A new lattice construction for partial key exposure attack for RSA

Yoshinori Aono

Dept. of Mathematical and Computing Sciences
Tokyo Institute of Technology, Tokyo, Japan
aono5@is.titech.ac.jp

Abstract. In this paper we present a new lattice construction for a lattice based partial key exposure attack for the RSA cryptography. We consider the situation that the RSA secret key d is small and a sufficient amount of the LSBs (least significant bits) of d are known by the attacker. We show that our lattice construction is theoretically more efficient than known attacks proposed in [2, 7].

Keywords: RSA, cryptanalysis, partial key exposure attack, lattice basis reduction, the Coppersmith technique

1 Introduction

In this paper we present a new lattice construction for a lattice based partial key exposure attack for the RSA cryptography in the situation that the secret key d is small and its LSBs (least significant bits) are exposed.

Boneh and Durfee [2] proposed the lattice based attack for the RSA cryptography. Its basic idea is to reduce the RSA key finding problem to problems of finding small roots of a modular equation such as $f(x_1, \ldots, x_n) \equiv 0 \pmod{W}$, which are solved by the Coppersmith technique [6], the technique that solves a given modular equation by converting it to an algebraic equation by using a lattice basis reduction algorithm such as the LLL algorithm [11]. Boneh and Durfee [2] showed that the secret key d can be computed from a public key pair e and N in polynomial time in $\log N$ when $d < N^{0.292}$.

Since Boneh and Durfee's work, many of its variants have been proposed [4, 7]. Blömer and May [4] extended the technique for a partial key exposure attack, i.e., a problem of computing d from e, N and some partial information on d. This approach has been further extended by Ernst et al. [7] for several partial key exposure situations. In this paper we consider one of those situations where the secret key d is small and a sufficient amount of d is given (besides e and N), and we show an improvement over the algorithm by Ernst et al.[7], thereby solving an open problem raised in their paper.

In order to state our improvement we need some notations; see the next section for the precise definition. Let (e, N) be an RSA public key pair and let d be its corresponding secret key. Here as usual we use $\ell_N =$ (the bit-length of N) as a security parameter. We consider the situation that $\ell_d =$ (the bit length

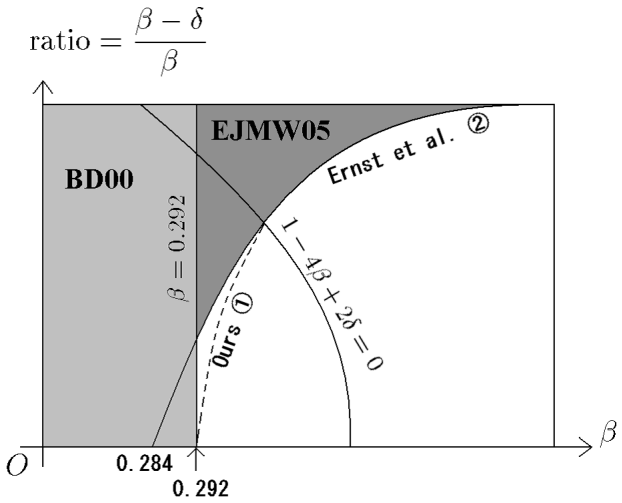

Fig. 1. Our recoverable range

The limit of Boneh and Durfee (1) and that of Ernst et al. (2) are corresponding to the left/above of the line $\beta = 0.292$ and the line `Ernst et al.` ② respectively. Our new improvement area (3) is the left side of the dashed line `Ours` ① in the area left of line $1 - 4\beta + 2\delta = 0$.

of d) is relatively small compared with ℓ_N, and some ℓ_0 least significant bits of d are known. Let $\beta = \ell_d/\ell_N$ and $\delta = (\ell_d - \ell_0)/\ell_N$; that is, they are respectively the ratios of the bit-length of d and its unknown part. Now the asymptotic performance of the algorithms in [2, 7] can be summarized in Figure 1. (This is a rough image, not accurate.)

The algorithm of [2] works asymptotically when the parameters take values in the left of a vertical line labelled "$\beta = 0.292$". That is, it obtains the secret key for

$$\beta < 1 - \frac{1}{\sqrt{2}} = 0.292... \text{ and any } \delta. \tag{1}$$

The algorithm of [7] works when

$$\delta < \frac{5}{6} - \frac{1}{3}\sqrt{1 + 6\beta}. \tag{2}$$

That is it works when β and δ take values in the left/above of a curve labelled `Ernst et al.` ②. As shown in the Figure 1, the algorithm of [7] improves the solvable parameter range when δ is small; it has been however left open [7] to develop an algorithm that has a better solvable parameter range than both [2] and [7].

In this paper we propose an algorithm that can work asymptotically when the parameters take values in the left/above of the dashed line of Figure 1. More precisely, it works when

$$1 - 4\beta + 2\delta > 0 \text{ if } 2\sqrt{2(1-2\beta)(\beta-\delta)}(\delta-\beta) - 2\beta^2 + 3\beta + \delta - 1 < 0 \quad (3)$$

and

$$1 - 4\beta + 2\delta \leq 0 \text{ if } \delta < \frac{5}{6} - \frac{1}{3}\sqrt{1+6\beta}. \quad (4)$$

Note that the range (3) is our new improvement area which is the left/above of the dashed line `Ours` ① in Figure 1, while the range (4) is already given by [7]. Consider the situation that we do not have any information on d, i.e., $\delta = \beta$. Substituting this to (3) and (4), we have $\beta < 1 - \sqrt{2}/2 \approx 0.292$ and $\beta < 7/6 - \sqrt{7}/3 \approx 0.284$ respectively. This means our range covers [7]'s range when $\delta \approx \beta$.

This paper is organized as follows. In Section 2, we introduce some notations and lemmas about the lattice based partial key exposure attack. Section 3 provides the overview of the lattice based partial key exposure attack. In Section 4 we describe the construction and the performance of our lattice. Section 5 provides the results of our computer experiments. The analysis of Section 4 is explained in Section 6.

2 Preliminaries

We introduce some notations and state some known facts used in the following discussions. Then we review some key technical lemmas used in the lattice based attack.

We use standard RSA notations throughout this paper. A given RSA instance is defined by p, q, e, and d, where p and q are large primes, e is a public key, and d is a secret key. Let $N = p \times q$, and let $\varphi(N)$ be the Euler's function; here we will simplify assume that $\varphi(N) = (p-1)(q-1)$. The key relation is

$$ed \equiv 1 \pmod{\varphi(N)}. \quad (5)$$

The partial key exposure attack is to compute the secret key d from partial information on d, and the public key (e, N). In this paper, we consider the situation that some LSBs of d are exposed, that is, recovering d from LSBs of d (together with e and N). We use d_0 to denote the exposed part and $\tilde{d}$ to denote the non-exposed part. That is, we assume that

$$d = \tilde{d} \cdot M + d_0, \quad (6)$$

where $M = 2^k$ and $k = \lg(d_0)$ [1]. We will use M for denoting this number throughout this paper. Define $\beta = \log_N d$ and $\delta = \log_N \tilde{d}$. That is, β and δ are the rough ratios of the bit-length of d and $\tilde{d}$ relative to that of N respectively.

[1] We use $\lg(x)$ to denote the length of the binary representation of x.

In the algorithm, we need to solve a modular equation such as $f(x,y) \equiv 0 \pmod{W}$ for a polynomial $f(x,y)$. Furthermore, we want to obtain a solution in a certain range. In general, this task is not easy. However there are some cases where we may be able to use the standard numerical method for solving modular equations. The Howgrave-Graham lemma [9] provides us with one of such cases.

In order to state the Howgrave-Graham Lemma, we introduce the following notation.

