Chinese Remaindering with Multiplicative Noise

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 23, 2005

Abstract

We use lattice reduction to obtain a polynomial time algorithm for recovering an integer (up to a multiple) given multiples of its residues modulo sufficiently many primes, when the multipliers are unknown but small.

1 Introduction

For integers s and $m \geq 1$ we denote by $\lfloor s \rfloor_m$ the remainder of s on division by m. For an integer $A \geq 1$ we denote by $\mathcal{Z}[A]$ the set of integers in the interval $[0, A-1]$.

We consider the following problem. For an integer $a \in \mathcal{Z}[A]$ (for some positive integer A), we are given n multiples $y_i = \lfloor r_i \cdot a \rfloor_{p_i}$ of the residues of a modulo known primes $p_1, \ldots, p_n$, where the multiplier integers $r_i \neq 0$ are unknown but small in absolute value, so that $|r_i| < p_i^\alpha$ for all $i = 1, \ldots, n$, for some $\alpha < 1$. Our goal is to recover the hidden integer a from $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathbf{p} = (p_1, \ldots, p_n)$.

Note that in general the integer a cannot be recovered uniquely from the given information. In particular, the vector $\mathbf{y}$ corresponding to the hidden integer a with multiplier vector $\mathbf{r} = (r_1, \ldots, r_n)$ also corresponds to the integer $\widetilde{a} = d \cdot a$ with multiplier vector $\widetilde{\mathbf{r}} = (r_1/d, \ldots, r_n/d)$, for any common divisor d of $r_1, \ldots, r_n$. Thus we are content with recovering the largest such integer, namely $D \cdot a$ where $D = \gcd(r_1, \ldots, r_n)$, rather than recovering a itself.

We give a polynomial time algorithm for the above "noisy" Chinese remaindering problem and show that it succeeds to recover $D \cdot a$ with overwhelming probability over a random choice of sufficiently many large primes whenever $\alpha < (1 - \varepsilon)/(n + 1)$ for any $\varepsilon > 0$. Here, as in many other works of similar spirit, we use lattice algorithms. It is interesting to remark that the above threshold on the level of "noise" corresponds to that of [3] where a similar problem is considered for polynomial evaluation with multiplicative noise.

Clearly if the primes $p_1, \ldots, p_n$ can be choosen in an advance, then we can simply take any n primes with $p_i \le 2^n$ which in our settings with $\alpha < 1/n$ immediately implies that $|r_i| < 2$, that is, $r_i = \pm 1$. Thus we know $a^2 \pmod{p_i}$, $i = 1, \ldots, n$. Provided $A^2 \le p_1 \ldots p_n$ this is enough to find a^2 and therefore a. However, our algorithm works in a much more general situation.

We remark that several more variants of the "noisy" Chinese remaindering problem has been considered in the literature, see [2, 4, 6, 13]. In particular, the case of "additive noise" has been considered in [13] with a much more genereous bound on the level of noise. The authors of [13] left it as an open problem to find a cryptographic application for their algorithm. Recently, an application of this "additive noise" algorithm to changeable-threshold secret-sharing schemes was given in [14]. Although the "multiplicative noise" algorithm in this paper is not well suited for the application in [14] due to a weaker algorithmic result, we hope that the hardness of this problem may find application in cryptography (although its hardness needs further study).

Throughout the paper $\log z$ denotes the binary logarithm of $z > 0$.

2 Lattices

Here we collect several well-known facts which form the background of our algorithm.

We review several related results and definitions on lattices which can be found in [5]. For more details and more recent references, we recommend to consult the brilliant surveys of Nguyen and Stern [9, 10].

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

$$\mathcal{L} = \{\mathbf{z}\colon \mathbf{z} = c_1\mathbf{b}_1 + \ldots + c_s\mathbf{b}_s,\ c_1, \ldots, c_s \in \mathbb{Z}\}$$

is called an *s-dimensional lattice* with *basis* $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$. If $s = r$, the lattice L is of *full rank.*

To each lattice $\mathcal{L}$ one can naturally associate its *volume*

$$\mathrm{vol}\,(\mathcal{L}) = \Big(\det\left(\langle \mathbf{b}_i, \mathbf{b}_j\rangle\right)_{i,j=1}^{s}\Big)^{1/2},$$

where $\langle \mathbf{a}, \mathbf{b}\rangle$ denotes the inner product, which does not depend on the choice of the basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$.

