Deterministic Polynomial Time Equivalence between Factoring and Key-Recovery Attack on Takagi's RSA

Noboru Kunihiro[1] and Kaoru Kurosawa[2]

[1] The University of Electro-Communications, Japan
kunihiro@ice.uec.ac.jp
[2] Ibaraki University, Japan, kurosawa@mx.ibaraki..ac.jp

Abstract. For RSA, May showed a deterministic polynomial time equivalence of computing d to factoring $N(=pq)$. On the other hand, Takagi showed a variant of RSA such that the decryption algorithm is faster than the standard RSA, where $N = p^r q$ while $ed = 1 \bmod (p-1)(q-1)$. In this paper, we show that a deterministic polynomial time equivalence also holds in this variant. The coefficient matrix T to which LLL algorithm is applied is no longer lower triangular, and hence we develop a new technique to overcome this problem.

Keywords: RSA, factoring, LLL algorithm

1 Introduction

1.1 Background

Is the key-recovery attack on RSA equivalent to factoring? This is one of the fundamental questions on RSA. Remember that in RSA, a public-key is $N(=pq)$ and e, where p and q are large primes, and the secret-key is d, where $ed = 1 \bmod (p-1)(q-1)$. Given (N, e), it is not easy to factor N from d while computing d is easy if factoring N is easy. More specifically, our problem is to find a *deterministic* polynomial time algorithm which can factor N on input the RSA parameter (N, e, d).

For this problem, there exists a *probabilistic* polynomial time algorithm [12] based on the work by Miller [10]. Miller further proved that under the Extended Riemann's Hypothesis, there exists a *deterministic* polynomial time algorithm. However, it is a strong assumption.

At Crypto 2004, May showed the first deterministic polynomial time algorithm for this problem [9] for $ed \leq N^2$ and $|p| = |q|$, where $|x|$ denotes the bit length of x. Coron and May extended this result to unbalanced p and q [4]. These results mean that the key-recovery attack on RSA is deterministically equivalent to factoring as far as $ed \leq N^2$.

On the other hand, Takagi proposed a variant of RSA [13] such that $N = p^r q$ while $ed \equiv 1 \bmod (p-1)(q-1)$. He observed that the decryption can be significantly faster in this variant. Hence it is important to study if there exists a deterministic polynomial time equivalence even in this variant.

1.2 Our Contributions

In this paper, we show a deterministic polynomial time equivalence between the key-recovery attack on Takagi's variant of RSA and factoring. More precisely, we show a deterministic polynomial time algorithm which can factor $N(= p^r q)$ from (N, e, d) such that $ed \equiv 1 \bmod (p-1)(q-1)$ if $ed \leq N^{\frac{4}{r+1}}$, $|p| = |q|$ and $r = O(\log\log N)$. It is interesting to see that May's result is obtained as a special case for $r = 1$. Hence, our result is a natural generalization of May [9].

Lenstra et al. developed an efficient lattice reduction algorithm known as LLL algorithm [8]. Based on it, Coppersmith showed a method of finding small roots of univariate modular polynomials [3] which was simplified by Howgrave-Graham [7].

May [9] and Coron and May [4] used the simplified version of Howgrave-Graham [7] to show the deterministic polynomial time equivalence on RSA. These methods first find a set of polynomials, and then apply the lattice reduction algorithm to the coefficient matrix T. It works well because T is lower triangular and hence it is easy to compute $\det T$.

We use the same approach. One of main issues of using Coron-May's strategy in the case of Takagi's RSA is the fact that the matrix T is not triangular, which makes computing the determinant a problem. We overcome this problem by using another matrix M containing polynomials $g(x, y)$, whereas the matrix T contains the polynomials $t(x, y) = g(x + A, y + B)$. We prove that determinant of T is equal to that of M. We develop a new technique to prove it and believe that our new technique will be useful for many other lattice related problems.

1.3 Related Works

Boneh, Durfee and Howgrave-Graham studied how to factor $N = p^r q$ by using lattice reduction [2]. This type of composite N is very important since it is used in EPOC [11] and ESIGN [6][3] in addition to Takagi's

[3] In EPOC and ESIGN, r is restricted in 2. And, our results give no influence to the security of EPOC and ESIGN.

variant of RSA. They showed a deterministic algorithm of finding p in time $O(p^{\frac{2}{r+1}})$. They also proved that p can be recovered in polynomial time if we can find an integer P such that $|P-p| < p^{\frac{r-1}{r+1}}$.

At Eurocrypt2005 [1], Blömer and May proposed a general method of finding small roots of bivariate polynomials over integers, and improved Boneh et al.'s result.

1.4 Organization

The rest of paper is organized as follows. The next section contains the preliminaries. First, we review LLL algorithm and Howgrave-Graham's Lemma. Then we explain Takagi's variant of RSA and describe the motivation of this research. In section 3, we introduce and prove our main theorem. In particular, we show that the deterministic polynomial time equivalence holds for $ed \leq N^{\frac{4}{r+1}}$ and $r = O(\log\log N)$. Finally, Section 4 concludes the paper.