Definition 1 *XY-norm* *Let X and Y be natural numbers and $f(x,y) = \sum_{i,j} a_{i,j} x^i y^j$ be a polynomial with integral coefficient.*

We denote the length of a coefficient vector of $f(Xx, Yy)$ by $||f(x,y)||_{XY}$, i.e.,

$$||f(x,y)||_{XY} \stackrel{\text{def}}{=} \sqrt{\sum_{i,j} a_{i,j}^2 X^{2i} Y^{2j}}.$$

We call this the XY-norm of $f(x,y)$.

Lemma 1 (Howgrave-Graham [9]) *For any positive integers X, Y and W, let $f(x,y)$ be a bivariate polynomial consisting of w terms with integral coefficient. We suppose that the following holds*

$$||f(x,y)||_{XY} < \frac{W}{\sqrt{w}}.$$

Then we have

$$f(x,y) \equiv 0 \ (mod\ W) \Leftrightarrow f(x,y) = 0$$

within the range of $|x| < X$ and $|y| < Y$.

Note that $f(x,y) = 0$ clearly implies $f(x,y) \equiv 0 \pmod{W}$. What is important is its converse. This lemma guarantees that the solution of $f(x,y) \equiv 0 \pmod{W}$ in the target range can be found (if they exist) from the solutions of $f(x,y) = 0$, which can be obtained by the standard numerical method.

In order to use the lemma, we need to obtain a polynomial with a small XY-norm. The key idea of the lattice based attack is to formulate this task as the shortest vector problem and use approximate solutions computed by a polynomial time lattice basis reduction algorithm for the shortest vector problem.

We introduce some definitions and some lemmas about the lattice. Consider linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^{\tilde{n}}$, then the lattice with basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is defined by

$$L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} a_i \mathbf{b}_i \,\middle|\, a_i \in \mathbb{Z} \text{ for } i = 1, \ldots, n \right\}. \tag{7}$$

That is, the lattice is the set of integral linear combinations of its basis vectors. We denote by n a number of vectors, which is usually called *lattice dimension*, and denote by $\tilde{n}$ a number of component of vector in basis, which we call *lattice component size*. Note that the lattice is a additive subgroup of $\mathbb{R}^{\tilde{n}}$.

The shortest vector problem, for given basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$, is to find a vector $\mathbf{v}$ such that $\mathbf{v} \in L(\mathbf{b}_1, \ldots, \mathbf{b}_n) \setminus \{\mathbf{0}\}$ and $|\mathbf{v}| \leq |\mathbf{v}'|$ for $^\forall\mathbf{v}' \in L(\mathbf{b}_1, \ldots, \mathbf{b}_n) \setminus \{\mathbf{0}\}$. That is, this problem is to find a non-zero vector having the minimum length in $L(\mathbf{b}_1, \ldots, \mathbf{b}_n)$.

In order to obtain polynomials with small XY-norms, we need to compute short vectors as approximate solutions of this problem. We will use a polynomial time algorithm, named LLL, proposed in [11]. Some improvements have been proposed [13, 14], but as shown later, these improvements are not essential for our application.

The approximation ratio of the LLL algorithm is exponential, it is however enough for our propose. The following theorem guarantees the upper bounds of the length of the computed vectors. The LLL algorithm computes a special basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$, named reduced basis, from given basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$. Our interest is short vectors in the reduced basis in the following theorem.

Theorem 1 *[2, Fact 3.3] Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a given linearly independent basis. Then we can find linearly independent lattice vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ such that*

$$\begin{array}{l} |\mathbf{v}_1| \leq 2^{(n-1)/4} |\det(L)|^{1/n}, \text{ and} \\ |\mathbf{v}_2| \leq 2^{n/2} |\det(L)|^{1/(n-1)}. \end{array} \tag{8}$$

Here, L is the lattice with basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$, and $\det(L)$ is the determinant of the lattice defined by using their Gram-Schmidt orthogonal basis $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$ as follows

$$\det(L) = \prod_{i=1}^{n} |\mathbf{b}_i^*|. \tag{9}$$

We will use (9) to evaluate the determinant of our lattice in the later section.

Note that the shortest vector problem is defined on vectors, while our targets are polynomials. Thus we consider a way to map polynomials to vectors. For example, the polynomial $f(x, y) = -3x^3 + 4x^2y - 2xy^2 + 7xy^3$ is mapped to the vector $(-3X^3, 4X^2Y, -2XY^2, 7XY^3)$ by some natural numbers X and Y. To state this correspondence formally, we first need to fix some linear ordering on pairs (i, j) of nonnegative integers. With respect to this ordering let $(i(t),j(t))$ denote the t-th pair. Then our correspondence between polynomials and vectors is defined as follows.

Definition 2 Polynomials $\leftrightarrow$ vectors *Let J be a sequence of pairs of non-negative integers, where we assume some linear order on J, let it be fixed, and let $\tilde{n}$ denote $|J|$, the length of the sequence. We also fix some positive integers X and Y. W.r.t. these X and Y, for any $f(x, y) = \sum_{1 \leq t \leq \tilde{n}} a_{i(t),j(t)} x^{i(t)} y^{j(t)}$, the following vector $\mathbf{b}$ is the vectorisation of $f(x, y)$ with parameter X and Y, and it is denoted by $\mathcal{V}_J(f; X, Y)$.*

On the other hand, for any $\mathbf{b}$ of size $\tilde{n}$, a polynomial $f(x, y)$ defined from $\mathbf{b}$ by interpreting it as below is called the functionalisation of $\mathbf{b}$ and it is denoted by $\mathcal{F}_J(\mathbf{b}; X, Y)$.

$$
\begin{array}{lll}
f(x,y) = & a_{i(1),j(1)}x^{i(1)}y^{j(1)} + a_{i(2),j(2)}x^{i(2)}y^{j(2)} + \cdots + a_{i(|J|),j(|J|)}x^{i(|J|)}y^{j(|J|)} \\
& \downarrow \qquad\qquad \downarrow \qquad\qquad\qquad \downarrow \\
\mathbf{b} \quad = (& a_{i(1),j(1)}X^{i(1)}Y^{j(1)} \ , \ a_{i(2),j(2)}X^{i(2)}Y^{j(2)} \ , \ \ldots \ , \ a_{i(|J|),j(|J|)}X^{i(|J|)}Y^{j(|J|)} \ ).
\end{array}
$$

Remark When J is clear from the context, we often omit J and write as $\mathcal{V}(f; X, Y)$ and $\mathcal{F}(\mathbf{b}; X, Y)$. Then from the definition, the following relationships are immediate.

$$
\begin{array}{l}
||f(x,y)||_{XY} = |\mathcal{V}(f; X, Y)|, \text{ and} \\
||\mathcal{F}(\mathbf{b}; X, Y)||_{XY} = |\mathbf{b}|.
\end{array} \tag{10}
$$

That is, these are equivalent to the length of a coefficient vector of $f(Xx, Yy)$.

3 Overview of the Partial Key Exposure Attack

We give an overview of the lattice based partial key exposure attack in the situation that LSBs of d are exposed. The goal of the attack is to compute the secret key d from d_0, least significant bits of d, and a given public key pair. The lattice based attack achieves this goal by using a lattice reduction algorithm and the Howgrave-Graham lemma. It is said in [7] (and some papers) that the attack is effective if

(i) d, and unknown part of d are short,
(ii) e and N are of similar bit length, and
(iii) p and q are of similar bit length.

In order to be precede, we consider in this paper, the following conditions.