For a vector $\mathbf{u}$, let $\|\mathbf{u}\|$ denote its *Euclidean norm.*

Given a lattice $\mathcal{L}$, the problem of finding a shortest vector in a lattice which is known as the *shortest vector problem*, or **SVP**. For a lattice $\mathcal{L}$, with a given basis, we say that a certain lattice algorithm is *polynomial time* if its running time is bounded by a polynomials in s and the total bit length of components of the basis vectors of the lattice, which are assumed to be given as rational numbers. Unfortunately, there are several indications that the **SVP** is **NP**-complete (when the dimension grows), see [9, 10]. However, for a relaxed task of finding a short vector, $\mathbf{v} \in \mathcal{L}$ satisfying

$$\|\mathbf{v}\| \le \gamma_s \min\{\|\mathbf{z}\| \ :\ \mathbf{z} \in \mathcal{L} \setminus \{0\}\}$$

with some *approximation factor* γ_s may be more feasible.

Indeed, the celebrated *LLL algorithm* of Lenstra, Lenstra and Lovász [8] provides a desirable solution with the approximation factor $\gamma_s = 2^{s/2}$ thus producing, in deterministic polynomial time a vector $\mathbf{v} \in \mathcal{L}$ satisfying

$$\|\mathbf{v}\| \le 2^{s/2} \min\{\|\mathbf{z}\| \ :\ \mathbf{z} \in \mathcal{L} \setminus \{0\}\}.$$

Later developments of Schnorr [12] and quite recently by Ajtai, Kumar, and Sivakumar [1] lead to some (rather slight) improvements of our results.

3 Algorithm

The algorithm is given as input $(\mathbf{y}, \mathbf{p}, A)$, where $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{Z}^n$ and $\mathbf{p} = (p_1, \ldots, p_n) \in \mathcal{P}_\ell^n$, where $y_i = \lfloor r_i \cdot a \rfloor_{p_i}$ for $i = 1, \ldots, n$, $a \in \mathcal{Z}[A]$, and $\mathcal{P}_\ell$ is a set of primes which exceed 2^ℓ for some integer length parameter $\ell \geq 1$. We also assume that we are given an **SVP** algorithm with approximation factor γ_n.

The algorithm is based on the observation that $(R/r_i) \cdot y_i \equiv R \cdot a \pmod{p_i}$ for all $i = 1, \ldots, n$, where $R = \mathrm{lcm}(r_1, \ldots, r_n)$, and hence

$$\sum_{i=1}^{n} \lambda_i \cdot (R/r_i) \cdot y_i \equiv R \cdot a \pmod{P},$$

where $P = p_1 \ldots p_n$, and λ_i denotes the ith Chinese Remainder coefficient satisfying $\lambda_i \equiv 1 \pmod{p_i}$ and $\lambda_i \equiv 0 \pmod{p_j}$ for all $j \neq i$. This observation suggests to set up a lattice to search for the "small" linear combination coefficients (R/r_i) of $\lambda_i y_i$ which give a "small" residue $R \cdot a$ modulo P (the residue is "small" as long as P is sufficiently large compared to $R \cdot a$).

The Algorithm MultNoise-CRT proceeds as follows:

Algorithm $\mathsf{MultNoise} - \mathsf{CRT}(\mathbf{y} = (y_1, \ldots, y_n), \mathbf{p} = (p_1, \ldots, p_n), A, \gamma_n)$

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

$$B = \begin{pmatrix} 1 & 0 & \ldots & 0 & \lambda_1 \cdot y_1/A \\ 0 & 1 & \ldots & 0 & \lambda_2 \cdot y_2/A \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & \lambda_n \cdot y_n/A \\ 0 & 0 & \ldots & 0 & P/A \end{pmatrix}. \quad (1)$$

2. Run a polynomial-time **SVP** algorithm with approximation factor γ_n on input B. Denote by $\mathbf{b} = (b_1, \ldots, b_{n+1}) \in \mathcal{L}$ the vector returned by this algorithm, which approximates the shortest vector in $\mathcal{L}$.

