space of polynomials corresponding to vectors in the lattice $L_i$ is exactly the ideal $I_i^{z_i}$ restricted to polynomials of degree at most $\ell$.*

**Proof:** By construction, the polynomials corresponding to integer linear combinations of the rows of $M^{(i)}$ are a subset of $I_i^{z_i}$ restricted to polynomials of degree at most $\ell$. Let $c(x) = \lambda^1(x) \cdot p_i^{z_i} + \lambda^2(x) \cdot p_i^{(z_i-1)}(x-r_i) + \cdots + \lambda^{z_i} \cdot (x-r_i)^{z_i}$ be an arbitrary polynomial of degree at most $\ell$ in $I_i^{z_i}$. Since $p_i \cdot p_i^j (x-r_i)^{(z_i-j)} = (x-r_i) \cdot p_i^{j+1}(x-r_i)^{(z_i-j-1)}$, we may assume without loss of generality that the degree of $\lambda^j(x)$ is at most 0 for each $j < z_i$ (if this fails for a particular $j < z_i$, subtract the appropriate multiple of $(x-r_i)$ from $\lambda^j(x)$ and add the appropriate multiple of $p_i$ to $\lambda^{j+1}(x)$). Thus, $c(x)$ is an integer linear combination of $\{p_i^{(z_i-a)}(x-r_i)^a : 0 \leq a \leq z_i\}$ and $\{x^a \cdot (x-r_i)^{z_i}\}_{a=1}^{\ell-z_i}$, as claimed. □*(Lemma 2)*

We remark that using standard techniques (see Appendix B), given bases for the (full-dimensional) lattices $L_i$, a basis $B$ for the intersection lattice $L = \bigcap_{i=1}^n L_i$ can be computed. By Lemma 2, $L$ corresponds exactly to the space of polynomials in the intersection ideal $I$ of degree at most $\ell$.

Let $L'$ be a re-scaling of the lattice $L$ where $(v_0, v_1, \ldots, v_\ell) \in L$ iff $(v_0, v_1 \cdot (K/2), \ldots, v_\ell \cdot (K/2)^\ell) \in L'$. We now show that applying the LLL algorithm to the lattice $L'$ gives us the polynomial we are looking for:

**Lemma 3** *Let $v'$ be the lattice vector returned by the LLL lattice basis reduction algorithm when applied to the lattice $L'$, and let $c(x)$ be the corresponding polynomial. Then for any $m$ with $|m| \leq K/2$ such that:*

$$\prod_{i=1}^n p_i^{a_i z_i} > (\ell+1) \cdot \left( \prod_{i=1}^n p_i^{\binom{z_i+1}{2}} \right)^{\frac{1}{\ell+1}} \cdot K^{\ell/2}$$

*we have that $m$ will be an integer root of the polynomial $c(x)$.*

**Proof:** Let $G = \left( \prod_{i=1}^n p_i^{\binom{z_i+1}{2}} \right)^{\frac{1}{\ell+1}} \cdot (K/2)^{\ell/2}$. Lemma 1 shows that there exists a vector $v$ in $L'$ such that for all $j$, we have that $|v_j| \leq G$, and so $\|v\| \leq \sqrt{\ell+1} \cdot G$. Thus, the LLL lattice reduction algorithm returns a vector $v' \in L'$ which is at most a factor of $2^{\ell/2}$ times larger: $\|v'\| \leq \left(2^{\ell/2}\right) \cdot \sqrt{\ell+1} \cdot G$. Hence, $\|v'\|_1 \leq \sqrt{(\ell+1)} \cdot \|v'\|_2 \leq \left(2^{\ell/2}\right) \cdot (\ell+1) \cdot G$. This corresponds to a polynomial $c(x)$ such that for all integers $m$ with $|m| \leq K/2$, we have that $|c(m)| \leq \left(2^{\ell/2}\right) \cdot (\ell+1) \cdot G$; on the other hand, of course, $c(x) \in I$, so by construction $c(m) \equiv 0 \bmod (\prod_i p_i^{a_i z_i})$. The Lemma follows. □ *(Lemma 3)*