(a) $\delta = \log_N \tilde{d}$ is smaller than 0.5,
(b) $\lg(e) = \lg(N)$, and
(c) $\lg(p) = \lg(q)$

In the following, we assume all parameters satisfy these conditions. More precisely, we will use the following inequalities in the later.

$$
e < \varphi(N) \text{ and } p + q < 3\sqrt{N}. \tag{11}
$$

Our objective is to compute d from a public key pair (e, N) and d_0. As explained in Introduction, the key relation is the modular equation (5), from which it is easy to derive $ed = 1 - x\varphi(N) = 1 - x(y + N)$ for some $x, y \in \mathbb{Z}$. Also by using (6), we can deduce from the above that $e(\tilde{d} \cdot M + d_0) = 1 - x(y + N)$ and hence we have

$$
x(N + y) + (ed_0 - 1) \equiv 0 \pmod{eM}. \tag{12}
$$

We show here that it is relatively easy to enumerate all solutions (x, y) of (12). First note that a solution (x, y) exists if for integer y,

$$
\gcd\left(\frac{N+y}{g}, \frac{eM}{g}\right) = 1 \text{ where } g = \gcd(N + y, eM, ed_0 - 1).
$$

In fact in this case, we can compute x by $x = \left(\frac{1-ed_0}{g}\right) \cdot \left(\left(\frac{N+y}{g}\right)^{-1} \bmod \frac{eM}{g}\right)$.
But clearly what we need is some specific solution of (12). Among solutions (x, y) of (12), we say that (x_0, y_0) is *useful* if it indeed satisfies the following equation, from which we can recover the secret key.

$$d = \frac{1 - x_0(N + y_0)}{e} \tag{13}$$

Thus, our task is not computing *some* solutions (x, y), but computing this useful solution among (x, y) satisfying (12). Below we use (x_0, y_0) to denote this useful solution. Let us consider a size of the useful solution (x_0, y_0). We have the following upper bounds. Here, we use (11) and the fact that $\varphi(N) = N + y_0$ if (x_0, y_0) is the useful solution.

$$\begin{aligned} |x_0| &= \left|\frac{ed-1}{N+y_0}\right| < \frac{ed}{\varphi(N)} < d = N^{\beta}, \text{ and} \\ |y_0| &= |N - \varphi(N)| = p + q - 1 < 3N^{0.5}. \end{aligned} \tag{14}$$

Now let $X = \lceil N^{\beta} \rceil$ and $Y = \lceil 3N^{0.5} \rceil$. Then, the useful solution (x_0, y_0) is a solution of (12) satisfying $|x_0| < X$ and $|y_0| < Y$.

Conversely, we consider some heuristic condition on δ for a solution satisfying $|x| < X$ and $|y| < Y$ is useful. We assume that solutions of (12) are random numbers on $\{0, \ldots, eM - 1\}^2$. Since the number of solution pairs of (12) is smaller than eM, we expect the number of solutions satisfy $|x| < X$ and $|y| < Y$ is smaller than

$$eM \cdot \frac{4XY}{(eM)^2} = \frac{4XY}{eM} \approx \frac{4 \cdot N^{\beta} \cdot 3N^{0.5}}{N \cdot N^{\beta-\delta}} \approx N^{\delta-0.5}.$$

Thus, if this value is smaller than 1, we may expect a solution within the range $|X| < N^{\beta}$ and $|y| < N^{0.5}$ is only one, which is the useful solution guaranteed by (14). From this observation we propose a condition $\delta < 0.5$ and the following heuristic assumption.

Heuristic Assumption Consider the case $\delta < 0.5$. Then, there is only useful solution (x_0, y_0) within the range of $|x| < N^{\beta}, |y| < 3N^{0.5}$ of the following equation.

$$x(N + y) + (ed_0 - 1) \equiv 0 \pmod{eM}$$

Furthermore we can recover the secret key d by (13)[2]. □

Remark. This assumption shows we can obtain the secret key by the exhaustive search when $\delta < 0.5$.

[2] Moreover we can compute the factoring of N by $\varphi(N) = N + y_0$.

1. Based on $f_{\mathrm{main}}(x, y)$ (and $f_{\mathrm{M}}(x, y)$), define a certain family of polynomials $h_1(x, y), \ldots, h_n(x, y)$ such that

$$f_{\mathrm{main}}(x, y) \equiv 0 \pmod{eM} \Rightarrow h_c(x, y) \equiv 0 \pmod{(eM)^m} \text{ for } c = 1, \ldots, n.$$

   (Here m and n are some algorithm parameter defined later.)
2. Set $X = \lceil N^{\beta} \rceil$ and $Y = \lceil 3N^{0.5} \rceil$. Consider vectors by $\mathbf{b}_c = \mathcal{V}_J(h_c; X, Y)$ for $c = 1, \ldots, n$. Here, a sequence J is a set of appropriately ordered integer pairs (i, j) such that a monomial $x^i y^j$ appears in $h_c(x, y)$.
3. For $\mathbf{b}_1, \ldots, \mathbf{b}_n$, compute reduced basis by a lattice basis reduction algorithm. We denote by $\mathbf{v}_1, \ldots, \mathbf{v}_n$ this reduced basis.
4. Define $g_1(x, y)$ and $g_2(x, y)$ by $g_a(x, y) = \mathcal{F}_J(\mathbf{v}_a; X, Y)$ respectively. Obtain solutions of $g_1(x, y) = g_2(x, y) = 0$ numerically. Then from these solutions, compute d as given by (13).

Fig. 2. Outline of the lattice based attack

For our discussion, let us define the following two functions [3]

$$\begin{aligned} f_{\mathrm{main}}(x, y) &\stackrel{\mathrm{def}}{=} x(N + y) + (ed_0 - 1) \\ &= (ed_0 - 1) + Nx + xy, \text{ and} \end{aligned} \tag{15}$$

$$f_{\mathrm{M}}(x, y) \stackrel{\mathrm{def}}{=} M(-1 + x(N + y)). \tag{16}$$

$f_{\mathrm{main}}(x, y)$ is the left-hand side of equation (12). The motivation of $f_{\mathrm{M}}(x, y)$ will be explained later; here we only point out that $f_{\mathrm{M}}(x_0, y_0) \equiv 0 \pmod{eM}$, where (x_0, y_0) is a useful solution.

Now we summarise the above explanation. Our technical goal is to obtain the useful solution (x_0, y_0) satisfying (12), in other words, a pair satisfying *both* $f_{\mathrm{main}}(x_0, y_0) \equiv 0 \pmod{eM}$, $|x_0| < N^{\beta}$ *and* $|y_0| < 3N^{0.5}$. For achieving this technical goal by solving the modular equation, we make use of the Howgrave-Graham Lemma, and for this purpose, we modify $f_{\mathrm{main}}(x, y)$ to some family of functions with small XY-norm. The task of defining these polynomials is formulated as the shortest vector problem, and a known polynomial time algorithm such as the LLL algorithm is used. This is the rough sketch of the lattice based attack. The outline of our algorithm is stated as Figure 2.