3. Compute $z_i = A \cdot b_{n+1}/b_i$ for $i = 1, \ldots, n$ (if any z_i is not an integer then the algorithm fails) and output $\widetilde{a} = \gcd(z_1, \ldots, z_n)$ as an estimate for the desired integer $\gcd(r_1, \ldots, r_n) \cdot a$.

4 Analysis

4.1 Heuristic and Necessary Conditions for Algorithm Success

As explained above, our algorithm works by first building a lattice $\mathcal{L}$ which contains a "short" non-zero vector $\mathbf{b}^*$ (such that the solution to our problem can be easily recovered from $\mathbf{b}^*$), and then searching the lattice for (an approximation to) the shortest non-zero vector, hoping that $\mathbf{b}^*$ is the shortest non-zero vector in $\mathcal{L}$. Thus, a necessary condition for our algorithm to succeed by recovering the vector $\mathbf{b}^*$ (or an integral multiple thereof) is that $\mathbf{b}^*$ is the shortest non-zero vector in $\mathcal{L}$.

From the discussion in the previous section, one can see that $\mathbf{b}^* = (R/r_1, \ldots, R/r_n, R \cdot a/A)$, where $R = |\mathrm{lcm}(r_1, \ldots, r_n)|$. Hence, in the worst case, the Euclidean length $\|\mathbf{b}^*\|$ of $\mathbf{b}^*$ is approximately equal to R. On the other hand, we know by the Minkowski theorem in the geometry of numbers [5] that the Euclidean length $\lambda_1(\mathcal{L})$ of the shortest non-zero vector in any lattice $\mathcal{L}$ of dimension s is at most $s^{1/2} \det(\mathcal{L})^{1/s}$. For our lattice we have $s = n+1$ and $\det(\mathcal{L}) = P/A$, so $\lambda_1(\mathcal{L}) \leq (n+1)^{1/2}(P/A)^{1/(n+1)}$. Thus a necessary condition for having $\|\mathbf{b}^*\| = \lambda_1(\mathcal{L})$ is that $R \leq (n+1)^{1/2}(P/A)^{1/(n+1)}$. Putting

$$\alpha = \frac{\log R}{\log P} \qquad \text{and} \qquad \beta = \frac{\log A}{\log P},$$

we get the necessary condition

$$\alpha \leq \frac{1 - \beta + (1 + 1/n)\log(n+1)/(2\ell)}{n+1}.$$

Therefore, the vector $\mathbf{b}^*$ is not be the shortest vector in $\mathcal{L}$ (and hence our algorithm fails) if the bit-length of the noise integers r_i exceeds approximately a fraction $(1-\beta)/(n+1)$ of the (average) bit-length of the prime moduli $p_1, \ldots, p_n$. Heuristically, we expect that this necessary condition for success is also approximately sufficient, that is, that our algorithm succeeds as soon as $\alpha \leq (1 - \beta - \varepsilon)/(n+1)$ for some small $\varepsilon > 0$, because in this case we expect that $\mathbf{b}^*$ will be the shortest non-zero vector in $\mathcal{L}$. In the next section, we rigorously prove that this heuristic is correct with high probability when we choose the prime moduli randomly from a sufficiently large set.

4.2 Sufficient Condition for Algorithm Success

We now give a rigorous sufficient success condition for our algorithm which quite closely matches the heuristic sufficient condition (and the necessary condition) discussed above.