Thus, we see that under the condition prescribed by the Theorem, we can efficiently find a polynomial whose set of roots contains all codewords with sufficient weighted agreement with the received word, and the decoding can then be completed by finding all integer roots of this polynomial using the polynomial time algorithm for factoring polynomials in $\mathbb{Z}[x]$ from [10]. □ *(Theorem 3)*

List Decode($\vec{r}, \ell, z_1, z_2, \ldots, z_n$)

1. Let $I_1^{z_i}$ be the set of polynomials that are integer linear combinations of $\{p_i^a(x - r_i)^{(z_i - a)}\}_{a=0}^{z_i}$.
2. Compute a basis for the *lattice* $L$ of all degree $\ell$ polynomials belonging to $\bigcap_{i=1}^n I_i^{z_i}$.
3. Scale this lattice by multiplying the $i$'th coordinate by $(K/2)^{i-1}$ to produce the lattice $L'$.
4. Run LLL to find a short vector $v'$ in $L'$; let it correspond to a degree $\ell$ polynomial $c(x) \in \mathbb{Z}[x]$.
5. Find all integer roots $m$ of $c(x)$ (for example, by factoring $c(x)$ over $\mathbb{Z}[x]$ using [10]).
6. For each root $m$ with $|m| \leq K/2$, define the vector $\vec{a} = (a_1, a_2, \ldots, a_n)$ by $a_i = 1$ if $m \equiv r_i (\mathrm{mod} p_i)$, and $a_i = 0$ otherwise. Output $m$ if $\vec{a}$ satisfies Condition (6).

Figure 1: The list decoding algorithm

For easy reference, we summarize in Figure 1 the main steps of the algorithm from the above proof of Theorem 3.

We now present an alternative algorithmic proof of Theorem 3, in which we translate the ideal-based reasoning given above directly into an explicit lattice.

**Alternative Proof (of Theorem 3):** As in the original proof, we will seek to find a polynomial $c(x)$, with coefficients that are bounded in size and with degree at most $\ell$, such that for all $i$, $1 \leq i \leq n$, the following Condition holds:

(*) For all integers $m$ such that $|m| \leq K/2$, we have that $m \equiv r_i \bmod p_i$ implies $c(m) \equiv 0 \bmod p_i^{z_i}$.

Here, we present an alternative method to find such polynomials. Recall the definitions of the ideals $I_i^{z_i}$ and the intersection ideal $I$ from the original proof, to provide intuition for our construction. We build an explicit lattice $L$ in which *all* polynomials (of degree at most $\ell$) are represented, but where we also represent, for each ideal $I_i^{z_i}$, the possible translates of these polynomials by elements of the ideal (again restricted to degree at most $\ell$). Thus, polynomials that are present in all the ideals $I_i^{z_i}$ can be translated to $0$ in each of the ideals. We constrain all *non-zero* translations to contribute a very large factor to the norm of the corresponding vector. Thus, we obtain a lattice $L$ in which all polynomials are represented (by many vectors), but in which polynomials *outside* the intersection ideal $I$ must be represented by very long vectors, whereas polynomials *inside* the ideal $I$ have one representative vector (where the polynomial has been translated to 0 for each ideal) that is quite short. Thus, the construction essentially mimics the steps of the proof of Lemma 1 to give this implicit representation, allowing us to extract small polynomials in the intersection ideal $I$.