Some remarks may be necessary. Note first that m and n are algorithm parameters; m is chosen appropriately and n is the number of polynomials $h_c(x, y)$ that is also determined appropriately based on m. Secondly note that we have

[3] Ernst et al. [7] reduced the problem to the problem of finding small solution of an algebraic equation

$$f_{\mathrm{LSB}}(x, y, z) = eMx - Ny + yz + e\tilde{d} - 1 = 0.$$

for $|x| < X$ and $|y| < Y$ the following relation between these polynomials.

$$\begin{aligned} &f_{\mathrm{main}}(x,y) \equiv 0 \pmod{eM} \;\Rightarrow\; h_c(x,y) \equiv 0 \pmod{(eM)^m} \text{ for } c = 1,\ldots,n \\ &\Rightarrow\; g_a(x,y) \equiv 0 \pmod{(eM)^m} \text{ for } a = 1,2 \;\Leftrightarrow\; g_a(x,y) = 0 \text{ for } a = 1,2. \end{aligned} \tag{17}$$

The key point of (17) is the relation $g_a(x,y) \equiv 0 \pmod{(eM)^m} \Leftrightarrow g_a(x,y) = 0$ for $a = 1,2$, $|x| < X$ and $|y| < Y$. This holds when $||g_a(x,y)||_{XY} = |\mathbf{v}_a|$ is smaller than $(eM)^m/\sqrt{w}$ from the Howgrave-Graham lemma. (Here w is the number of terms of each $g_a(x,y)$.) Then we have the relation

$$f_{\mathrm{main}}(x,y) \equiv 0 \pmod{eM} \Rightarrow g_a(x,y) = 0 \text{ for } a = 1,2.$$

Hence we can obtain the useful solution from computing all solutions of $g_1(x,y) = g_2(x,y) = 0$ by some numerical method if it exists.

By (8) and (10), we have

$$\begin{aligned} ||g_1(x,y)||_{XY} = |\mathbf{v}_1| &\le 2^{(n-1)/4} \det(L)^{1/n}, \text{ and} \\ ||g_2(x,y)||_{XY} = |\mathbf{v}_2| &\le 2^{n/2} \det(L)^{1/(n-1)}. \end{aligned}$$

Since the second bound is larger, we obtain the following sufficient condition for using the Howgrave-Graham lemma:

$$2^{n/2} \det(L)^{1/(n-1)} < \frac{(eM)^m}{\sqrt{w}}. \tag{18}$$

We modify (18) to a more simple approximate bound as follows.

$$\det(L)^{1/n} < (eM)^m \tag{19}$$

Now our goal is to construct a lattice satisfying (19), and we will show in the next section that it is possible if β and δ satisfies the condition (3) given in Introduction.

4 Our Construction

In this section, we explain our construction and a main result. The largest difference between former algorithm and ours is a lattice construction satisfying (19). We will give its analysis in detail in Section 6.

Let β and δ be assumed bounds defined above, m be an algorithm parameter introduced in the above outline, τ be a parameter used to optimise the bounds by β and δ. We fix them throughout this section. We introduce index series $I_a(m,\tau,\beta,\delta)$ for constructing our lattice $L(m,\tau,\beta,\delta)$.

Definition 3 *We define our sequence $I_1(m,\tau,\beta,\delta), I_2(m,\tau,\beta,\delta)$ and $I_3(m,\tau,\beta,\delta)$ (In short, I_1, I_2 and I_3 respectively). Here we set*

$$\begin{aligned} I_1(m,\tau,\beta,\delta) &= \{(i,j) \in \mathbb{Z}\times\mathbb{Z} | 0 \le i \le m, 0 \le j \le i\}, \\ I_2(m,\tau,\beta,\delta) &= \{(i,j) \in \mathbb{Z}\times\mathbb{Z} | 0 \le i \le m, i < j \le i + \tau m\} \text{ and} \\ I_3(m,\tau,\beta,\delta) &= \{(i,j) \in \mathbb{Z}\times\mathbb{Z} | 0 \le i \le m, j \le 2(1-\beta)i\} \setminus (I_1 \cup I_2). \end{aligned}$$

We consider the order $\prec$ in I_1, I_2 and I_3 by the lexicographic order of (i,j) [4]*. Then we define the index sequence $I(m,\tau,\beta,\delta)$ by concatenating I_1, I_2 and I_3. That is, order of elements in I is defined as follows for $(i,j) \in I_k$ and $(i',j') \in I_{k'}$,*

$$(i,j) \prec (i',j') \Leftrightarrow \begin{cases} k < k' \text{ or} \\ k = k' \text{ and } (i,j) \prec (i',j') \text{ in } I_k. \end{cases}$$

We define our polynomials $f_{i,j}(x,y)$ to construct our lattice (this is $h_c(x,y)$ in the outline).

Definition 4

$$f_{i,j}(x,y) = \begin{cases} (eM)^{m-j} x^{i-j} (f_{\text{main}}(x,y))^j & \text{for } (i,j) \in I_1 \\ (eM)^{m-i} y^{j-i} (f_{\text{main}}(x,y))^i & \text{for } (i,j) \in I_2 \\ e^{m-i} y^{j-i} (f_{\text{M}}(x,y))^i & \text{for } (i,j) \in I_3 \end{cases} \tag{20}$$

Then we define a sequence $J(m,\tau,\beta,\delta) = \{(i',j') | a\ monomial\ x^{i'}y^{j'}\ is\ appeared\ in\ some\ f_{i,j}(x,y)\}$ where we assume the standard lexicographic order in $J(m,\tau,\beta,\delta)$. We simply denote this by J.

It is clear that $f_{i,j}(x_0,y_0) \equiv 0 \pmod{(eM)^m}$ for $(i,j) \in I$. The number of polynomials $|I|$ is just n in Figure 2. Note also that $|J| = \tilde{n}$, the number of components of each vector $\mathbf{b}_{i,j}$ is $O(|I|)$ since we can rewrite a set J by $\{(i,j) \in \mathbb{Z} \times \mathbb{Z} | 0 \le i \le m, 0 \le j \le i + (1-2\beta)m\}$. Hence $|I|$ and $|J|$ has a same order $\Theta(m^2)$.

By using these polynomials and indecies, we define our lattice $L(m,\tau,\beta,\delta)$ by

$$L(m,\tau,\beta,\delta) = L(\mathbf{b}_{i_1,j_1}, \ldots, \mathbf{b}_{i_n,j_n}).$$

Here $(i_1,j_1), \ldots, (i_n,j_n)$ are the index sequence in I and $\mathbf{b}_{i_\ell,j_\ell} = \mathcal{V}_J(f_{i_\ell,j_\ell}; X, Y)$, a vectorisation of $f_{i_\ell,j_\ell}(x,y)$ with parameters $X = \lceil N^\beta \rceil$ and $Y = \lceil 3N^{0.5} \rceil$. The lattice dimension and lattice component size of $L(m,\tau,\beta,\delta)$ are $|I|$ and $|J|$ respectively.

For evaluating the determinant of L, we will show

$$\begin{aligned} |\mathbf{b}^*_{i,j}| &= (eM)^{m-j} X^i Y^j \text{ for } (i,j) \in I_1, \\ |\mathbf{b}^*_{i,j}| &= (eM)^{m-i} X^i Y^j \text{ for } (i,j) \in I_2, \text{and} \\ e^{m-j} M^m X^i Y^j \le |\mathbf{b}^*_{i,j}| &< 2e^{m-j} M^m X^i Y^j \text{ for } (i,j) \in I_3. \end{aligned} \tag{21}$$

Here $\mathbf{b}^*_{i_1,j_1}, \ldots, \mathbf{b}^*_{i_n,j_n}$ is a Gram-Schmidt orthogonal basis of given basis. We give the proof of these bounds in Section 6. (Lemma 3, Lemma 4 and Lemma 5). Now we assume that these bounds hold, and we introduce an "evaluator" for deciding suitable τ. The evaluator $\text{eval}(i,j)$ for $\mathbf{b}^*_{i,j}$ is defined by

$$\text{eval}(i,j) = \log_N \left(|\mathbf{b}^*_{i,j}| / (eM)^m \right). \tag{22}$$

[4] Notice that a symbol $\prec$ means R.H.S. is exactly larger than L.H.S., not equal. A symbol $\preceq$ means R.H.S. is equal to or larger than L.H.S.

We define the evaluator for index sequence I by

$$\mathrm{eval}(I) = \sum_{(i,j)\in I} \mathrm{eval}(i,j). \tag{23}$$

Then we can state the condition (19) in terms of eval.