Theorem. *For integer $n \geq 2, \ell, \alpha, A$ and real $c > 0$ and $0 < \delta < 1$, such that $A \geq c^n$, fix a non-zero integer $a \in \mathcal{Z}[A]$ and n non-zero integer multipliers $(r_1, \ldots, r_n)$ with $|r_i| < 2^{\alpha\ell}$ for all $i = 1, \ldots, n$. Let $\mathcal{P}_\ell$ denote a set of primes all exceeding 2^ℓ with $\#\mathcal{P}_\ell \geq c2^\ell/\ell$. If for*

$$\beta = \frac{\log A}{\ell n}, \qquad \eta = \left\lceil (n+1)^{1/2} \right\rceil \gamma_n, \qquad \rho = \log(2A\eta R^2),$$

the following condition holds

$$\alpha \leq \frac{1 - \beta - \varepsilon}{n+1}$$

with

$$\varepsilon = \frac{\log\left(\rho c^{-1}(3\eta)^{1+1/n}\delta^{-1/n}\right)}{\ell},$$

then, on input $(\mathbf{y} = (y_1, \ldots, y_n),\ \mathbf{p} = (p_1, \ldots, p_n), A, \gamma_n)$, where $y_i = \lfloor r_i a \rfloor_{p_i}$ for $i = 1, \ldots, n$, $\mathbf{p} \in \mathcal{P}_\ell^n$, Algorithm MultNoise-CRT *computes Da where $D = \gcd(r_1, \ldots, r_n)$, in time polynomial in $\log(p_1 \ldots p_n)$ for at least a fraction $1 - \delta$ of prime bases $\mathbf{p} \in \mathcal{P}_\ell^n$.*

Proof. We call a vector $\mathbf{b} = (b_1, \ldots, b_{n+1}) \in \mathcal{L}$ *good* if $Ab_{n+1} = b_i r_i a$ for all $i = 1, \ldots, n$. A vector which is not good is called *bad*.

First, we observe that if the **SVP** algorithm returns a good vector $\mathbf{b}$, then our algorithm recovers $z_i = r_i a$ for all $i = 1, \ldots, n$ and hence $\widetilde{a} = \gcd(r_1, \ldots, r_n)a$, so our algorithm succeeds.

Now we observe that by construction, the lattice $\mathcal{L}$ contains the "short" good vector

$$\mathbf{b}^* = (R/r_1, \ldots, R/r_n, Ra/A),$$

whose components are all less than $R = \mathrm{lcm}(r_1, \ldots, r_n)$, and whose Euclidean length is therefore less than $(n+1)^{1/2}R$. It follows that the lattice vector $\mathbf{b}$ returned by the **SVP** algorithm has length less than $\gamma_n(n+1)^{1/2}R \leq \eta R$. So a sufficient condition for success of our algorithm is that the lattice $\mathcal{L}$ does not contain any bad vectors of length less than $\eta \cdot R$, since this implies that $\mathbf{b}$ must be good.

For fixed a and $\mathbf{r}$, we now upper bound the number N of "bad" prime bases $\mathbf{p} \in \mathcal{P}_\ell^n$ for which the lattice $\mathcal{L}$ contains a bad vector of length less than $\eta \cdot R$.

We call a bad vector $\mathbf{b} = (b_1, \ldots, b_{n+1})$ a *fully bad* vector if $A \cdot b_{n+1} \neq b_i \cdot r_i \cdot a$ for all $i = 1, \ldots, n$ and *partially bad* otherwise.

Let $\mathbf{b}$ denote a fully bad vector of length less than $\eta \cdot R$. The number of bases $\mathbf{p}$ such that $\mathcal{L}$ contains $\mathbf{b}$ can be bounded by observing that by construction of $\mathcal{L}$ we have for any lattice vector $\mathbf{b}$ that

$$Ab_{n+1} \equiv b_i r_i a \pmod{p_i}, \qquad i = 1, \ldots, n.$$

On the other hand, since $\mathbf{b}$ is fully bad we know that $Ab_{n+1} \neq b_i r_i a$ for all $i = 1, \ldots, n$. So p_i divides the non-zero integer $u_i = Ab_{n+1} - b_i r_i a$ for all $i = 1, \ldots, n$. Since

$$|u_i| < |Ab_{n+1}| + |b_i r_i A| \leq A\eta R + A\eta R r_i \leq 2A\eta R^2,$$

we know that p_i must be one of less than $\log(2A\eta R^2)/\ell$ prime factors of u_i in $\mathcal{P}_\ell$, for each $i = 1, \ldots, n$. Thus there are less than $(\log(2A\eta R^2)/\ell)^n$ bases $\mathbf{p}$ for which $\mathcal{L}$ contains $\mathbf{b}$, and since the number of fully bad vectors of length less than ηR is at most $(2\eta R)^n 2A\eta R$, we conclude that that there are

$$N_F < 2A\eta R(2\eta R \log(2A\eta R^2)/\ell)^n$$

bases $\mathbf{p}$ for which $\mathcal{L}$ contains a fully bad vector of length less than ηR.