2 Preliminaries

This section describes LLL algorithm, Howgrave-Graham's lemma and Takagi's variant of RSA.

2.1 Notation

For a vector $\boldsymbol{b}$, $||\boldsymbol{b}||$ denotes the Euclidean norm of $\boldsymbol{b}$. For a bivariate polynomial $h(x,y) = \sum h_{ij}x^i y^j$, define

$$||h(x,y)|| = \sqrt{\sum h_{ij}^2}.$$

That is, $||h(x,y)||$ denotes the Euclidean norm of the vector which consists of coefficients of $h(x,y)$.

2.2 LLL Algorithm and Howgrave-Graham's Lemma

Let $M = \{a_{ij}\}$ be a nonsingular $w \times w$ matrix of integers. The rows of M generate a lattice L, a collection of vectors closed under addition and subtraction; in fact the rows forms a basis of L. The lattice L is also represented as follows. Letting $\boldsymbol{a_i} = (a_{i1}, a_{i2}, \ldots, a_{iw})$, the lattice L

spanned by $\langle \boldsymbol{a_1}, \ldots, \boldsymbol{a_w} \rangle$ consists of all integral linear combinations of $\boldsymbol{a_1}, \ldots, \boldsymbol{a_w}$, that is :

$$L = \left\{ \sum_{i=1}^{w} n_i \boldsymbol{a_i} | n_i \in \mathbb{Z} \right\}. \tag{1}$$

LLL algorithm outputs a short vector in the lattice L. This algorithm works in deterministic polynomial time.

Proposition 1 (LLL [8]). *Let $M = \{a_{ij}\}$ be a nonsingular $w \times w$ matrix of integers. The rows of M generate a lattice L. Given M, LLL algorithm finds a vector $\boldsymbol{b} \in L$ such that*

$$||\boldsymbol{b}|| \leq 2^{(w-1)/4} (\det M)^{1/w}$$

in polynomial time in (w, B), where $B = \max \log_2 |a_{ij}|$.

Lemma 1 (Howgrave-Graham [7]). *Let $h(x, y) \in \mathbb{Z}[x, y]$ be a polynomial, which is a sum of at most w monomials. Let m be an integer. Suppose that*

1. *$h(x_0, y_0) = 0 \bmod \phi^m$, where $|x_0| < X$ and $|y_0| < Y$.*
2. *$||h(xX, yY)|| < \phi^m / \sqrt{w}$.*

Then $h(x_0, y_0) = 0$ holds over integers.

2.3 Takagi's Variant of RSA

Takagi proposed a variant of RSA such that $N = p^r q$ and showed that a faster decryption algorithm can be obtained [13, 14]. For example, for $r = 2$, it is 42% faster than the original RSA decryption algorithm.

Key Generation Generate two distinct primes p and q. Let $N = p^r q$. Find e and d such that

$$ed \equiv 1 \bmod (p-1)(q-1). \tag{2}$$

Let $d_p = d \bmod p - 1$ and $d_q = d \bmod q - 1$. Then, e and N are the encryption keys and d_p, d_q, p, q are the decryption keys.

Encryption For a plaintext $M \in \mathbb{Z}_N^*$, the ciphertext is computed as

$$C = M^e \bmod N. \tag{3}$$

Decryption Given a ciphertext C, do:

1. Compute $M_q = C^{d_q} \bmod q$, where $M_q = M \bmod q$.
2. Compute $M_p = C^{d_p} \bmod p$, where $M_p = M \bmod p$.
3. Find $M_p^{(r)}$ such that $M_p^{(r)} = M \bmod p^r$ by using Hensel lifting.
4. Compute M by applying Chinese remainder theorem to M_q and $M_p^{(r)}$.

3 Deterministic Polynomial Time Equivalence in Takagi's RSA

In this section, we show a deterministic polynomial time equivalence between the key recovery attack on Takagi's variant of RSA and factoring.

3.1 Main Theorem

We say that (r, N, e, d) is a Takagi's RSA parameter[4] if

$$N = p^r q, ed = 1 \bmod (p-1)(q-1) \text{ and } |p| = |q|.$$

We then present a deterministic polynomial time algorithm which can factor $N = p^r q$ on input such a parameter.

Theorem 1. *Suppose that a Takagi's RSA parameter* (r, N, e, d) *is given such that* $ed \leq N^{\frac{4}{r+1}}$. *Then we can factor* N *in deterministic polynomial time in* $(\log N, 2^r)$.

Corollary 1. *Suppose that a Takagi's RSA parameter* (r, N, e, d) *is given such that* $ed \leq N^{\frac{4}{r+1}}$ *and* $r = O(\log \log N)$. *Then we can factor* N *in deterministic polynomial time in* $\log N$.

Proof (of Corollary 1). Since $r = O(\log \log N)$, $2^r < (\log N)^c$ for some constant c. Then the running time of the factoring algorithm given by Theorem 1 is bounded by a polynomial time in $\log N$. □

Remark 1. Let $r = 1$ in Theorem 1. Then we obtain the following corollary: Given (N, e, d), $N = pq$ can be factorized in deterministic polynomial time in $\log N$ if $ed \leq N^2$. This corollary coincides with the result of May [9] and Coron and May [4] for balanced p and q. Hence, our result is a natural generalization of their result on RSA.