Lemma 2 *For our index sequence I satisfying*

$$\mathrm{eval}(I) < 0, \tag{24}$$

the condition (19) holds.

Proof. By (9) and the definition of the evaluator, we have

$$N^{\mathrm{eval}(I)} = N^{\sum_{(i,j)\in I} \mathrm{eval}(i,j)} = \prod_{(i,j)\in I} \left(\frac{|\mathbf{b}^*_{i,j}|}{(eM)^m} \right) = \det(L)/(eM)^{|I|\cdot m}.$$

Thus, (24) is equivalent to $\det(L) < (eM)^{|I|\cdot m}$. Which is indeed the condition (19). □

Thus, we use $\mathrm{eval}(I) < 0$ as our (approximate) sufficient condition that the lattice based attack (under our construction of the lattice L) breaks a given RSA instance. Note that this condition is based on our heuristic assumption and it is only an approximate condition because of the approximation of (18) by (19); yet further approximation is used below for estimating eval. This means that the condition for parameters β and δ we will derive below is, strictly speaking, not accurate nevertheless, we will argue by using our approximation for avoiding unnecessary complications. Justification of our approximation analysis and together with our heuristic assumption will be given by computer experiments shown later.

Now by using the bound (21) (see Proposition 1 in Section 6.1), we can approximately evaluate $\mathrm{eval}(i,j)$ as follows [5].

$$\mathrm{eval}(i,j) \approx \begin{cases} i\beta - (\beta - \delta + 0.5)j & (i,j) \in I_1 \\ (\delta - 1)i + 0.5j & (i,j) \in I_2 \\ (-1+\beta)i + 0.5j & (i,j) \in I_3. \end{cases}$$

From this we can approximately estimate $\mathrm{eval}(I)$. From some calculation (see Section 6.2) we have

$$\mathrm{eval}(I) = \left(\frac{1}{6}(\beta + \delta - 0.5) + \frac{\tau}{4}(2\delta + \tau - 1) - \frac{1}{12}\frac{(1 - 2\beta - \tau)^3}{1 - 2\beta} \right) m^3 + o(m^3) \tag{25}$$

for $1 - 2\beta - \tau > 0$, and we have

$$\mathrm{eval}(I) = \left(\frac{1}{6}(\beta + \delta - 0.5) + \frac{\tau}{4}(2\delta + \tau - 1) \right) m^3 + o(m^3) \tag{26}$$

[5] Here a symbol $A \approx B$ means $\frac{A}{B} \to 1$ when N (the RSA bit length) goes to infinity.

for $1-2\beta-\tau \leq 0$. Note that conditions $1-2\beta-\tau > 0$ and $1-2\beta-\tau \leq 0$ are corresponding to the cases $I_3 \neq \phi$ and $I_3 = \phi$ respectively. Then following the argument of [2, 7], we analyze the bound for *the ideal case* by assuming that m is sufficiently large. (We draw a figure to show the bounds for some concrete values of m, see Figure 4 in Section 5.)

First we consider the case of $1-2\beta-\tau > 0$. Assuming that m is sufficiently large, the condition (24) is approximately equivalent to

$$\frac{1}{6}(\beta+\delta-0.5)+\frac{\tau}{4}(2\delta+\tau-1)-\frac{1}{12}\frac{(1-2\beta-\tau)^3}{1-2\beta} < 0, \tag{27}$$

where its left-hand side is minimised when τ takes value $\tau_0 = \sqrt{2(1-2\beta)(\beta-\delta)}$. Hence, substituting this to (27), we have

$$\frac{1}{3}\sqrt{2(1-2\beta)(\beta-\delta)}(\delta-\beta)-\frac{1}{3}\beta^2+\frac{1}{2}\beta+\frac{1}{6}\delta-\frac{1}{6} < 0. \tag{28}$$

This is equivalent to (3).

Next we consider the case of $1-2\beta-\tau \leq 0$. Again assuming m is sufficiently large, we have the following approximate condition (24).

$$\frac{1}{6}(\beta+\delta-0.5)+\frac{\tau}{4}(2\delta+\tau-1) < 0. \tag{29}$$

By similar argument, we substitute $\tau_1 = \frac{1-2\delta}{2}$ to τ for minimising (29), and derive the following condition (29).

$$\delta < \frac{5}{6}-\frac{1}{3}\sqrt{1+6\beta}. \tag{30}$$

This is equivalent to the condition proposed by [7]. Therefore, we can recover the secret key of RSA in polynomial time in $\log N$ when β and δ satisfies (3) or (4).

5 Computer Experiments

We carried out our *preliminary* computer experiments to check that our approach works and estimate its efficiency. We conducted our computer experiments on the TSUBAME supercomputer [6] We implemented our experiment program by the C++ language using Shoup's NTL [12] of version 5.4.2. We carried out the lattice reduction part by L^2 algorithm [13, 14] with parameter $\delta = 0.99$ and

[6] TSUBAME is a grid type supercomputer at Tokyo Inst. of Tech., whose performance is currently (by Top500, Nov. 2008) the 29th in the World. A node of the supercomputer which we used contains eight Opteron Dual Core model 880 processors of 2.4GHz and 32GB RAM. Note, however, we have not been able to make a parallel version of our algorithm; TSUBAME's massive parallelism has been used only for reducing the total experiment time.

Step 1: (Make sample RSA instance) Randomly choose $\ell/2$-bit primes p and q, and let $N = pq$. (In our program, we choose p and q the Euler-Jacobi pseudoprime to bases 2, 3, 5, 7 and 11.) Randomly choose $\lfloor \beta\ell \rfloor$-bit random odd integer as the secret key d such that $\gcd(d, (p-1)(q-1)) = 1$, and let $M = 2^{\lfloor (\beta-\delta)\ell \rfloor}$ and $d_0 = d \bmod M$. Compute the public key $e \equiv d^{-1} \pmod{(p-1)(q-1)}$, and let $f_{\mathrm{main}}(x, y) = x(N + y) + (ed_0 - 1)$, and let $y_0 = 1 - p - q$ and $x_0 = (1 - ed)/((p-1)(q-1))$.

Step 2: Let $\tau = \sqrt{2(1-2\beta)(\beta-\delta)}$, $X = \lfloor N^{\beta} \rfloor$ and $Y = \lfloor 3N^{0.5} \rfloor$. Then construct our lattice $L(m, \tau, \beta, \delta)$.

Step 3: Apply the L^2 algorithm for $L(m, \tau, \beta, \delta)$ with L^2 parameter $(\delta, \eta) = (0.99, 0.51)$.

Step 4: Sort the vectors of reduced basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ by these length to $\mathbf{v}'_1, \ldots, \mathbf{v}'_n$. Compute $g_i(x, y) = \mathcal{F}_I(\mathbf{v}'_i, X, Y)$ for $i = 1, 2$.

Step 5: Check $g_1(x_0, y_0) = 0$ and $g_2(x_0, y_0) = 0$. (If $g_1(x_0, y_0) \neq 0$ or $g_2(x_0, y_0) \neq 0$, the experiment is failure.) Compute $R(x) = \mathrm{Res}(h_1, h_2)$ and check $R(x) \neq 0$ and $R(x_0) = 0$ holds or not.

Fig. 3. Our computer experiment procedure

$\eta = 0.51$, [7] and implemented the resultant calculation algorithm by [8]. We compiled our source code by `gcc-4.1.2` (64 bit version) with `-O6` option.

