Noisy Chinese Remaindering in the Lee Norm

Igor E. Shparlinski
Department of Computing
Macquarie University, Sydney, NSW 2109, Australia
igor@ics.mq.edu.au

Ron Steinfeld
Department of Computing
Macquarie University, Sydney, NSW 2109, Australia
rons@ics.mq.edu.au

May 5, 2004

Abstract

We use lattice reduction to obtain a polynomial time algorithm for recovering an integer (up to a small interval) from its residues modulo sufficiently many primes, when the residues are corrupted by a small additive noise bounded in the Lee norm. Our results are similar to those obtained for Hamming norm, but based on rather different arguments.

1 Introduction

For integers s and $m \geq 1$ we denote by $\lfloor s \rfloor_m$ the remainder of s on division by m and by $\langle s \rangle_m$ the smallest (by absolute value) residue of s modulo m, that is

$$\langle s \rangle_m = \begin{cases} \lfloor s \rfloor_m & \text{if } 0 \leq \lfloor s \rfloor_m \leq m/2 \\ \lfloor s \rfloor_m - m & \text{if } m/2 < \lfloor s \rfloor_m < m \end{cases}$$

Also, for an integer $m \geq 2$ we denote by $\mathbb{Z}_m$ the residue ring modulo m. Finally for an integer $K \geq 1$ we denote by $\mathcal{Z}[K]$ the set integers in

the interval $[0, K-1]$. We assume that $\mathbb{Z}_m$ is represented by the elements of $\mathcal{Z}[m]$ but we do not identify them. In particular, for any $a, b \in \mathbb{Z}_m$ we certainly have $ab \in \mathbb{Z}_m$ while the similar property does not obviously hold for $\mathcal{Z}[m]$.

The *Chinese Remainder Code* encodes an integer $a \in \mathcal{Z}[K]$ as the vector

$$\mathbf{a_m} = (\lfloor a \rfloor_{m_1}, \ldots, \lfloor a \rfloor_{m_n})$$

of its residues modulo n pairwise relatively prime integers $\mathbf{m} = (m_1, \ldots, m_n)$. Then the Chinese Remainder Theorem (CRT) states that the encoding $\mathbf{a_m}$ uniquely determines a in $\mathcal{Z}[K]$ as soon as $m_1 \cdots m_n \geq K$, and moreover, using the Euclid algorithm, gives an efficient algorithm for recovering the unique a from $\mathbf{a_m}$ and $\mathbf{m} = (m_1, \ldots, m_n)$.

By choosing the product $m_1 \cdots m_n$ sufficiently large compared to K one can hope to use the resulting redundancy in $\mathbf{a_m}$ to recover a even in the presence of some noise, that is, to use the Chinese Remainder Code as an error correcting code. In this setting, one is given the vector $\mathbf{y} = \mathbf{a_m} + \mathbf{e}$ for some sufficiently 'small' noise vector $\mathbf{e}$, and tries to recover a. The precise meaning of 'small' noise can be quantified by defining some norm $\|.\|$ on the space of noise vectors and assuming that the noise vector satisfies $\|\mathbf{e}\| < h$ for some noise norm bound h. For some choices of the norm $\|.\|$ there exists a non-zero 'error-correcting bound' h^* such that $a \in \mathcal{Z}[K]$ is always uniquely determined from $\mathbf{y}$ when $h < h^*$. For $h \geq h^*$ we may not be able to uniquely determine a from $\mathbf{y}$, but we may still be able to find a small *list* containing all possible candidates for a in $\mathcal{Z}[K]$, that is, a list of all $a \in \mathcal{Z}[K]$ such that $\|\mathbf{a_m} - \mathbf{y}\| < h$ — this is called a *List Decoding Problem*.

Recently, there has been a series of interesting results (see [3, 8, 11, 32]) on efficient algorithms for list decoding of the Chinese Remainder Code and other codes as well. However, these results assume a noise vector $\mathbf{e}$ bounded in the *Hamming weight* norm, that is, $\mathbf{e}$ is assumed to have zero coordinates except for less than h non-zero ones, which are allowed to be arbitrary. In this paper we consider the case when all n coordinates of the noise vector $\mathbf{e}$ may be non-zero, but are all assumed to be smaller in magnitude than h — that is, we are assuming a noise vector bounded in the *Lee norm*; the precise formulation is presented in Section 3 below. We use lattice reduction techniques to construct a polynomial-time decoding algorithm for the Chinese Remainder Code in this setting, and give rigorous conditions under which the algorithm successfully recovers an interval $I \subseteq \mathcal{Z}[K]$ of width at most $2h$ consisting of all solutions to our decoding problem.

We remark that our algorithm for the Lee norm uses lattice basis reduction techniques more directly than the list decoding algorithm for the Hamming norm [8], which first transforms the problem to an algebraic interpolation problem and then applies lattice techniques to the algebraic problem. Moreover, in these list decoders, the number of solutions (size of the list) is *small*, that is, bounded by a polynomial in the length of the input, so the decoder can explicitly list the solutions in polynomial time. In our case, the number of solutions is in general exponential in the length of the input, so we cannot hope to explicitly list them. Instead, we use the fact that this large set of solutions is compactly representable as an interval, and our algorithm computes the endpoints of this interval in polynomial time.

The problem which we consider in this paper as well as our approach has taken some motivation from the *hidden number problem* introduced by Boneh and Venkatesan in [4, 5], see also [30] for a survey of several recent developments and new applications.

Indeed, in the hidden number problem we are given approximations to the residues of at_i, $i = 1, \ldots, n$, modulo a given prime p for randomly selected multipliers $t_1, \ldots, t_n \in \mathbb{Z}_p$ (and we also know that $a \in \mathcal{Z}[p]$). In our situations, we are given approximations to the residues of a, $i = 1, \ldots, n$, modulo randomly selected primes p_i (and we also know that $a \in \mathcal{Z}[K]$, for a given bound K). Surprisingly enough, even though the algorithms and their analysis have very little in common (except for the general set up of the associated lattices), the obtained results are somewhat similar. It is shown in [4] that about $\log^{1/2} p$ most significant bits for each of the residues $at_i \pmod p$ are enough to recover a. Our algorithm works if we are given about $\log^{1/2} K$ most significant bits for each of the residues $a \pmod{p_i}$, see the discussion after the proof of Theorem 3. In fact, we use a slightly stronger lattice reduction algorithm of [1] than that used in [4], so we can reduce the number of bits to $o(\log^{1/2} K)$. Similar improvements applies to the algorithm of [4] as well, see [20, 30].

Interestingly, in some cases when the noise bound h is very small and the solutions can be explicitly listed, the list decoding algorithms for the Hamming norm can be modified to solve our Lee norm decoding problem. Such an algorithm has been presented (in a more general form) by Boneh [3]. This algorithm explicitly lists all solutions to our decoding problem in polynomial-time whenever $2h < n/k$, where $k = \log K / \log p_{min}$, and $p_{min} = \min_{1 \le i \le n} p_i$. Our algorithm is able to work with much larger (for example, exponentially larger) values of h.