Let us now describe the lattice $L$ formally, by presenting an explicit basis. The lattice $L$ will have $(n+1) \cdot (\ell+1)$ dimensions, conceptually separated into $n+1$ blocks of $\ell+1$ components each. We represent the basis vectors by the *rows* of the following matrix, described modularly:

$$\begin{pmatrix} \begin{array}{c||c|c|c|c} 0 & 0 & \cdots & 0 & qM^{(n)} \\ \hline 0 & 0 & \cdots & qM^{(n-1)} & 0 \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline 0 & qM^{(1)} & \cdots & 0 & 0 \\ \hline\hline \Lambda & qI_{(\ell+1)} & \cdots & qI_{(\ell+1)} & qI_{(\ell+1)} \end{array} \end{pmatrix} \qquad (7)$$

Above, all component matrices are $(\ell+1)$ by $(\ell+1)$. Let $G = \left(\prod_{i=1}^{n} p_i^{\binom{z_i+1}{2}}\right)^{\frac{1}{\ell+1}} \cdot (K/2)^{\ell/2}$. Above, $q = \sqrt{\ell+1} \cdot G \cdot 2^{(n+1)(\ell+1)}$. The matrix $I_{(\ell+1)}$ simply represents the $(\ell+1)$ by $(\ell+1)$ identity matrix. The matrix $\Lambda$ is an identity matrix with each diagonal entry scaled as shown:

$$\Lambda = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & K/2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & (K/2)^{\ell} \end{pmatrix}$$

Thus, the last $\ell+1$ basis vectors correspond to the polynomials $1,\ x,\ x^2,\ \ldots,\ x^{\ell}$. These vectors are broken up into $(n+1)$ blocks of $(\ell+1)$ dimensions each: the first block measures how large the polynomial can be when evaluated on an integer of magnitude at most $(K/2)$; the remaining $n$ blocks are each used to measure the residue of the polynomial modulo the ideals $I_1^{z_1}, \ldots, I_n^{z_n}$. In order to measure this residue, we need to have other vectors in our lattice that allow us to "reduce" this residue by the generators of the ideals. The matrices $M^{(i)}$, as defined in the original proof, serve this purpose. Recall that the rows of the matrix $M^{(i)}$ correspond to the generating polynomials of the ideal $I_i^{z_i}$ restricted to polynomials of degree at most $\ell$. Note, however, that in our proof, we will only need that the polynomials corresponding to integer linear combinations of the rows of $M^{(i)}$ satisfy Condition (*) – we do not need to refer to the ideals themselves:

Now, for any polynomial $c(x)$ of degree at most $\ell$, we can mimic the steps of the proof of Lemma 1 by adding and subtracting integer multiples of the polynomials represented by the rows of $M^{(i)}$ for each $i$. Thus, for any polynomial $c(x)$ of degree at most $\ell$, there exists a vector $v \in L$ such that for all $i$ between $1$ and $n$, in the $i$'th block of $v$ there is one of at most $p_i^{\binom{z_i+1}{2}}$ vectors. Hence, (again mimicking the proof of Lemma 1), if we consider all polynomials $c(x)$ of degree at most $\ell$ such that $0 \le c_j \le G/(K/2)^j$ for all $j$, we must find either a corresponding vector $v \in L$ with all 0's beyond the first $\ell+1$ coordinates, or two vectors $v^{(1)}$ and $v^{(2)}$ such that all coordinates beyond the first $\ell+1$ agree. In the second case, $v = v^{(1)} - v^{(2)}$ will have the property that $|v_j| \le G$ for all $j$ between $1$ and $\ell+1$, and $v_j = 0$ for all $j > \ell+1$. Thus, by construction, $v$ will be, for each $i$, an integer linear combination of the rows of $M^{(i)}$, and therefore correspond to a polynomial satisfying Condition (*) for all $i$.