The procedure of our experiments is shown in Figure 3. The algorithm part is essentially the same as the outline in Figure 2. At Step 4, the vectors obtained by the L^2 algorithm are sorted by their length; this is because those vectors are approximate ones and we cannot guarantee that $\mathbf{v}_1$ and $\mathbf{v}_2$ are the smallest in reduced basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$. We check the algebraic independence of g_1 and g_2 by checking $R(x) \neq 0$ holds or not. More precisely, we regard the experiment is succeeded if $R(x) \neq 0$ and $R(x_0) = 0$.

Input parameters of this experiments are ℓ, m, β and δ, which are respectively the bit-size of N, the parameter for constructing lattice, the ratio of $\lg(d)$ to $\lg(N)$, and the ratio of $\lg(\tilde{d})$ to $\lg(N)$. (See Figure 2 for the parameters m and n, and see Section 3 for the parameters β and δ.) A word "dim." means the lattice dimension in experiments, i.e., $n = |I|$ in our construction. By "total time"and "L^2 time" we mean the CPU time of Step 2 through Step 4 and that of Step 3 respectively. Recall that the lattice component size is bounded by a constant time of the lattice dimension. Here, we disregard the time for computing the resultant of g_1 and g_2. Results are in Table 1 and Table 2. Table 1 shows the qualities of our lattices, that is, whether the experiment is succeeded or not, for these parameters. On the other hand, Table 2 shows the computational time of our experiments for various ℓ and m, and fixed β and δ.

Quality of Lattice

For checking our approach indeed works and estimating the quality of our lattice construction, we carried out our preliminary computer experiments.

[7] This δ is L^2 algorithm's parameter, not relating to the RSA secret key.

Note first that the bounds (3) and (4) are the ideal ones obtained by the asymptotic analysis assuming m is sufficiently large. For each given value of m, we can determine the range of β and δ satisfying $\mathrm{eval}(I(m, \tau_0, \beta, \delta)) < 0$ by numerically analyzing the original expressions (i.e., (34) $\sim$ (36) of Appendix A.2). Figure 4 shows the bounds of β and δ obtained in this way for some m values, and the theoretical limit of our construction (3) and that of [7]'s construction (2). It can be seen that these bounds get close to our ideal bound when m gets large. We focus on the range of $0.28 \leq \beta \leq 0.32$ and $0 \leq \mathrm{ratio} \leq 0.2$, mainly our new improvement area.

Our computer experiments are summarized in Figure 5, which are for the cases $m = 10$ and 14. The instance size ℓ $(= \lg(N))$ is 1024 for both cases; since these are still preliminary ones, we conducted only one execution for each instance. Note that a shade area in each figure is the area that $\mathrm{eval}(I(m, \tau_0, \beta, \delta)) < 0$ obtained numerically for each m. A black circle (resp., a white circle) indicates the parameter (β, δ) (or the point $(\beta, (\beta - \delta)/\beta)$) that the experiment succeeds (resp., fails). Those results are shown in detail in Table 1. For $m = 10$, the word "yes" in the column "success" in the tables means $R(x) \neq 0$ and $R(x) = 0$ at the Step 5. On the other hand, "yes" for $m = 14$ means $g_1(x_0, y_0) = g_2(x_0, y_0) = 0$ at the Step 5. The difference is because of the resultant calculation is too heavy operation for our C++ implementation. We expect this problem will be solved by the technique of Glöbner basis. [8]

The word "no(1)" in the column "success" means $g_1(x_0, y_0) = 0$ whereas $g_2(x_0, y_0) \neq 0$ at the Step 5. In this case, we may expect to get the correct solution by generating enough number of polynomials g_i by changing X and Y randomly within the same bit length. On the other hand, the word "no(0)" means that $g_1(x_0, y_0) \neq 0$ and $g_2(x_0, y_0) \neq 0$, or $R(x_0) \neq 0$ at Step 5, which we regarded as a failure.

Computational Time

Next we examine the efficiency of our algorithm based on our experiments. As seen by our analysis and experiments, the algorithm shows better performance by using large m. On the other hand, by using larger m, the lattice dimension also get larger. More specifically, the lattice dimension is $\Theta(m^2)$ by our construction. We examine how this indeed effects to the running time of the algorithm.

We carried out computer experiments with parameters $(\beta, \delta) = (0.3, 0.225)$ and $(\beta, \delta) = (0.4, 0.12)$, whereas parameters m and ℓ are chosen as $m = 6$, 8, 10 and 12, and $\ell = 512$, 1024, and 2048. Table 2 is experiments for these parameters. Total time and L^2 time in these tables are the average running time of five executions. Notice that we checked the algebraic independence of $g_1(x, y)$ and $g_2(x, y)$, and compute the resultant $R(x)$ for $m = 6$, 8 and 10. All of them are success. However we did not check the algebraic independence for $m = 12$, only checked $g_1(x_0, y_0) = g_2(x_0, y_0) = 0$. At this meaning, experiments for $m = 12$ are success.

[8] We carried out the check of algebraic independence because referee's comment suggested it. However some of these checks are not completed at the deadline.

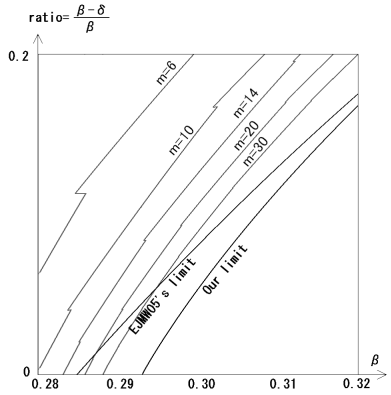

Fig. 4. Our recoverable range for many m

These results show the total time and L^2 time are close, which shows a lattice reduction algorithm is the main part of the lattice based attack. From these tables, we obtain our L^2 time is approximately

$$\text{Time} \approx 0.35 \left(\frac{\ell}{512}\right)^2 \cdot \left(\frac{m}{2}\right)^8 \cdot \log_2 \ell \cdot \log_2 m \text{ sec for } \beta = 0.3 \text{ and } \delta = 0.225, \text{ and}$$
$$\text{Time} \approx 0.6 \left(\frac{\ell}{512}\right)^2 \cdot \left(\frac{m}{2}\right)^8 \cdot \log_2 \ell \cdot \log_2 m \text{ sec for } \beta = 0.4 \text{ and } \delta = 0.12.$$

6 Analysis

This section provides the precise analysis for some results in Section 4. Our objective is to derive our theoretical recoverable limit of our construction (28) and (30).
In order to obtain these inequalities, we first prove the approximated estimation

$$\text{eval}(i,j) \approx \begin{cases} \beta i - (\beta - \delta + 0.5)j & (i,j) \in I_1 \\ (\delta - 1)i + 0.5j & (i,j) \in I_2 \\ (-1+\beta)i + 0.5j & (i,j) \in I_3 \end{cases} \tag{31}$$

in Section 6.1. Next in Section 6.2 we explain the way to derive our theoretical limit. We fix algorithm parameters and RSA instances throughout this section, and denote $f^*_{i,j}(x,y)$ the functionalisation of $\mathbf{b}^*_{i,j}$. (We recall that $\{\mathbf{b}^*_{i,j}\}_{(i,j)\in I}$ is the Gram-Schmidt orthogonal basis of our basis
$\{\mathbf{b}_{i,j}\}_{(i,j)\in I} \stackrel{\text{def}}{=} \{\mathcal{V}(f_{i,j}; X, Y)\}_{(i,j)\in I}$.)