We also remark that the result of [29] is a Lee norm analogue of Hamming norm results of [2, 9, 12, 13, 18, 23, 26, 28, 29, 31, 32] on noisy polynomial reconstruction problem and algebraic geometry codes list decoding.

Finally, several possible cryptographic applications of noisy polynomial reconstruction have been outlined in [16, 17]. It would be interesting to study possible cryptographic applications of noisy Chinese remaindering as well.

2 Preliminaries

For a prime p and an integer z we denote *Lee norm of z modulo p* as

$$\|z\|_{L,p} = \min_{k \in \mathbb{Z}} |z - kp| = |\langle z \rangle_p|.$$

Accordingly, given a *basis* set of n primes $\mathbf{p} = (p_1, \ldots, p_n)$, and a residue vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{Z}_{p_1} \times \ldots \mathbb{Z}_{p_n}$, we define the associated *Lee norm of* $\mathbf{z}$ *modulo* $\mathbf{p}$ by

$$\|\mathbf{z}\|_{L,\mathbf{p}} = \max_{1 \le i \le n} \|z_i\|_{L,p_i}.$$

Given two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{Z}_{p_1} \times \ldots \mathbb{Z}_{p_n}$ we then define the corresponding *Lee metric* as $d_{L,\mathbf{p}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_{L,\mathbf{p}}$, where the difference $\mathbf{x} - \mathbf{y}$ is computed componentwise.

Given a set S, we denote by $|S|$ its cardinality.

We also use $\|\mathbf{x}\|$ to denote the Euclidean norm of a finitely dimensional vector $\mathbf{x}$ with real components.

As we have mentioned, our algorithm is based on lattice reduction. Here we recall some definitions and relevant results.

Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$ be a set of linearly independent vectors in $\mathbb{R}^s$. The set of vectors

$$L = \left\{ \sum_{i=1}^{s} n_i \mathbf{b}_i \mid n_i \in \mathbb{Z} \right\},$$

is called an s-dimensional full rank lattice. The set $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$ is called a *basis* of L, and L is said to be spanned by $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$. We refer to [10] for the general background on lattices.

A basic lattice problem is the *shortest vector problem* (SVP): given a basis of a lattice L in $\mathbb{R}^s$, find a nonzero lattice vector $\mathbf{v} \in L$ of the smallest possible Euclidean norm $\|\mathbf{v}\|$ among all lattice vectors. The shortest vector problem generally refers to the Euclidean norm, but of course, other norms

are possible as well. Although the shortest vector problem appears to be **NP**-hard various approximate polynomial time algorithms can be designed, see [14, 21, 22] for references.

In this paper we actually need to solve a variation of SVP called the *closest vector problem* (CVP): given a basis of a lattice L in $\mathbb{R}^s$ and a "target" vector $\mathbf{t} \in \mathbb{R}^s$, find a lattice vector $\mathbf{v}$ which is closest in the Euclidean metric to $\mathbf{t}$. Fortunately, Kannan [15] has shown how to convert any polynomial-time approximation algorithm for SVP into a polynomial-time approximation algorithm for CVP. Namely, given an SVP algorithm which, given a basis of a lattice of dimension s finds a non-zero lattice vector of length within an approximation factor $\gamma(s) > 0$ (where $\gamma(s)$ is a non-decreasing function of s) of the shortest vector in L, the Kannan reduction transforms it into a polynomial-time CVP algorithm which given a basis of L in $\mathbb{R}^s$ and vector $\mathbf{t} \in \mathbb{R}^s$ finds a lattice vector whose distance from $\mathbf{t}$ is within approximation factor $s^{3/2}\gamma(s)^2$ of the distance of the closest vector in L to $\mathbf{t}$. For example, one can combine the reduction [15] from CVP to SVP with the pioneering SVP algorithm of Lenstra, Lenstra and Lovász [19]. However, here we use [15] together with the best known approximation polynomial-time result for SVP given in Corollary 15 of [1] to get the following statement (which is slightly stronger than that implied by [15] and [19]).

Lemma 1. *For any constant $\tau > 0$, there exists a randomised polynomial-time algorithm which, given an s-dimensional full rank lattice L, and a vector $\mathbf{t} \in \mathbb{R}^s$, finds a lattice vector $\mathbf{v}$ satisfying with probability exponentially close to 1 the inequality*

$$\|\mathbf{v} - \mathbf{t}\| < 2^{\tau s \log\log s/\log s} \min\{\|\mathbf{z} - \mathbf{t}\| \ : \ \mathbf{z} \in L\}.$$

Proof. By taking $k = \lceil c \log n \rceil$ in Corollary 15 of [1] where $c > 0$ is a sufficiently large constant, we obtain a randomised polynomial-time algorithm which approximates the shortest vector within $2^{0.25\tau s \log\log s/\log s}$ for any constant $\tau > 0$. If s is sufficiently large such that

$$s^{3/2} 2^{0.5\tau s \log\log s/\log s} \le 2^{\tau s \log\log s/\log s},$$

the result follows by using this algorithm as a shortest vector approximation oracle in the Kannan reduction [15] from the closest vector problem to shortest vector problem. For smaller values of s (that is, fixed s) we can use any of the exact polynomial-time algorithms of [1, 15] to find the closest vector in fixed-dimension lattices. □

The best deterministic polynomial-time algorithm known for the SVP problem, which can be also used in Lemma 1, has a slightly larger approximation factor $2^{\tau s \log^2 \log s / \log s}$, see [27].

3 Decoding Problem

Let $\mathcal{P}_\ell$ be a set of primes which exceed 2^ℓ for some integer length parameter $\ell \geq 1$.

Choose a basis of n distinct primes $\mathbf{p} = (p_1, \ldots, p_n) \in \mathcal{P}_\ell^n$. Fix the message space as $\mathcal{Z}[K]$. The pair $(\mathbf{p}, K)$ determines a Chinese Remainder Code as follows: an integer $a \in \mathcal{Z}[K]$ is encoded as its residue vector $\mathbf{a_p}$.

We consider the following decoding problem for the above Chinese Remainder Code $(\mathbf{p}, K)$. We are given the code parameters $(\mathbf{p}, K)$ and a vector

$$\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{Z}_{p_1} \times \ldots \mathbb{Z}_{p_n}$$

defined as $\mathbf{y} = \mathbf{a_p} + \mathbf{e}$, where $\mathbf{e} = (e_1, \ldots, e_n) \in \mathbb{Z}_{p_1} \times \ldots \mathbb{Z}_{p_n}$ is an additive noise vector of bounded Lee norm relative to prime basis $\mathbf{p}$, that is, $\|\mathbf{e}\|_{L,\mathbf{p}} < h$ for some $h \in \mathbb{N}$. Our goal is to find an interval $I = [A, B] \subseteq \mathcal{Z}[K]$ of a small length containing all the integers $b \in \mathcal{Z}[K]$ such that

$$d_{L,\mathbf{p}}(\mathbf{b_p}, \mathbf{y}) < h. \tag{1}$$

Note that by construction of the given vector $\mathbf{y}$, we know that an interval in $\mathcal{Z}[K]$ of width less than $2h$ consisting of integers b satisfying (1) always exists. Namely, the interval

$$I(a, h) = [\alpha_I, \beta_I], \tag{2}$$

where

$$\alpha_I = \max(0, a - (h-1) - \min_{1 \leq i \leq n} \langle a - y_i \rangle_{p_i})$$

and

$$\beta_I = \min(K-1, a + (h-1) - \max_{1 \leq i \leq n} \langle a - y_i \rangle_{p_i}),$$

which contains at least one integer a, is such an interval. The interval $I(a, h)$ can contain up to $2h - 1$ integers. Hence our problem has multiple solutions in general and the unique recovery of a is not possible. However, in the proof

of Theorem 3 we show that under certain conditions the interval $I(a,h)$ (of length at most $2h$) is in fact the set of *all* solutions to our problem in $\mathcal{Z}[K]$.