Now we consider the case of a partially bad vector $\mathbf{b}$ of length less than ηR. We claim in this case, that if the conditions

$$2\eta R \cdot \max_{i=1,\ldots,n} |r_i| \leq 2^\ell \tag{2}$$

and

$$a \not\equiv 0 \pmod{p_i}, \qquad \text{for all } i = 1, \ldots, n \tag{3}$$

hold, then $\mathcal{L}$ does not contain $\mathbf{b}$. To establish this claim, we suppose, towards a contradiction, that $\mathcal{L}$ contains a partially bad vector $\mathbf{b}$ of length less than ηR. Because $\mathbf{b}$ is bad and is in $\mathcal{L}$, we know there exists $j \in \{1, \ldots, n\}$ such that

$$Ab_{n+1} \neq b_j r_j a \text{ and } Ab_{n+1} \equiv b_j r_j a \pmod{p_j}. \tag{4}$$

On the other hand, because $\mathbf{b}$ is partially bad, we know there also exists $i \in \{1, \ldots, n\}$ such that

$$Ab_{n+1} = b_i r_i a. \tag{5}$$

It follows from (3), (4), and (5) that $b_i r_i - b_j r_j = k \cdot p_j$ for some non-zero integer k, and therefore

$$|b_j| = \frac{|b_i r_i - k p_j|}{|r_j|} \geq \frac{1}{|r_j|}\left(\min_{\nu=1,\ldots,n} p_\nu - \eta R|r_i|\right), \tag{6}$$

using the fact that $k \neq 0$ and $|b_i| < \eta R$. But the condition (2) implies that the right-hand side of (6) is lower bounded as

$$\frac{1}{|r_j|}\left(\min_{\nu=1,\ldots,n} p_\nu - \eta R|r_i|\right) \geq \frac{1}{|r_j|}\left(\eta R \max_{\nu=1,\ldots,n} |r_\nu|\right) \geq \eta R,$$

and therefore (6) leads to a contradiction with our assumption that the length of $\mathbf{b}$ is less than ηR, as required to prove our claim above.

We now show that both conditions (2) and (3) hold for all except at most $(\log A/\ell)^n$ bases $\mathbf{p}$ in $\mathcal{P}_\ell^n$. Since $\max_i |r_i| < 2^{\alpha\ell}$, the condition (2) is implied by the condition

$$\alpha \leq \frac{1 - \ell^{-1}\log(2\eta)}{n+1}.$$

But this latter condition follows from the assumption of the theorem that

$$\alpha \leq \frac{1 - (\beta + \varepsilon)}{n+1}$$

and

$$\beta = \frac{\log A}{\ell n} \geq \frac{\log c}{\ell}$$

since we have

$$\beta + \varepsilon \geq \frac{\log(\rho(3\eta)^{1+1/n}\delta^{-1/n})}{\ell} \geq \frac{\log(2\eta)}{\ell}$$

using $\rho\delta^{-1/n} \geq 1$. Thus condition (2) is implied by the theorem hypotheses. Since $0 < a < A$, we know that a has at most $\log A/\ell$ prime factors in $\mathcal{P}_\ell$ and therefore condition (3) also holds unless $\mathbf{p}$ contains one of those primes. We conclude that, for all except

$$N_P < (\log A/\ell)^n$$

bases $\mathbf{p}$ in $\mathcal{P}_\ell^n$, both conditions (2) and (3) hold, and thus $\mathcal{L}$ contains no partially bad vectors of length less than ηR.