Hence, there exists a lattice vector $v \in L$ such that $||v|| \le \sqrt{\ell+1} \cdot G$. We use the LLL lattice basis reduction algorithm to find a short vector $v'$ in this lattice $L$. The standard analysis of LLL would only guarantee that $||v'||$ is within a $2^{((n+1)\cdot(\ell+1)-1)/2}$ factor of the shortest vector, but because of the special structure of the lattice $L$, we show that LLL returns a vector that is within a $2^{\ell/2}$ factor of the shortest vector:

**Lemma 4** *When the LLL algorithm is applied to the lattice $L$ above, the first basis vector $b_1$ returned by LLL is such that $||b_1|| \le 2^{\ell/2} \cdot \lambda_1(L)$, where $\lambda_1(L)$ is the norm of the shortest nonzero vector in $L$.*

**Proof:** Let $\tilde{n} = (n+1)\cdot(\ell+1)$ be the dimension of the lattice $L$. We will refer to standard facts about LLL-reduced bases and shortest vectors in a lattice, which can be found for example in Section 2.6 of Cohen [2].

We recall two basic facts. Let $b_1^*, \ldots, b_{\tilde{n}}^*$ be the orthogonalization of the LLL-reduced basis $b_1, \ldots, b_{\tilde{n}}$ returned by the algorithm. In other words, $b_i^*$ is defined inductively to equal $b_i - \sum_{j<i} \langle b_i, b_j^* \rangle \cdot b_j^*$. Then we have:

1. For any $i, j$ such that $1 \leq j \leq i \leq \tilde{n}$, we have that $\|b_j^*\| \leq 2^{(i-j)/2} \cdot \|b_i^*\|$.
2. $\lambda_1(L) \geq \min_{j \in \{1,\ldots,\tilde{n}\}} \|b_j^*\|$.

These two facts together imply that $\|b_1\| \leq 2^{(\tilde{n}-1)/2} \lambda_1(L)$. We show that in fact: $\lambda_1(L) \geq \min_{j \in \{1,\ldots,\ell+1\}} \|b_j^*\|$, which when combined with the first fact, establishes the Lemma.

Recall that $L$ is constructed so that any non-zero entry of a lattice vector beyond the first $(\ell+1)$ coordinates must have magnitude at least $q = \sqrt{\ell+1} \cdot G \cdot 2^{\tilde{n}}$. We know that a vector in $L$ exists that has norm at most $\sqrt{\ell+1} \cdot G$. We thus know already that $\|b_1\| \leq 2^{\tilde{n}/2} \cdot \lambda_1(L) \leq q/(2^{\tilde{n}/2})$. Hence, $b_1$ can be non-zero only in the first $\ell+1$ coordinates. On the other hand, at least one of the basis vectors $b_1, \ldots, b_{\ell+2}$ must have a non-zero component beyond the first $\ell+1$ coordinates by linear independence. Let $b_a$ be the first vector to have a non-zero component in some coordinate $t > \ell+1$ (by the previous statement, $a \leq \ell+2$). Then $\|b_a\| \geq q$, and by construction of orthogonalization, it must be that $\|b_a^*\| \geq q$ as well, since $b_a^*$ must also have a non-zero component in coordinate $t$.

Now, for all $i \geq a$, we have that:

$$\|b_i^*\| \geq 2^{(a-i)/2} \cdot \|b_a^*\| \geq q/(2^{\tilde{n}/2}) \geq \|b_1\| = \|b_1^*\|.$$

Thus, $\min_{j \in \{1,\ldots,\ell+1\}} \|b_j^*\| = \min_{j \in \{1,\ldots,\tilde{n}\}} \|b_j^*\|$, and the Lemma is established. □ *(Lemma 4)*

Thus, the polynomial corresponding to the vector $b_1$ returned by the LLL lattice basis reduction algorithm has the property we seek:

**Lemma 5** *Let $v'$ be the lattice vector returned by the LLL lattice basis reduction algorithm when applied to the lattice L, and let $c(x)$ be the corresponding polynomial. Then for any $m$ with $|m| \leq K/2$ such that:*

$$\prod_{i=1}^{n} p_i^{a_i z_i} > (\ell+1) \cdot \left( \prod_{i=1}^{n} p_i^{\binom{z_i+1}{2}} \right)^{\frac{1}{\ell+1}} \cdot K^{\ell/2}$$