We note that there exist instances of our decoding problem when the interval $I(a,h)$ consists of exactly one integer a. Namely, this is readily seen to be the case when the error vector $\mathbf{e} = (e_1, \ldots, e_n)$ satisfies $e_i = h-1$ and $e_j = -(h-1)$ for some coordinate positions i and j in $\{1, \ldots, n\}$ such that $h - 1 < \lfloor p_i/2 \rfloor$ and $h - 1 < \lfloor p_j/2 \rfloor$.

4 Localization of Solutions

For a given CR code $(K, \mathbf{p})$ and a given vector $\mathbf{y}$ approximating the residue-vector of integer $a \in \mathcal{Z}[K]$, our problem is to find all solution integers in $\mathcal{Z}[K]$ whose residue-vectors are "close" (within a distance h) to $\mathbf{y}$ in the Lee Metric (namely are in the ball $B_h(\mathbf{y}) \subseteq \mathbb{Z}^n$ consisting of all vectors within distance h of $\mathbf{y}$). We know that the residue-vector of the integer $a \in \mathcal{Z}[K]$ is in the ball $B_h(\mathbf{y})$, and so are the residue-vectors of all integers in some neighbourhood of a, namely the interval $I(a,h)$ of Section 3 — all these solutions differ from a by less than h. The question we address here is the existence of solution integers outside the vicinity interval $[a-h, a+h]$ of a: are there any integers in $\mathcal{Z}[K] \backslash [a-h, a+h]$ whose residue-vectors are in the ball $B_h(\mathbf{y})$? We show that there are none, except for some exceptional "bad" choices of the code, namely those for which the prime base $\mathbf{p}$ is in some "bad subset" of $\mathcal{P}_\ell^n$ denoted $\mathcal{B}_{\ell,n}(h,K)$. This means that as long as $\mathbf{p}$ is not "bad" in the sense that $\mathbf{p} \notin B_{\ell,n}(h,K)$, any residue-vector in the ball $B_h(\mathbf{y})$ must be the residue-vector of an integer in $[a-h, a+h]$. We actually use this result in two ways in analysing our decoding algorithm in Section 5:

1. *Uniqueness of Solution:* We assume that the code is not "bad" (in the sense $\mathbf{p} \notin \mathcal{B}_{\ell,n}(h,K)$), and as explained above under this condition the vicinity interval $[a-h, a+h]$ contains all integer solutions to our problem (note that in our proof we show with an additional assumption that actually the interval $I(a,h) \subseteq [a-h, a+h]$ defined in Section 3 is the set of all solutions to our problem in $\mathcal{Z}[K]$).

2. *Correctness of Efficient Algorithm:* To make our decoding algorithm for finding the interval $I(a,h)$ efficient (poly-time), we make use of an efficient but *approximate* (non-exact) lattice algorithm for the Closest Vector Problem (CVP) to compute an integer c "close" to the vicinity

interval $[a-h, a+h]$. Ideally, we would like that c actually falls in $[a-h, a+h]$, which by the uniqueness result above is the case if the code is not "bad" in the sense $\mathbf{p} \notin \mathcal{B}_{\ell,n}(h, K)$, and if we can guarantee that $c < K$ and the residue vector of c is within distance h of $\mathbf{y}$ (namely in the ball $B_h(\mathbf{y})$). However, because of the approximate nature of the CVP algorithm, we are only able to guarantee instead that $c < \gamma \cdot K$ and that the residue-vector of c is within distance $\gamma \cdot h$ of $\mathbf{y}$ (namely in the ball $B_{\gamma h}(\mathbf{y})$), for some approximation constant $\gamma > 1$. By applying the above uniqueness result with K replaced by $\gamma \cdot K$ and h replaced by $\gamma \cdot h$, we are able to conclude that if the code is not "bad" in the stronger sense $\mathbf{p} \notin \mathcal{B}_{\ell,n}(\gamma \cdot h, \gamma \cdot K)$, then the integer c is in the interval $[a-\gamma \cdot h, a+\gamma \cdot h]$ and hence is "close" to the desired $[a-h, a+h]$ interval (note that our actual proof is slightly more complicated than the above description because of the technical fact that the CVP algorithm may output a negative c with $|c| < K$).

Summarising, we show that our algorithm succeeds to compute $I(a, h)$ under the assumption that $\mathbf{p}$ is not in a "bad" subset $\mathcal{B}_{\ell,n}(H, M)$ of $\mathcal{P}_\ell^n$, for some integers H and M. The purpose of the following lemma is to upper bound the size of $\mathcal{B}_{\ell,n}(H, M)$, so that we can upper bound the probability that our algorithm fails over a random choice of $\mathbf{p}$.

To bound the size of $\mathcal{B}_{\ell,n}(H, M)$, we note that if $\mathbf{p} \in \mathcal{B}_{\ell,n}(H, M)$, then there exist two integers a and b in $\mathcal{Z}[M]$ whose residue-vectors are both at distance less than H from a given vector $\mathbf{y}$ (that is, $d_{L,\mathbf{p}}(\mathbf{a_p}, \mathbf{y}) < H$ and $d_{L,\mathbf{p}}(\mathbf{b_p}, \mathbf{y}) < H$) but b differs from a by at least $2H$ (that is, $|b-a| \geq 2H$). Then by the triangle inequality and the linearity of the encoding function $x \to \mathbf{x_p}$ it follows that the integer $z = |b-a|$ in $\mathcal{Z}[M]$ is of magnitude at least $2H$ but has residue-vector of Lee norm less than $2H$. Therefore $\mathcal{B}_{\ell,n}(H, M)$ is a subset of $\mathcal{E}_{\ell,n}(2H, M)$, which we define as the set of "bad" $\mathbf{p}$ for which there exists an integer z in $\mathcal{Z}[M]$ which exceeds $2H$ but its residue-vector has Lee norm less than $2H$. We have reduced the problem of bounding the size of $\mathcal{B}_{\ell,n}(H, M)$ to the simpler problem of bounding the size of $\mathcal{E}_{\ell,n}(2H, M)$, which we do directly in the following lemma. It may be of independent interest.