Therefore, there are

$$N = N_F + N_P < 3A\eta R(2\eta R\log(2A\eta R^2)/\ell)^n$$

bases $\mathbf{p}$ in $\mathcal{P}_\ell^n$ for which our algorithm may fail. It follows that for any $0 < \tilde{\delta} < 1$, the fraction of bases $\mathbf{p}$ in $\mathcal{P}_\ell^n$ for which our algorithm may fail is at most $\tilde{\delta}$ if the following condition is satisfied:

$$\frac{N}{\#\mathcal{P}_\ell^n} < \frac{3A\eta R(2\eta R\log(2A\eta R^2)/\ell)^n}{(c2^\ell/\ell)^n} \le \tilde{\delta}.$$

Plugging in $A = 2^{\beta\ell n}$, $R < 2^{\alpha\ell n}$, and defining $\rho = \log(2A\eta R^2)$, we find that the above condition is satisfied if

$$(3\eta)^{n+1}(\rho c^{-1})^n 2^{(\beta+\alpha\cdot(n+1)-1)\ell n} \le \tilde{\delta},$$

which is equivalent to condition

$$\alpha \le \frac{1-\beta-\varepsilon}{n+1},$$

where

$$\varepsilon = \frac{\log\left(\rho c^{-1}(3\eta)^{1+1/n}\tilde{\delta}^{-1/n}\right)}{\ell}.$$

By the theorem hypothesis this latter condition is satisfied with $\tilde{\delta} = \delta$, meaning that our algorithm fails for at most a fraction δ of bases $\mathbf{p}$ in $\mathcal{P}_\ell^n$, as claimed. This completes the proof of the theorem. $\square$

5 Remarks

Suppose we take for $\mathcal{P}_\ell$ the set of primes in the interval $[2^\ell, 2^{\ell+1}]$. It is known [11] that a lower bound on the size of this set is $\#\mathcal{P}_\ell \ge 2^{\ell-1}/\ell$ for all $\ell \ge 5$. In this case, our result applies with $c = 1/2$.

For the polynomial-time LLL **SVP** algorithm [8] we have $\log\eta = O(n)$ and $\log\rho = O(\log(n\ell))$ so in this case our algorithm succeeds for at least a fraction $1-\delta$ of $\mathbf{p}$ in $\mathcal{P}_\ell^n$ whenever

$$\alpha \le \frac{1-\beta-\varepsilon}{n+1}$$

for some positive ε with

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

Finally, we remark that although the condition $\alpha \le (1-\beta-\varepsilon)(n+1)$ for some small $\varepsilon > 0$ for Algorithm MultNoise-CRT is essentially optimal (in the sense that, as shown in Section 4.1, Algorithm MultNoise-CRT does not succeed to recover $\mathbf{b}^*$ when $\alpha > (1-\beta+\delta)/(n+1)$ for small $\delta > 0$), it remains an open problem whether there exist better polynomial-time algorithms for our multiplicative noise Chinese remaindering problem, which succeed even under "noisier" conditions, namely $\alpha > (1-\beta)/(n+1)$. In particular, a simple heuristic argument suggests that the solution to our problem remains unique as long as $\alpha < 1-\beta-\varepsilon$ for some small $\varepsilon > 0$. If our problem is computationally hard for

$$\frac{1-\beta}{n+1} < \alpha < 1-\beta-\varepsilon,$$

it may be possible to exploit it as the basis for the security of efficient cryptographic constructions. Finding such cryptographic applications of our problem (besides the generic uses of one-way functions) is another interesting research problem.

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.*, ACM, 2001, 601–610.

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

[3] J. von zur Gathen and I. E. Shparlinski, 'Polynomial interpolation from multiples', *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2004, 1125–1130.

[4] O. Goldreich, D. Ron and M. Sudan, 'Chinese remainding with errors', *IEEE Trans. Inform. Theory*, **46** (2000), 1330–1338.

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

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

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

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

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

[10] 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.

[11] J. B. Rosser and L. Schoenfeld, 'Approximate formulas for some functions of prime numbers', *Illinois. J. Math.*, **6** (1962), 64–94.

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

[13] I. E. Shparlinski and R. Steinfeld, 'Noisy Chinese remaindering in the Lee norm', *J. Compl.*, **20** (2004), 423–437.

[14] R. Steinfeld, J. Pieprzyk and H. Wang, 'Lattice-based threshold-changeability for standard CRT secret-sharing schemes', *Finite Fields and Their Applications*, 2005 (To Appear).