*we have that $m$ will be an integer root of the polynomial $c(x)$.*

**Proof:** By the existence of a vector in the lattice with norm at most $\sqrt{\ell+1} \cdot G$ and Lemma 4, we have that $\|v'\| \leq \left(2^{\ell/2}\right) \cdot \sqrt{\ell+1} \cdot G$. Hence, $\|v'\|_1 \leq \sqrt{(\ell+1)} \cdot \|v'\|_2 \leq \left(2^{\ell/2}\right) \cdot (\ell+1) \cdot G$. This corresponds to a polynomial $c(x)$ such that for all integers $m$ with $|m| \leq K/2$, we have that $|c(m)| \leq \left(2^{\ell/2}\right) \cdot (\ell+1) \cdot G$. On the other hand, since $v'$ is 0 in all coordinates beyond the first $\ell+1$, the corresponding polynomial $c(x)$ satisfies Condition (*), and so $c(m) \equiv 0 \bmod \left(\prod_i p_i^{a_i z_i}\right)$. The Lemma follows. □ *(Lemma 3)*

Thus, we see that under the condition prescribed by the Theorem, we can efficiently find a polynomial whose set of roots contains all codewords with sufficient weighted agreement with the received word, and the decoding can then be completed by finding all integer roots of this polynomial using the polynomial time algorithm for factoring polynomials in $\mathbb{Z}[x]$ from [10]. □ *(Theorem 3)*

### 3.4 Decoding for Interesting Weightings

We now get specific results for the CRT code for interesting choice of weights on the coordinate positions through an appropriate choice of parameters (like $\ell, z_i$) in Theorem 3. We begin by proving a version of Theorem 3 with arbitrary (not necessarily integer) values of $z_i$.

**Theorem 4** *For list decoding of CRT codes, for any tolerance parameter $\varepsilon > 0$, and non-negative reals $z_i$, given a received word $r$, we can in time polynomial in $n, \log N$ and $1/\varepsilon$, find a list of all codewords such that*

$$\sum_{i=1}^{n} a_i z_i \log p_i \geq \sqrt{\log K \Big( \sum_{i=1}^{n} z_i^2 \log p_i + \varepsilon z_{\max}^2 \Big)} \ . \tag{8}$$

**Proof:** We may assume that $z_{\max} = 1$ (note that the condition of (8) is invariant under scaling of the $z_i$'s, so this can be ensured by dividing out all weights by $z_{\max}$). We will prove the claimed result by appealing to Theorem 3 on a suitably chosen set of *integer* weights $z_i^*$.

Let $A$ be a large integer to be specified later in the proof. Set $z_i^* = \lceil A z_i \rceil$. By Theorem 3, for any positive integer $\ell$ we can decode (in $\mathrm{poly}(n, \log N, A, \ell)$ time) provided

$$\sum_{i=1}^{n} a_i z_i^* \log p_i > \log(\ell+1) + \frac{\ell}{2} \log K + \frac{1}{\ell+1} \sum_{i=1}^{n} \binom{z_i^*+1}{2} \log p_i .$$

We would like to pick a good choice for $\ell$. Since $A z_i \leq z_i^* < A z_i + 1$, the above condition is met whenever

$$\sum_{i=1}^{n} a_i z_i \log p_i \geq \frac{\log(\ell+1)}{A} + \frac{\ell}{2A} \log K + \frac{A}{2(\ell+1)} \sum_{i=1}^{n} \Big( z_i^2 + \frac{3}{A} z_i + \frac{2}{A^2} \Big) \log p_i \ . \tag{9}$$

Define $Z_i = z_i^2 + \frac{3}{A} z_i + \frac{2}{A^2}$ for $1 \leq i \leq n$. Let us pick

$$\ell = \left\lceil A \sqrt{\frac{\sum_{i=1}^{n} Z_i \log p_i}{\log K}} \right\rceil - 1. \tag{10}$$