Lemma 2. *Fix $H \in \mathcal{Z}[M]$. There exists a set $\mathcal{E}_{\ell,n}(H, M)$ of cardinality*

$$|\mathcal{E}_{\ell,n}(H, M)| \leq M \left(\frac{2H \log 2M}{\ell} \right)^n$$

such that for $\mathbf{p} = (p_1, \ldots, p_n) \in \mathcal{P}_\ell^n \backslash \mathcal{E}_{\ell,n}(H, M)$ *there is no integer* $z \in [H, M-1]$ *with* $\|\mathbf{z}_\mathbf{p}\|_{L,\mathbf{p}} < H$.

Proof. Suppose that $\mathbf{p} = (p_1, \ldots, p_n) \in \mathcal{P}_\ell^n$ is such that there exists $z \in \mathbb{Z}$ such that

$$H \le z < M \qquad \text{and} \qquad \|\mathbf{z}_\mathbf{p}\|_{L,\mathbf{p}} < H.$$

Hence, for each $i \in \{1, \ldots, n\}$, there exists $\delta_i \in \mathbb{Z}$ such $|\delta_i| < H$ and p_i divides $z + \delta_i$. It follows that $p_i \in S_z$ for all $i \in \{1, ..., n\}$, where S_z is the set of prime divisors of all integers in the interval $I(z, H) = [z-H+1, z+H-1]$. Observe that $I(z, H)$ contains less than $2H$ integers, all upper bounded by $M + H < 2M$, and we also know that $0 \notin I(z, H)$. Hence each integer in $I(z, H)$ is divisible by at most $\ell^{-1} \log 2M$ primes $p \in \mathcal{P}_\ell$, and we have $|S_z| < 2H\ell^{-1} \log 2M$. So for each possible choice of $z \in [H, M-1]$, there are less than $(2H\ell^{-1} \log 2M)^n$ "bad" choices for $\mathbf{p} = (p_1, \ldots, p_n) \in \mathcal{P}_\ell^n$.

Since there are less than M possible values for $z \in [H, M-1]$, we get desired results. □

Certainly the result of Lemma 2 depends on our upper bounds for the number of prime divisors of the integers in the interval $I(z, H)$. For example, for very small values of ℓ the results can slightly be improved if one notices that in fact the number of prime divisors $\nu(m)$ of an integer $m \ge 2$ is $O(\log m / \log(1 + \log m))$ (because $\nu(m)! \le m$). We remark that estimation of the total number of prime divisors of the integers in a given interval is classical number theoretic problem [7, 24]. Thus, for values of H and M in certain ranges much stronger bounds can be obtained, which however do not improve our final results.

5 Algorithm

We are now ready to describe our main Algorithm LeeDecode which computes the endpoints of the solution interval $I(a, h)$ defined in (2) in time polynomial in $\log p_1, \ldots, \log p_n$, $\log K$ and n. The algorithm depends on K, h, the choice of base $\mathbf{p} = (p_1, \ldots, p_n) \in \mathcal{P}^n$, the approximating vector $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{Z}^n$ and a certain real parameter $\rho > 0$ which controls the cardinality of the set of bases for which the algorithm may produce a wrong result, see Theorem 3.

To aid the understanding of our algorithm and proof we first give a rough outline of the steps of the algorithm. Given the code parameters $(K, \mathbf{p})$,

the approximation vector $\mathbf{y} = \mathbf{a_p} + \mathbf{e}$ and the error Lee norm bound h, the algorithm proceeds as follows to find endpoints of the solution interval $I(a,h) = [\alpha^*, \beta^*] \subseteq [a-h, a+h]$ of all integers $b \in \mathcal{Z}[K]$ with $d_{L,p}(\mathbf{b_p}, \mathbf{y}) < h$.

The algorithm is divided in two stages.

The first stage (Steps 1 to 4) is lattice-based and its purpose is to compute an approximation $\alpha \in \mathbb{Z}$ (from below) to the lower end-point α^* of $I(a,h)$. If the first stage is successful, then α is smaller than α^* by an error not exceeding $(\gamma+1)h$, where $\gamma = (n+1)^{1/2} \cdot 2^{\rho n \log\log(n+1)/\log(n+1)}$ is the approximation factor of the CVP lattice algorithm (with an additional polynomial factor $(n+1)^{1/2}$).

The second stage (Step 5) consists of a loop which maintains an integer (initialized to α) and increments it by some positive amount at each iteration. The purpose of this loop is to remove the error of up to $(\gamma+1)h$ of the first stage by stepping towards the smallest integer greater than α whose residue-vector is within distance h of $\mathbf{y}$, namely α^*, at which point the loop terminates. Then Step 6 computes the other endpoint β^* of $I(a,h)$, which completes the determination of $I(a,h)$.

We now give some insight into the operation of the two stages and our analysis of them in the proof of Theorem 3.

The first stage works as follows. First we construct a basis for a lattice $\mathcal{L}$ in $\mathbb{Z}^{n+1}$ (see matrix in (3)). The lattice $\mathcal{L}$ is constructed so that for each integer $z \in \mathbb{Z}$ there is a corresponding lattice vector of the form $(\mathbf{z_p}, (z/K)h)$ having its first n coordinates equal to the residue-vector $\mathbf{z_p}$ of z. Therefore the (unknown) lattice vectors corresponding to the solution integers in $I(a,h)$ are close to the (known) vector $\bar{\mathbf{y}} = (\mathbf{y}, 0)$ having its first n coordinates equal to the target vector $\mathbf{y}$ (within Lee distance h). So one may hope to recover an integer "close" to $I(a,h)$ by finding a corresponding lattice vector "close" to the known vector $\bar{\mathbf{y}}$. Accordingly, the first stage runs the CVP algorithm with approximation factor γ on the lattice $\mathcal{L}$ with target vector $\bar{\mathbf{y}}$ and recovers an integer c less than γK such that the residue-vector of c is within distance $\gamma \cdot h$ of $\mathbf{y}$. Then, assuming that $\mathbf{p}$ is not in a certain "bad" set in the sense explained in Section 4, we can apply the uniqueness Lemma 2 to conclude that indeed c is "close" to $I(a,h)$ within error at most $(\gamma+1)h$, as required. The bound on the size of the "bad" set of vectors $\mathbf{p}$ given by Lemma 2 therefore gives us a bound on the number of vectors $\mathbf{p}$ for which the first stage of our algorithm may fail, as stated in Theorem 3 below.