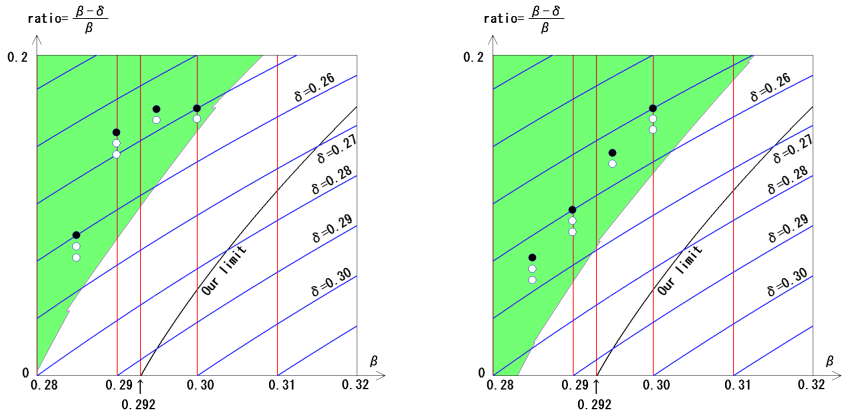

Fig. 5. Summary of our experiments for $\ell = 1024$

6.1 Approximating the Evaluator

We first compute the exact value of $|\mathbf{b}^*_{i,j}|$ for $(i,j) \in I_1$ and I_2 in Lemma 3 and Lemma 4 respectively. Next we consider the bound for $|\mathbf{b}^*_{i,j}|$ for $(i,j) \in I_3$ in Lemma 5. Then we estimate the approximated value of $\mathrm{eval}(i,j) \stackrel{\mathrm{def}}{=} \log_N(|\mathbf{b}^*_{i,j}|/(eM)^m)$ in Proposition 1.

The following lemmas are immediately derived by considering the diagonal part of the matrix representation of L. Hence we omit the proofs of them.

Lemma 3 *For any $(i,j) \in I_1$, we have*

$$|\mathbf{b}^*_{i,j}| = (eM)^{m-j} X^i Y^j. \tag{32}$$

Lemma 4 *For any $(i,j) \in I_2$, we have*

$$|\mathbf{b}^*_{i,j}| = (eM)^{m-i} X^i Y^j. \tag{33}$$

On the other hand, the proof of the following lemma is a bit involved. See the full version of this paper for the proofs of Lemma 3 to Lemma 5.

Lemma 5 *For any $(i,j) \in I_3$, we have*

$$e^{m-i} M^m X^i Y^j \leq |\mathbf{b}^*_{i,j}| < 2e^{m-i} M^m X^i Y^j.$$

By using these lemmas, we obtain the approximated bounds for evaluators for each (i,j).

Proposition 1

$$\mathrm{eval}(i,j) \approx \begin{cases} \beta i - (\beta - \delta + 0.5)j & (i,j) \in I_1 \\ (\delta - 1)i + 0.5j & (i,j) \in I_2 \\ (-1+\beta)i + 0.5j & (i,j) \in I_3 \end{cases}$$

Experiment parameters			Results	
β	δ	dim.	L^2 time	Success
0.285	0.260	88	3 h 42 m	yes
0.285	0.262	88	3 h 46 m	no(1)
0.285	0.264	88	3 h 53 m	no(0)
0.290	0.246	87	3 h 15 m	yes
0.290	0.248	87	3 h 24 m	no(1)
0.290	0.250	87	3 h 1 m	no(0)
0.295	0.246	92	4 h 2 m	yes
0.295	0.248	87	2 h 53 m	no(0)
0.300	0.250	91	3 h 25 m	yes
0.300	0.252	85	2 h 16 m	no(0)

$m = 10$

Experiment parameters			Results	
β	δ	dim.	L^2 time	Success
0.285	0.264	162	2 d 20 h	yes
0.285	0.266	162	2 d 18 h	no(1)
0.285	0.268	162	2 d 9 h	no(0)
0.290	0.260	165	2 d 22 h	yes
0.290	0.262	165	2 d 16 h	no(1)
0.290	0.264	165	2 d 16 h	no(0)
0.295	0.254	164	2 d 19 h	yes
0.295	0.256	164	2 d 17 h	no(0)
0.300	0.250	163	3 d 7 h	yes
0.300	0.252	163	3 d 14 h	no(1)
0.300	0.254	163	3 d 11 h	no(0)

$m = 14$

Table 1. Quality of our lattice for $m = 10$ and $m = 14$

Proof. As we explained at the overview, we assumed that

$$X \approx N^{\beta},\ Y \approx N^{0.5},\ \text{and}\ e \approx N,$$

from which by using (6) we derive $M = \frac{d-d_0}{d} \approx \frac{N^{\beta}}{N^{\delta}} = N^{\beta-\delta}$.
Substituting these for each bound of $\mathbf{b}^*_{i,j}$, we have by using $\mathrm{eval}(i,j) \stackrel{\mathrm{def}}{=} \log_N \left(\frac{|\mathbf{b}^*_{i,j}|}{(eM)^m}\right)$,

$$\begin{array}{ll} \mathrm{eval}(i,j) = \log_N (eM)^{-j} X^i Y^j \approx \beta i - (\beta - \delta + 0.5)j & \text{for } (i,j) \in I_1, \\ \mathrm{eval}(i,j) = \log_N (eM)^{-i} X^i Y^j \approx (\delta - 1)i + 0.5j & \text{for } (i,j) \in I_2, \text{ and} \\ \mathrm{eval}(i,j) \approx \log_N e^{-j} X^i Y^j \approx (-1+\beta)i + 0.5j & \text{for } (i,j) \in I_3. \ \square \end{array}$$

6.2 Some Calculations on eval(I)

This section shows how to get our conditions (28) and (30) in detail.

First we state the expressions for $\mathrm{eval}(I_1)$, $\mathrm{eval}(I_2)$, and $\mathrm{eval}(I_3)$ derived from (23) and Proposition 1.

$$\mathrm{eval}(I_1) = \sum_{i=0}^{m} \sum_{j=0}^{i} \mathrm{eval}(i,j) \ = \ \sum_{i=0}^{m} \sum_{j=0}^{i} (i\beta - (\beta - \delta + 0.5)j)\,, \tag{34}$$

$$\mathrm{eval}(I_2) = \sum_{i=0}^{m} \sum_{j=i+1}^{i+\lfloor \tau m \rfloor} \mathrm{eval}(i,j) \ = \ \sum_{i=0}^{m} \sum_{j=i+1}^{i+\lfloor \tau m \rfloor} ((\delta - 1)i + 0.5j)\,, \text{ and} \tag{35}$$

$$\mathrm{eval}(I_3) = \sum_{i=\lfloor Bm \rfloor}^{m} \sum_{j=i+\lceil \tau m \rceil}^{\lfloor 2(1-\beta)i \rfloor} \mathrm{eval}(i,j) \ = \sum_{i=\lfloor Bm \rfloor}^{m} \sum_{j=i+\lceil \tau m \rceil}^{\lfloor 2(1-\beta)i \rfloor} ((-1+\beta) + 0.5j)\,. \tag{36}$$

Here we use B to denote $\frac{\tau}{1-2\beta}$. Figure 6 shows an area of I_1, I_2 and I_3. By definition, these are sets of integral points in each I_k. The cross point of the line

Experiment parameters				Results	
β	δ	m	deg.	L^2time	Total time
0.3	0.225	6	36	1 m 6 s	1 m 8 s
		8	58	10 m 31 s	10 m 42 s
		10	91	1 h 25 m	1 h 26 m
		12	124	6 h 13 m	6 h 15 m
		14	171	25 h 28 m	25 h 35 m
0.4	0.12	6	35	2 m 4 s	2 m 7 s
		8	54	16 m 32 s	16 m 46 s
		10	77	1 h 10 m	1 h 11 m
		12	117	10 h 11 m	10 h 14 m
		14	150	29 h 56 m	30 h 3 m