It is not difficult to see that for this choice of $\ell$, the Condition (9) is met whenever

$$\sum_{i=1}^{n} a_i z_i \log p_i \geq \frac{1}{A} \log \Big( A \sqrt{\frac{\sum_{i=1}^{n} Z_i \log p_i}{\log K}} + 1 \Big) + \sqrt{\log K \sum_{i=1}^{n} Z_i \log p_i} \ . \tag{11}$$

For $A \geq \frac{10 \log N}{\varepsilon}$, the right side of Equation (11) above is at most

$$O(\frac{\log \log N}{\log N}) + \sqrt{\log K \Big( \sum_{i=1}^{n} z_i^2 \log p_i + \frac{\varepsilon}{2} \Big)} \leq \sqrt{\log K \Big( \sum_{i=1}^{n} z_i^2 \log p_i + \varepsilon \Big)}$$

for large $N$. Thus the Condition (11) is met provided

$$\sum_{i=1} a_i z_i \log p_i \geq \sqrt{\log K \Big( \sum_{i=1}^{n} z_i^2 \log p_i + \varepsilon \Big)},$$

and the proof is complete by noting that $A = O(\frac{\log N}{\varepsilon})$ and $\ell = O(\varepsilon^{-1} \log^{3/2} N)$, and so the overall runtime is polynomial in $n, \log N$ and $\varepsilon^{-1}$. $\square$

**Corollary 4** *For list decoding of CRT codes, for any tolerance parameter $\varepsilon > 0$, and non-negative real weights $\beta_i$, given a received word $r$, we can, in time polynomial in $n, \log N$ and $1/\varepsilon$, find a list of all codewords whose $\vec{\beta}$-weighted agreement with $r$ satisfies:*

$$\sum_{i=1}^{n} a_i \beta_i \geq \sqrt{\log K \left( \sum_{i=1}^{n} \frac{\beta_i^2}{\log p_i} + \varepsilon \max_j \frac{\beta_j^2}{\log p_j} \right)} . \qquad \square$$

Note that the above Corollary implies that, in the limit of large $p_i$, we can decode up to (essentially) the combinatorial bound of Theorem 1 with $\alpha_i = \log p_i$ and $D_\alpha = \log(N/K)$. Let us now collect further results for the "usual" uniform weighting of the codeword positions, i.e., $\beta_i = 1$ for all $i$.

**Theorem 5** *For list decoding of CRT codes, for any $\varepsilon > 0$, we can in time polynomial in $n, \log N$ and $1/\varepsilon$, find a list of all codewords which agree with a received word in $t$ places provided $t \geq \sqrt{k(n+\varepsilon)}$.*

**Proof:** Let us apply Theorem 4 with $z_i = 1/\log p_{k+1}$ for $1 \leq i \leq k$, $z_i = 1/\log p_i$ for $k < i \leq n$, and $\varepsilon' = \varepsilon \log p_{k+1}$. This gives that we can decode whenever the number of agreements $t$ is at least

$$k - \frac{\log K}{\log p_{k+1}} + \sqrt{\frac{\log K}{\log^2 p_{k+1}} \left( \log K + \sum_{i=k+1}^{n} \frac{1}{\log p_i} + \varepsilon' \right)} .$$

Define $\Delta \stackrel{\text{def}}{=} k - \frac{\log K}{\log p_{k+1}}$; clearly $\Delta \geq 0$. Since $\log p_{k+1} \leq \log p_i$ for $i = k+1, \cdots, n$, the above condition is met whenever $t \geq \Delta + \sqrt{(k-\Delta)(n-\Delta+\varepsilon)}$. Now, a simple application of Cauchy-Schwartz shows $\Delta + \sqrt{(k-\Delta)(n-\Delta+\varepsilon)} \leq \sqrt{k(n+\varepsilon)}$, and thus our decoding algorithm works whenever $t \geq \sqrt{k(n+\varepsilon)}$. $\square$