The loop in the second stage (Step 5) works as follows. It maintains an integer variable whose value at the start of the jth iteration is denoted by

α_j and is initialized to $\alpha_0 = \alpha$. Recall that if the first stage is successful then $\alpha_0 = \alpha$ approximates the desired α^* from below by an error less than $(\gamma+1)h$. At the jth iteration, the loop checks if α_j has reached the desired α^* (by checking if the Lee distance between the residue vector of α_j and $\mathbf{y}$ is less than h). If so, the goal is achieved and the loop terminates. If not, it is because at least one of the residues of α_j, say the ith one $\langle \alpha_j \rangle_{p_i}$, differs from the corresponding residue y_i of $\mathbf{y}$ by more than h. In that case, the loop increments α_{j+1} to the smallest integer greater than α_j such that the ith residue difference $\langle \alpha_{j+1} - y_i \rangle_{p_i}$ is restored to be less than h in magnitude—this is achieved by the "if" statements 5a and 5b. It is easy to see that this implies that $\alpha_{j+1} - y_i \equiv -(h-1) \pmod{p_i}$. For all the 'skipped' integers between α_j and α_{j+1}, the ith residue difference with y_i modulo p_i exceeds h in magnitude, so we are certain that none of these integers can be α^*. There is a potential danger that the loop can run too many times before finding α^* (that is, not a polynomial in n and $\log p_i$). However, assuming that all the primes in $\mathbf{p}$ exceed $2(\gamma+1)h$ (note the restriction $\ell \geq 2(\gamma+1)h$ in the statement of Theorem 3), the loop runs for at most n iterations and hence its running time is polynomial in n. The reason is that if the loop would run for even $n+1$ iterations, this would already imply (see below) that it 'skipped over' α^*, which is impossible because as explained above the loop cannot 'skip over' α^*. This implication arises because if the loop runs for $n+1$ iterations then $\alpha_{n+1} - y_i \equiv \alpha_k - y_i \equiv -(h-1) \pmod{p_i}$ for some i which means we have already moved up by more than $|\alpha_{n+1} - \alpha_0| \geq p_i \geq 2(\gamma+1)h$ from the original approximation α, which was guaranteed to be smaller than α^* by at most $(\gamma+1)h$.

Algorithm $\mathsf{LeeDecode}(\mathbf{p} = (p_1, \ldots, p_n), \mathbf{y} = (y_1, \ldots, y_n), K, h, \rho)$

1. Build the following $(n+1) \times (n+1)$ matrix B, whose rows form a basis for a full-rank lattice $\mathcal{L}$ in $\mathbb{Q}^{n+1}$:

$$B = \begin{pmatrix} p_1 & 0 & \ldots & 0 & 0 \\ 0 & p_2 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & p_n & 0 \\ 1 & 1 & \ldots & 1 & h/K \end{pmatrix}. \tag{3}$$

2. Define $\overline{\mathbf{y}} = (y_1, \ldots, y_n, 0) \in \mathbb{Z}^{n+1}$.

3. Run the algorithm of Lemma 1 for lattice $\mathcal{L}$ given by B and the target vector $\overline{\mathbf{y}}$, with the precision parameter $\tau = \rho/2$. Denote by $\overline{\mathbf{c}} = (c_1, \ldots, c_n, c_{n+1}) \in \mathbb{Q}^{n+1}$ the output vector returned by the algorithm, which approximates the closest vector to $\overline{\mathbf{y}}$ in the lattice $\mathcal{L}$.

4. Compute $c = (K/h)c_{n+1}$, and

$$\alpha = \max\{0, \lfloor c - (\gamma+1)h \rfloor\},$$

where $\gamma = (n+1)^{1/2} \cdot 2^{\rho n \log\log(n+1)/\log(n+1)}$.

5. While $d_{L,\mathbf{p}}(\alpha_{\mathbf{p}}, \mathbf{y}) \geq h$ repeat

   (a) If there exists $i \in \{1, \ldots, n\}$ such that $\langle \alpha - y_i \rangle_{p_i} \geq h$, then set $\alpha \leftarrow \alpha + p_i - (h-1) - \langle \alpha - y_i \rangle_{p_i}$ for the smallest such i.

   (b) Else if there exists $i \in \{1, \ldots, n\}$ such that $\langle \alpha - y_i \rangle_{p_i} \leq -h$, then set $\alpha \leftarrow \alpha - (h-1) - \langle \alpha - y_i \rangle_{p_i}$ for the smallest such i.

6. Compute

$$\beta = \min\left\{K - 1, \alpha + (h-1) - \max_{i=1,\ldots,n} \langle \alpha - y_i \rangle_{p_i}\right\}$$

and output the interval $I = [\alpha, \beta]$.

In the following statement we give sufficient conditions under which the interval I output by Algorithm LeeDecode actually consists of all solutions to our decoding problem.

Theorem 3. *Fix natural numbers K, ℓ, h, n and real $\rho > 0$ such that $n \geq 2$ and $2^\ell \geq 2(\gamma+1)h$, where:*

$$\gamma = (n+1)^{1/2} \cdot 2^{\rho n \log\log(n+1)/\log(n+1)}.$$

There exists a "bad" prime base set $\mathcal{F}_{\ell,n}(h, K, \rho) \subseteq \mathcal{P}_\ell^n$ of cardinality bounded as

$$|\mathcal{F}_{\ell,n}(h, K, \rho)| \leq (\gamma+1)K\left(\frac{2(\gamma+1)h\log\left(2(\gamma+1)K\right)}{\ell}\right)^n,$$

such that for any prime base $\mathbf{p}$ which is not "bad", that is, $\mathbf{p} \in \mathcal{P}_\ell^n \backslash \mathcal{F}_{\ell,n}(h, K, \rho)$ and any approximation $\mathbf{y} \in \mathbb{Z}_{p_1} \times \ldots \mathbb{Z}_{p_n}$ to $a \in \mathcal{Z}[K]$ with $d_{L,\mathbf{p}}(\mathbf{a}_{\mathbf{p}}, \mathbf{y}) < h$ the Algorithm LeeDecode *runs in time polynomially bounded in $\log p_1, \ldots, \log p_n$,*

$\log K$ *and* n *and, with probability exponentially close to 1, outputs an interval* I *of length* $|I| \leq 2h$ *and such that* I *consists of all* $b \in \mathcal{Z}[K]$ *satisfying* $d_{L,\mathbf{p}}(\mathbf{b_p}, \mathbf{y}) < h$ *(namely* I *coincides precisely with the interval* $I(a,h)$ *defined in (2)).*

Proof. As explained informally above, we show that the algorithm succeeds when the prime base is not in some "bad" set $\mathcal{F}_{\ell,n}(h,K,\rho) \subseteq \mathcal{P}_\ell^n$. Let us define this "bad" set in terms of the "bad" set $\mathcal{E}_{\ell,n}(H,M)$ of Lemma 2, as follows: $\mathcal{F}_{\ell,n}(h,K,\rho) = \mathcal{E}_{\ell,n}((\gamma+1)h,(\gamma+1)K)$. Thus the stated upper bound on $|\mathcal{F}_{\ell,n}(h,K,\rho)|$ is immediate from Lemma 2, and it remains to show that the algorithm indeed succeeds when $\mathbf{p}$ is not "bad" (that is, when $\mathbf{p} \in \mathcal{P}_\ell^n \backslash \mathcal{F}_{\ell,n}(h,K,\rho)$). Note that we use the 'uniqueness' statement of Lemma 2 that if $\mathbf{p}$ is not "bad' then any integer z in $\mathcal{Z}[(\gamma+1)K]$ whose residue-vector has Lee norm $\|\mathbf{z_p}\|_{L,\mathbf{p}}$ less than $(\gamma+1)h$ must be an integer smaller than $(\gamma+1)h$.