$\ell = 512$

Experiment parameters				Results	
β	δ	m	deg.	L^2time	Total time
0.3	0.225	6	36	4 m 34 s	4 m 42 s
		8	58	40 m 5 s	40 m 44 s
		10	91	5 h 33 m	5 h 36 m
		12	124	21 h 25 m	21 h 33 m
		14	171	127 h 0 m	127 h 10 m
0.4	0.12	6	35	8 m 41 s	8 m 52 s
		8	54	64 m 22 s	65 m 10 s
		10	77	4 h 56 m	4 h 59 m
		12	117	43 h 35 m	43 h 45 m
		14	150	159 h 40 m	160 h 4 m

$\ell = 1024$

Experiment parameters				Results	
β	δ	m	deg.	L^2time	Total time
0.3	0.225	6	36	21 m 34 s	22 m 3 s
		8	58	2 h 46 m	2 h 48 m
		10	91	24 h 8 m	24 h 17 m
0.4	0.12	6	35	42 m 34 s	43 m 13 s
		8	54	4 h 37 m	4 h 40 m
		10	77	21 h 19 m	21 h 29 m

$\ell = 2048$

Our implementation code is not completely optimized. These are for comparing the ratio of the L^2 times to the total times. The times in these tables are the average running time of five executions. The resultant check was carried out for $m = 6$, 8 and 10.

Table 2. Total time for ℓ=512, 1024 and 2048

$j = 2(1-\beta)$ and $j = i + \tau m$, i.e., $(\frac{\tau}{1-2\beta}m, \frac{2(1-\beta)\tau}{1-2\beta}m)$, exists below the vertex point $(m, 2(1-\beta)m)$ when $1-2\beta-\tau > 0$. Thus, we separate our discussion into the case of $1-2\beta-\tau > 0$ and that of $1-2\beta-\tau \leq 0$.

First we consider the case of $1-2\beta-\tau > 0$. This case corresponds $I_3 \neq \phi$ and the following holds.

$$\begin{aligned}
\mathrm{eval}(I_1) = \sum_{i=0}^{m}\sum_{j=0}^{i}\mathrm{eval}(i,j) &= \frac{1}{6}(\beta+\delta-0.5)m^3 + o(m^3),\\
\mathrm{eval}(I_2) = \sum_{i=0}^{m}\sum_{j=i+1}^{i+\lfloor\tau m\rfloor}\mathrm{eval}(i,j) &= \frac{\tau}{4}(2\delta+\tau-1)m^3 + o(m^3),\\
\mathrm{eval}(I_3) = \sum_{i=\lfloor Bm\rfloor}^{m}\sum_{j=i+\lceil\tau m\rceil}^{\lfloor 2(1-\beta)i\rfloor}\mathrm{eval}(i,j) &= -\frac{1}{12}\frac{(1-2\beta-\tau)^3}{1-2\beta}m^3 + o(m^3).
\end{aligned}$$

From these we derive the equation (25) for $\mathrm{eval}(I) \stackrel{\mathrm{def}}{=} \mathrm{eval}(I_1) + \mathrm{eval}(I_2) + \mathrm{eval}(I_3)$.

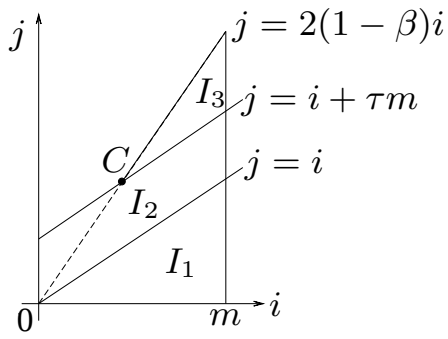

Fig. 6. I_1, I_2 and I_3

We next consider the case of $1 - 2\beta - \tau \leq 0$. This is a case of $I_3 = \phi$, thus by $\mathrm{eval}(I) = \mathrm{eval}(I_1) + \mathrm{eval}(I_2)$ we obtain (26).

7 Conclusion

We gave the new lattice construction for the lattice based attack for the RSA cryptography in the situation that d is small and LSBs of d is exposed. By this construction, the theoretical recoverable range has been improved as shown in Figure 1, which solves the open problem raised in [7]. Also as shown by our preliminary experimental results, the total efficiency of the lattice based attack has been improved significantly compared with [7]. Some more improvement, however, is necessary for using this technique with large m. One possibility is to make use of parallelization for processing the lattice basis reduction algorithm. From a theoretical view point, it would be interesting if we can understand the limitation of this approach.

Acknowledgement

I am grateful to Osamu Watanabe for his advice, careful reading, and correct some expressions. The author was supported by the Global CompView Project. This research was supported in part by JSPS Global COE program "Computationism as a Foundation for the Sciences".

References

1. Y. Aono, Degree reduction of the lattice based attack for RSA (in Japanese) Vol.107, No.24(20070419), pp. 33-40, COMP2007-5, 2007.

2. D. Boneh and G. Durfee, Cryptanalysis of RSA with private key d less than $N^{0.292}$, *IEEE Transactions on Information Theory*, vol. 46, No. 4, pp. 1339-1349, 2000.
3. D. Boneh, G. Durfee and Y. Frankel, An attack on RSA given a small fraction of the private key bits, in *Proceedings of ASIACRYPT 1998*, Lecture Notes in Computer Science, vol. 1514, pp. 25-34, 1998.
4. J. Blömer and A. May, New partial exposure attacks on RSA, in *Proceedings of CRTPTO 2003*, Lecture Notes in Computer Science, vol. 2729, pp. 27-43, 2003.
5. D. Coppersmith, Finding a small root of a univariate modular equation, in *Proceedings of EUROCRYPT 1996*, Lecture Notes in Computer Science, vol. 1070, pp. 155-165, 1996.
6. D. Coppersmith, Small solutions to polynomial equations, and low exponent RSA vulnerabilities, *Journal of Cryptology*, vol. 10, No. 4, pp. 233-260, 1997.
7. M. Ernst, E. Jochemsz, A. May, and B. Weger, Partial key exposure attacks on RSA up to full size exponents, in *Proceedings of EUROCRYPT 2005*, Lecture Notes in Computer Science, vol. 3494, pp. 371-386, 2005.
8. A. D. Healy, Resultants, Resolvents and the Computation of Galois Groups, Available online at `http://www.alexhealy.net/papers/math250a.pdf`
9. N. Howgrave-Graham, Finding small roots of univariate modular equations revisited, in *Proceedings of Cryptography and Coding*, Lecture Notes in Computer Science, vol. 1355, pp. 131-142, 1997.
10. E. Jochemz and A. May, A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants, in *Proceedings of ASIACRYPT 2006*, LNCS, vol. 4284, pp. 267-282, 2006.
11. A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász Factoring polynomials with rational coefficients, *Mathematische Annalen*, vol. 261, No. 4, pp. 515-534, 1982.
12. V. Shoup, NTL: A Library for doing Number Theory, available online at `http://www.shoup.net/ntl/index.html`
13. P. Nguyen and D. Stehlé, Floating-Point LLL revisited, in *Proceedings of EUROCRYPT 2005*, Lecture Notes in Computer Science, vol. 3494, pp. 215-233, 2006.
14. P. Nguyen and D. Stehlé, Floating-Point LLL (Full version), available online at `ftp://ftp.di.ens.fr/pub/users/pnguyen/FullL2.pdf`
15. R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptsystems, *Communications of the ACM*, vol. 21, No. 2, pp. 120-128, 1978.
16. C. P. Schnorr, A more efficient algorithm for lattice basis reduction, *Journal of algorithms*, vol. 9, No.1, pp. 47-62, 1988.
17. M. J. Wiener, Cryptanalysis of short RSA secret exponents, *IEEE Transactions on Information Theory*, vol. 36, no. 3, pp. 553-558, 1990.