**Theorem 6** *For list decoding of CRT codes, for any $\varepsilon > 0$, we can in time polynomial in $n, \log N$ and $1/\varepsilon$, find a list of all codewords which agree with a received word in $t$ places provided $t \geq \sqrt{\log K \left( \sum_{i=1}^{n} \frac{1}{\log p_i} + \varepsilon \right)}$.*

**Proof:** This follows from Corollary 4 with $\beta_i = 1$ for $1 \leq i \leq n$. $\square$

Note that the above matches the combinatorial bound of Corollary 2. The bounds in Theorem 5 and Theorem 6 are incomparable in general.

# Acknowledgments

We thank Dan Boneh for informing us of his work [1] and making a copy of his paper available on his homepage. We thank Daniele Micciancio for useful discussions on Lattices and pointers to [13].

# References

[1] D. Boneh. Finding Smooth integers in short intervals using CRT decoding. *Proc. of STOC 2000*, to appear.

[2] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer Verlag, Berlin-Heidelberg, 1993.

[3] P. ELIAS. List decoding for noisy channels. *Wescon Convention Record*, Part 2, Institute of Radio Engineers (now IEEE), pp. 94-104, 1957.

[4] G. D. Forney. Generalized Minimum distance decoding. *IEEE Trans. on Information Theory*, Vol. 12 (1966), pp. 125-131.

[5] O. Goldreich, D. Ron and M. Sudan. Chinese Remaindering with errors. *IEEE Trans. on Information Theory*, to appear. Preliminary version appeared in *Proc. of 31st STOC*, 1999, pp. 225-234.

[6] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and Algebraic-geometric codes. *IEEE Trans. on Information Theory*, 45 (1999), pp. 1757-1767. Preliminary version appeared in *Proc. of FOCS'98*.

[7] V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. *Proc. of STOC 2000*, to appear.

[8] J. Håstad and M. Näslund. The security of all RSA and Discrete Log bits. *Proc. of 39th FOCS*, 1998, pp. 510-519.

[9] H. Krishna, B. Krishna, K. Y. Lin and J. D. Sun. *Computational Number Theory and Digital Signal Processing: Fast algorithms and error control techniques*. Boca Raton, FL: CRC, 1994.

[10] A. K. Lenstra, H. W. Lenstra and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261 (1982), pp. 515-534.

[11] D. M. Mandelbaum. On a class of arithmetic codes and a decoding algorithm. *IEEE Trans. on Information Theory*, 21 (1976), pp. 85-88.

[12] D. M. Mandelbaum. Further results on decoding arithmetic residue codes. *IEEE Trans. on Information Theory*, 24 (1978), pp. 643-644.

[13] D. Micciancio. Lecture notes on *Lattices in Cryptography and Cryptanalysis*, Fall 1999, UCSD. Available at `http://www-cse.ucsd.edu/~daniele/cse291fa99.html`.

[14] M. A. Shokrollahi and H. Wasserman. List decoding of algebraic-geometric codes. *IEEE Trans. on Information Theory*, Vol. 45, No. 2, March 1999, pp. 432-437.

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

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

[17] J. M. Wozencraft. List Decoding. *Quarterly Progress Report*, Research Laboratory of Electronics, MIT, Vol. 48 (1958), pp. 90-95.

# A Ideals and Error-correcting codes

In this section we describe a framework for studying algebraic error-correcting codes and the decoding problem in the setting of ideals in commutative rings. We give a decoding algorithm in the same framework — this decoding algorithm abstracts and unifies known algorithms for this task, and specializes to the algorithm given in Section 3.2 for CRT decoding. Here we focus only on the qualitative features of codes and decoding. A more quantitative version of this abstraction can be developed using appropriately defined notions of "size" for elements and ideals of the underlying rings. This will be the focus of the journal version of the paper where this unified ideal-theoretic framework will be emphasized and developed further. In the following discussion, we assume that the reader is familiar with the concept of commutative rings, integral domains and ideals.