Therefore, for the rest of the proof we assume that $\mathbf{p}$ is not "bad' ($\mathbf{p} \in \mathcal{P}_\ell^n \backslash \mathcal{F}_{\ell,n}(h,K,\rho)$). First we show that this implies that the set J of solution integers to our problem forms an interval in $\mathcal{Z}[K]$. Since we have already exhibited in Section 3 an interval of solutions $I(a,h) \subseteq [a-h, a+h]$, it follows that $J = I(a,h)$. Then we show that the interval I output by our algorithm coincides with the solution interval $J = I(a,h)$.

Suppose that J is not an interval. Then, by existence of a solution, there must exist two distinct solutions a and b in $\mathcal{Z}[K]$, thus

$$d_{L,\mathbf{p}}(\mathbf{a_p}, \mathbf{y}) < h \qquad \text{and} \qquad d_{L,\mathbf{p}}(\mathbf{b_p}, \mathbf{y}) < h, \tag{4}$$

with $b > a+1$ and such that

$$d_{L,\mathbf{p}}(\mathbf{c_p}, \mathbf{y}) \geq h \tag{5}$$

for all $c \in [a+1, b-1]$. Applying (5) to $c = a+1$ we see that there exists $i \in \{1,\ldots,n\}$ such that $\langle a - y_i \rangle_{p_i} = h-1$, hence $\|c - y_i\|_{L,p_i} \geq h$ and thus also $d_{L,\mathbf{p}}(\mathbf{c_p}, \mathbf{y}) \geq h$ for all $c \in [a+1, a+1+(p_i - 2h)]$. Therefore we have that $b > a+1+(p_i-2h)$, so $z = b-a$ satisfies $z \in [p_i - 2h, K-1]$. On the other hand, using (4) and the triangle inequality, we derive the upper bound $\|\mathbf{z_p}\|_{L,\mathbf{p}} < 2h \leq (\gamma+1)h$, since $\gamma \geq 1$. So the existence of z contradicts the assumption $\mathbf{p} \in \mathcal{P}_\ell^n \backslash \mathcal{F}_{\ell,n}(h,K,\rho)$, because

$$p_i - 2h > 2^\ell - 2h \geq 2(\gamma+1)h - 2h \geq (\gamma+1)h$$

and $K \leq (\gamma+1)K$ using $\gamma \geq 1$. Hence J is an interval, as required.

Now we show that Algorithm LeeDecode finds the interval J, that is, that $I = J$.

First observe that from the assumption on the existence of at least one solution $a \in \mathcal{Z}[K]$ satisfying $\|\mathbf{e}\|_{L,\mathbf{p}} = \|\mathbf{y} - \mathbf{a_p}\|_{L,\mathbf{p}} < h$, we know that there exist modular reduction coefficients $k_i \in \mathbb{Z}$ such that $|a - y_i - k_i p_i| < h$ for all $i \in \{1, \ldots, n\}$. Consequently, the vector $\mathbf{u} = (a - k_1 p_1, \ldots, a - k_n p_n, ha/K)$, which is clearly in the lattice $\mathcal{L}$, satisfies

$$\|\mathbf{u} - \overline{\mathbf{y}}\| = \|(a - k_1 p_1 - y_1, \ldots, a - k_n p_n - y_n, ha/K)\| < (n+1)^{1/2} h,$$

(because $ha/K < h$). So

$$\min_{\mathbf{v} \in \mathcal{L}} \|\mathbf{v} - \overline{\mathbf{y}}\| < (n+1)^{1/2} h,$$

and from Lemma 1 the vector $\overline{\mathbf{c}}$, returned by the CVP algorithm of Lemma 1 in Step 3 of Algorithm LeeDecode satisfies, with probability exponentially close to 1, the inequality

$$\|\overline{\mathbf{c}} - \overline{\mathbf{y}}\| < 2^{\tau(n+1)\log\log(n+1)/\log(n+1)} (n+1)^{1/2} h < \gamma h,$$

since $\tau(n+1) < 2\tau n = \rho n$ for $n \geq 2$.

Since $\overline{\mathbf{c}} \in \mathcal{L}$, it has the form $\overline{\mathbf{c}} = (c + t_1 p_1, \ldots, c + t_n p_n, ch/K)$ for some $t_1, \ldots, t_n$ and c in $\mathbb{Z}$, so the above bound on $\|\overline{\mathbf{c}} - \overline{\mathbf{y}}\|$ implies $|c + t_i p_i - y_i| < \gamma h$ for all $i \in \{1, \ldots, n\}$ and $|ch/K| < \gamma h$. Hence the integer c (which is computed in Step 4 of Algorithm LeeDecode using $c = (K/h)c_{n+1}$) has a residue vector $\mathbf{c_p}$ satisfying $d_{L,\mathbf{p}}(\mathbf{c_p}, \mathbf{y}) < \gamma h$ while $|c| < \gamma K$. So for any solution $b \in \mathcal{Z}[K]$ satisfying $d_{L,\mathbf{p}}(\mathbf{b_p}, \mathbf{y}) < h$, if we define $z = |c - b|$ then we get from the triangle inequality, that $0 \leq z < (\gamma+1)K$ and

$$\|\mathbf{z_p}\|_{L,\mathbf{p}} = \|\mathbf{c_p} - \mathbf{b_p}\|_{L,\mathbf{p}} \leq d_{L,\mathbf{p}}(\mathbf{c_p}, \mathbf{y}) + d_{L,\mathbf{p}}(\mathbf{b_p}, \mathbf{y}) < (\gamma+1)h.$$

But since $\mathbf{p} \in \mathcal{P}_\ell^n \backslash \mathcal{F}_{\ell,n}(h, K, \rho)$, we conclude from Lemma 2 that $z = |c - b| < (\gamma+1)h$ for any solution $b \in \mathcal{Z}[K]$. This shows that if $\mathbf{p} \in \mathcal{P}_\ell^n \backslash \mathcal{F}_{\ell,n}(h, K, \rho)$ then the first stage of our algorithm (Steps 1 to 4) succeeds to compute an integer c which approximates the lower endpoint α^* of the desired solution interval J (from below) with an error at most $(\gamma+1)h$.

To proceed further we first need to establish some properties of the loop in the second stage the algorithm LeeDecode(Step 5), as follows. Let α_0 denote

the value of α before entering the loop, and for $k \geq 1$, let α_k denote the value of α at the end of the kth iteration of the loop. Let $J = [\alpha^*, \beta^*]$ be the desired interval of all $b \in \mathcal{Z}[K]$ with $d_{L,\mathbf{p}}(\mathbf{b}_\mathbf{p}, \mathbf{y}) < h$. We show by induction that for all $k \geq 1$, $\alpha_{k-1} \leq \alpha^*$, $\alpha_k > \alpha_{k-1}$ and there exists $i_k \in \{1, \ldots, n\}$ such that $\langle \alpha_k - y_{i_k} \rangle_{p_{i_k}} = -(h-1)$. For the basis case $k = 1$, we know by definition that $\alpha_0 = \max(0, \lfloor c - (\gamma+1)h \rfloor)$, and using the above bound $\alpha^* > c - (\gamma+1)h$ and $\alpha^* \geq 0$ we get $\alpha_0 \leq \alpha^*$. For the induction step, we suppose for some $k \geq 1$ that $\alpha_{k-1} \leq \alpha^*$. We assume that α_{k-1} satisfies the 'while' loop condition $d_{L,\mathbf{p}}((\alpha_{k-1})_\mathbf{p}, \mathbf{y}) \geq h$ since otherwise α_k does not exist. So there are two possible cases: either there exists $i \in \{1, \ldots, n\}$ such that $\langle \alpha_{k-1} - y_i \rangle_{p_i} \geq h$ (we let i_k denote the smallest such i), or else there must exist $i \in \{1, \ldots, n\}$ such that $\langle \alpha_{k-1} - y_i \rangle_{p_i} \leq -h$ (we let i_k denote the smallest such i). We consider these two cases in turn. In the first case, the 'if' condition in Step 5a of LeeDecode is satisfied, and consequently

$$\alpha_k = \alpha_{k-1} + p_{i_k} - (h-1) - \langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}}.$$

Using $p_{i_k} > 2^\ell \geq 2(\gamma+1)h$ and $\langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}} \leq p_{i_k}/2$ we have $p_{i_k} - (h-1) - \langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}} > 0$ and hence $\alpha_k > \alpha_{k-1}$. Also,

$$\langle \alpha_k - y_{i_k} \rangle_{p_{i_k}} = \left\langle \alpha_{k-1} - y_{i_k} + p_{i_k} - (h-1) - \langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}} \right\rangle_{p_{i_k}} = -(h-1),$$

while there are no solutions in $[\alpha_{k-1}, \alpha_k - 1]$ because for any integer $a \in [\alpha_{k-1}, \alpha_k - 1]$ either

$$h \leq \langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}} \leq \langle a - y_{i_k} \rangle_{p_{i_k}} \leq p_{i_k}/2$$

or

$$-p_{i_k}/2 < \langle a - y_{i_k} \rangle_{p_{i_k}} \leq -h.$$

Using the induction hypothesis $\alpha_{k-1} \leq \alpha^*$ we thus obtain $\alpha_k \leq \alpha^*$, as required. It remains to consider the second case, when the 'if' condition in Step 5b of LeeDecode is satisfied. In this case

$$\alpha_k = \alpha_{k-1} - (h-1) - \langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}}.$$

Using $\langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}} \leq -h$ we immediately have $\alpha_k > \alpha_{k-1}$. Also,

$$\langle \alpha_k - y_{i_k} \rangle_{p_{i_k}} = \left\langle \alpha_{k-1} - y_{i_k} - (h-1) - \langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}} \right\rangle_{p_{i_k}} = -(h-1),$$

while there are no solutions in $[\alpha_{k-1}, \alpha_k - 1]$ because

$$\langle \alpha_{k-1} - y_{i_k} \rangle_{p_{i_k}} \leq \langle a - y_{i_k} \rangle_{p_{i_k}} \leq -h$$

for all $a \in [\alpha_{k-1}, \alpha_k - 1]$, and again using the induction hypothesis $\alpha_{k-1} \leq \alpha^*$ we get $\alpha_k \leq \alpha^*$, as required.

We are now ready to show that the interval $I = [\alpha, \beta]$ output by Algorithm LeeDecode coincides with the desired interval $J = [\alpha^*, \beta^*]$. Suppose first that the loop in Step 5 terminates after N iterations. Using the property $\alpha_k \leq \alpha^*$ for all k we have in particular that $\alpha_N \leq \alpha^*$, while $\alpha_N \geq \alpha^*$ by the loop termination condition. So $\alpha = \alpha_N = \alpha^*$. The upper end point β^* of J is the smallest integer greater or equal to α^* such that $\langle \beta^* - y_i \rangle_{p_i} = h - 1$ for some $i \in \{1, \ldots, n\}$, (or $\beta^* = K - 1$), namely

$$\beta^* = \min \left\{ K - 1, \alpha^* + (h-1) - \max_{i=1,\ldots,n} \langle \alpha^* - y_i \rangle_{p_i} \right\} = \beta,$$

as defined in Step 6 of Algorithm LeeDecode. This shows that LeeDecode outputs J if the loop terminates. We now claim that the loop terminates after at most n iterations. Suppose, towards a contradiction, that the loop runs for $n+1$ or more iterations. Recall that for all $k \geq 1$, $\langle \alpha_k - y_{i_k} \rangle_{p_{i_k}} = -(h-1)$ for some $i_k \in \{1, \ldots, n\}$, but $\alpha_k > \alpha_i$ when $k > i$ because $\alpha_k > \alpha_{k-1}$ for all $k \geq 1$. By the Pigeonhole Principle, there must exist k and m such that $1 \leq k < m \leq n+1$ and $i_k = i_m = i$ for some $i \in \{1, \ldots, n\}$. This means $\langle \alpha_m - y_i \rangle_{p_i} = \langle \alpha_k - y_i \rangle_{p_i} = -(h-1)$ so $\alpha_m \equiv \alpha_k \pmod{p_i}$ but $\alpha_m > \alpha_k$. Hence $\alpha_m - \alpha_k \geq p_i \geq 2^\ell + 1 \geq 2(\gamma+1)h + 1$, so, using $\alpha_k \geq \alpha_0$ and $\alpha_0 = \max(0, \lfloor c - (\gamma+1)h \rfloor)$, we have that

$$\alpha_m \geq \alpha_k + 2(\gamma+1)h + 1 \geq \alpha_0 + 2(\gamma+1)h + 1 > c + (\gamma+1)h.$$

But since $\alpha_m \leq \alpha^*$, there are no solutions smaller than α_m, which contradicts the existence of a solution, using the earlier result that $|c - b| < (\gamma+1)h$ for any solution b. This proves that the loop terminates after at most n iterations.

The running time claim is readily obtained from Lemma 1 and the fact that all other steps in the algorithm take time polynomial in n and the bit length of the input parameters K and $p_1, \ldots, p_n$. □

It is natural to assume that the primes $p_1, \ldots, p_n$ are randomly chosen from the set $\mathcal{S}_\ell$ of primes in the interval $[2^\ell, 2^{\ell+1}]$. It is known [25] that a

lower bound on the size of this set is

$$|\mathcal{S}_\ell| > 0.6\frac{2^\ell}{\ell \ln 2} > \frac{2^{\ell-1}}{\ell}$$

for all $\ell \geq 5$. Plugging this in the probability bound of Theorem 3 and simplifying, we find that our Algorithm LeeDecode succeeds with probability at least $1-\delta$ over the random choice of $(p_1, \ldots, p_n) \in \mathcal{S}_\ell^n$ in recovering an interval containing all solutions to the decoding problem whenever

$$(\gamma+1)K\left(\frac{2(\gamma+1)h\log\left(2(\gamma+1)K\right)}{\ell}\right)^n \leq \delta\left(\frac{2^{\ell-1}}{\ell}\right)^n$$

or

$$2^{\ell-1} \geq \left(\delta^{-1}(\gamma+1)K\right)^{1/n} 2(\gamma+1)h\log\left(2(\gamma+1)K\right).$$