**Definition 1 (Ideal error-correcting code)** *An* Ideal Code $\mathcal{C}$ *is given by an integral domain* $R$ *and ideals* $J_1, \ldots, J_n \subseteq R$. *The message space of* $\mathcal{C}$ *is some subset* $\mathcal{M} \subseteq R$. *The alphabets of* $\mathcal{C}$ *are given by* $\Sigma_i = R/J_i$. *(Note that the definition is interesting only if* $R/J_i$ *is finite.) The code maps the element* $a \in \mathcal{M}$ *to the sequence* $\langle a + J_1, \ldots, a + J_n \rangle$.

While not every linear code is an ideal code, many commonly studied ones, including Reed-Solomon codes, Algebraic-geometry codes and the CRT codes, are ideal codes. Now, consider an instance of the list-decoding problem with a received vector $\langle r_1, \ldots, r_n \rangle$. Informally, the algorithm of [16, 14, 6, 5, 1] cast this problem as follows:

**Definition 2 ("Ideal"-lic list-decoding)** *Let* $R[x]$ *be the ring of polynomials in* $x$ *with coefficients from* $R$. *Let* $I_i = (x - r_i) + J_i$ *be the ideal* $\{a(x) \cdot (x - r_i) + b(x) \cdot p | a(x), b(x) \in R[x], p \in J_i\}$. *Find a list of all elements of* $R[x]$ *of the form* $x - f$, *with* $f \in R$, *such that* $x - f \in I_i$ *for "many" values of* $i \in \{1, \ldots, n\}$.

From this formulation, their algorithms (and the use of factoring there) emerge naturally.

**(Weighted) List-decoding algorithm**

1. Pick vector $z_1, \ldots, z_n$ appropriately.
2. Find a non-zero polynomial $c(x)$ (with "small" coefficients) such that $c \in \prod_{i=1}^{n} I_i^{z_i}$.
3. Factor $c$ and report the list of linear factors $x - f$.

Note that the notion of products of ideals is a well-studied one. When the ideals $J_i$ and $J_j$ are relatively prime, the product of the ideals $I_i^{z_i}$ and $I_j^{z_j}$ equals their intersection and this fact often leads to some quantitative improvements in the bounds; however is not critical to the correctness of the approach.

In specializing the algorithm above to specific cases, the following ingredients need to be added: (1) Algorithms for finding representations of intersections and products of ideals. (2) Explicit notion of "small" and algorithms for finding "small" elements in the ideal. (3) Choice of $z_i$'s and a quantitative analysis of the performance of the algorithm (since a list-decoding algorithm to recover from zero errors may not be very interesting). The application to CRT decoding in Section 3.2 is obtained by finding and adding these ingredients.

# B Lattice Algorithms

We recall some standard techniques in the algorithmics of lattices, in particular computing the intersection of full-dimensional lattices. A more formal treatment may be found in [2, 13].

Let $L$ be any full-dimensional lattice of dimension $d$, with basis given by the rows of the matrix $M$. We define the *dual* $L^*$ of the lattice $L$ to be $\{u \in \mathcal{R}^d \ : \ u \cdot v \in \mathbb{Z} \text{ for all } v \in L\}$. Note that the rows of $(M^{-1})^\top$ give a basis for $L^*$.

Note also that given bases for two lattices $L_1$ and $L_2$, a basis for the closure of union of the two lattices (denoted $L_1 \cup L_2$) can be found efficiently using algorithms for computing the Hermite Normal Form of a generating set of vectors. Now, to compute a basis for the intersection of two lattices $L_1$ and $L_2$, observe that $L_1 \cap L_2 = (L_1^* \cup L_2^*)^*$. Therefore, by combining the facts above, one obtains an efficient algorithm for computing the intersection of full-dimensional lattices.