This condition can be satisfied with some

$$\ell = \frac{\log \delta^{-1}K}{n} + \log h + \log\log K + O\left(\frac{n\log\log(n+1)}{\log(n+1)}\right).$$

For example, if we set $\log h = \ell - \left\lfloor \ell^{\vartheta_h} \right\rfloor$, $n = \left\lfloor 3\ell^{\vartheta_n} \right\rfloor$, $\log K = \ell^{1+\vartheta_K}$, and $\delta = 1/K$ for some constants ϑ_h, ϑ_n and ϑ_K (this means that we are given only approximately ℓ^{ϑ_h} most-significant bits of each residue, which has length at least ℓ bits). Then, fixing $\vartheta_K > 0$, we can satisfy the above sufficient success condition for sufficiently large ℓ with $\vartheta_h = 1/2 + \vartheta_K/2$, achieved by setting $\vartheta_n = 1/2 + \vartheta_K/2$. One of these possible choices could be $h = 2^{\ell - \left\lfloor \ell^{3/4} \right\rfloor}$ (which means that we are given very rough approximations), $n = \left\lfloor 3\ell^{3/4} \right\rfloor$, $K = \left\lfloor 2^{l^{3/2}} \right\rfloor$ and $\delta = K^{-1}$. This means that for each ℓ-bit prime p_i we are given about

$$\log\left(p_i/h\right) \sim l^{3/4} \sim \log^{1/2} K$$

most significant bits of $a \pmod{p_i}$, $i = 1, \ldots, n$.

References

[1] M. Ajtai, R. Kumar and D. Sivakumar, 'A sieve algorithm for the shortest lattice vector problem', *Proc. 33rd ACM Symp. on Theory of Comput.*, Crete, Greece, July 6-8, 2001, 601–610.

[2] S. Ar, R. Lipton, R. Rubinfeld and M. Sudan, 'Reconstructing algebraic functions from erroneous data', *SIAM J. Comput.*, **28** (1999), 487–510.

[3] D. Boneh, 'Finding smooth integers in short intervals using CRT decoding', *J. Comp. and Syst. Sci.*, **64** (2002), 768–784.

[4] D. Boneh and R. Venkatesan, 'Hardness of computing the most significant bits of secret keys in Diffie–Hellman and related schemes', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1109** (1996), 129–142.

[5] D. Boneh and R. Venkatesan, 'Rounding in lattices and its cryptographic applications', *Proc. 8th Annual ACM-SIAM Symp. on Discr. Algorithms*, ACM, 1997, 675–681.

[6] C. Ding, D. Pei and A. Salomaa, *Chinese Remainder Theorem: Applications in computing, coding, cryptography*, World Scientific, Singapore, 1996.

[7] P. Erdös and J. Turk, 'Products of integers in short intervals', *Acta Arith.*, **44** (1984), 147–174.

[8] O. Goldreich, D. Ron, and M. Sudan, 'Chinese remaindering with errors', *IEEE Transactions on Information Theory*, **46** (July 2000), 1330–1338.

[9] O. Goldreich, R. Rubinfeld and M. Sudan, 'Learning polynomials with queries: the highly noisy case', *Electronic Colloq. on Comp. Compl.*, Univ. of Trier, **TR1998-060** (1998), 1–34.

[10] M. Grötschel, L. Lovász and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, Berlin, 1993.

[11] V. Guruswami, A. Sahai and M. Sudan, '"Soft-decision" decoding of Chinese remainder codes', *Proc. 41st IEEE Symp. on Found. of Comp. Sci.*, Redondo Beach, California, 2000, 159–168.

[12] V. Guruswami and M. Sudan, 'Improved decoding of Reed-Solomon codes and algebraic-geometric codes', *IEEE Transactions on Information Theory*, **45** (1999), 1757–1767.

[13] T. Hoholdt and R. R. Nielsen, 'Decoding Hermitian codes with Sudan's algorithm', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1719** (1999), 260–270.

[14] A. Joux and J. Stern, 'Lattice reduction: A toolbox for the cryptanalyst', *J. Cryptology*, **11** (1998), 161–185.

[15] R. Kannan, 'Algorithmic geometry of numbers', *Annual Review of Comp. Sci.*, **2** (1987), 231–267.

[16] A. Kiayias and M. Yung, 'Secure games with polynomial expressions', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2076** (2001), 939–950.

[17] A. Kiayias and M. Yung, 'Polynomial reconstruction based cryptography', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2259** (2001), 129–133.

[18] M. Kiwi, F. Magniez and M. Santha, 'Exact and approximate testing/correcting of algebraic functions: A survey', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2292** (2002), 30–83.

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

[20] P. Q. Nguyen and I. E. Shparlinski, 'The insecurity of the digital signature algorithm with partially known nonces', *J. Cryptology*, **15** (2002), 151–176.

[21] P. Q. Nguyen and J. Stern, 'Lattice reduction in cryptology: An update', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1838** (2000), 85–112.

[22] P. Q. Nguyen and J. Stern, 'The two faces of lattices in cryptology', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2146** (2001), 146–180.

[23] R. R. Nielsen, 'A class of Sudan-decodable codes', *IEEE Transactions on Information Theory*, **46** (2000), 1564–1572.

[24] K. Ramachandra, T. N. Shorey and R. Tijdeman, 'On Grimm's problem relating to factorisation of a block of consecutive integers', *J. Reine Angew. Math.*, **273** (1975), 109–124.

[25] J.B. Rosser and L. Schoenfeld, 'Approximate Formulas for some functions of Prime Numbers', *Illinois. J. Math.*, **6** (1962), 64–94.

[26] S. Sakata, 'On fast interpolation method for Guruswami-Sudan list decoding of one-point algebraic-geometry codes', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2227** (2001), 36–45.

[27] C. P. Schnorr, 'A hierarchy of polynomial time basis reduction algorithms', *Theor. Comp. Sci.*, **53** (1987), 201–224.

[28] M.A. Shokrollahi and H. Wasserman, 'List decoding of algebraic-geometric codes, *IEEE Transactions on Information Theory*, **45** (1999), 432–437.

[29] I. E. Shparlinski, 'Sparse polynomial approximation in finite fields', *Proc. 33rd ACM Symp. on Theory of Comput.*, Crete, Greece, July 6-8, 2001, 209–215.

[30] I. E. Shparlinski, 'Playing "Hide-and-Seek" in finite fields: Hidden number problem and its applications', *Proc. 7th Spanish Meeting on Cryptology and Information Security, Vol.1*, Univ. of Oviedo, 2002, 49–72.

[31] M. Sudan, 'Decoding of Reed Solomon codes beyond the error-correction bound', *J. Complexity*, **13** (1997), 180–193.

[32] M. Sudan, 'Ideal error-correcting codes: Unifying algebraic and number-theoretic algorithms', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2227** (2001), 36–45.