Remark 2. In Takagi's variant of RSA, since $ed = 1 \bmod (p-1)(q-1)$, e and d are usually chosen in such a way that $e < (p-1)(q-1)$ and $d < (p-1)(q-1)$. In this case, it holds that

$$ed < ((p-1)(q-1))^2 \leq (pq)^2 \approx N^{\frac{4}{r+1}}.$$

Therefore, our bound is achieved for e and d that are chosen in the usual way.

[4] We omit the discussion of unbalanced prime factors due to limitations of space. We can easily extend our analysis to unbalanced case as Coron-May's paper [4]. In the Takagi's original paper[13], e and d are set as $ed \equiv 1(\bmod\ \mathrm{lcm}(p-1, q-1))$. In this case, we have the same result if $\gcd(p-1, q-1)$ is small or known.

Remark 3. The condition $r < c \log\log N$ leads to another equivalent condition: $r < c'(\log\log p + \log\log\log p)$ for some c'. On the other hand, Boneh et al. proved that if $r > c'' \log p$, N can be factorized in deterministic polynomial time of $\log N$ without the knowledge of d [2]. Consequently, the computational cost is not known when $c'(\log\log p + \log\log\log p) < r < c'' \log p$. But, this is a purely mathematical interest.

3.2 Affine Transform Lemma

We now prove an elemental lemma which plays an important role in the proof of Theorem 1. We believe that this lemma will be useful for many other lattice related problems.

Lemma 2. *Let $g_1(x), \cdots g_r(x)$ be r polynomials of degree $r-1$. For each $g_i(x)$, define $t_i(x)$ as*

$$t_i(x) = g_i(x+\alpha),$$

where α is an arbitrary constant. Let $M = (g_{ij})$ be the $r \times r$ coefficient matrix of $g_1(x), \cdots g_r(x)$, where

$$g_i(x) = \sum_{j=1}^{r} g_{ij} x^{r-j},$$

and let $T = (t_{ij})$ be the $r \times r$ coefficient matrix of $t_1(x), \cdots t_r(x)$, where

$$t_i(x) = \sum_{j=1}^{r} t_{ij} x^{r-j}.$$

Then it holds that

$$\det T = \det M.$$

Proof. It holds that

$$\begin{aligned} t_i(x) = g_i(x+\alpha) &= \sum_{u=1}^{r} g_{iu}(x+\alpha)^{r-u} = \sum_{u=1}^{r}\sum_{v=0}^{r-u} g_{iu} \times {}_{r-u}C_v \alpha^{r-u-v} x^v \\ &= \sum_{u=1}^{r}\sum_{j=u}^{r} g_{iu} \times_{r-u} C_{j-u} \alpha^{j-u} x^{r-j} \end{aligned}$$

Therefore, we obtain that

$$t_{ij} = \sum_{u=1}^{j} g_{iu} \times {}_{r-u}C_{j-u} \alpha^{j-u}. \tag{4}$$

Next, define an upper triangular $r \times r$ matrix $A = (a_{ij})$ as follows.

$$a_{ij} = \begin{cases} {}_{r-i}C_{j-i}\alpha^{j-i} & \text{if } i \leq j \\ 0 & \text{if } i > j. \end{cases}$$

Then we can see that $T = MA$. Further, we have $\det A = 1$ because $a_{ii} = {}_{r-i}C_0 \times \alpha^0 = 1$. Consequently we obtain that $\det T = \det M \times \det A = \det M$. □

3.3 Proof of Theorem 1

We will factor N by using the following strategy. Let X, Y, m, t be positive integers which will be determined later.

Step 0 Let $p = p_0 X + x_0$ and $q = q_0 Y + y_0$, where $x_0 < X$ and $y_0 < Y$. Suppose that p_0 and q_0 are known, and we want to compute x_0 and y_0.

Step 1 Construct a set of polynomials $t_{ijk}(x, y)$ such that

$$t_{ijk}(x_0, y_0) \equiv 0 \bmod (((p-1)(q-1))^m).$$

Step 2 Apply LLL algorithm to the coefficient matrix of $\{t_{ijk}(x, y)\}$ to obtain $h(x, y)$, where $h(x, y)$ is a non-zero integer combination of $t_{ijk}(x, y)$ with small coefficients.

Step 3 Let

$$h'(x) = h\left(x, \frac{N}{(p_0 X + x)^r} - q_0 Y\right)$$

Then x_0 is a solution of $h'(x) = 0$.

We will find p_0 and q_0 by exhaustive search in Step 0. In what follows, we will show how to construct polynomials t_{ijk}, how to compute the determinant of the coefficient matrix of $\{t_{ijk}\}$ and how to determine X, Y, m, t. It will be seen that the above algorithm runs in polynomial time in $(\log N, 2^r)$ if $\max(p/X, q/Y)$ is polynomially bounded because Step 1 $\sim$ Step 3 are computed in polynomial time and p_0 and q_0 are bounded by $\max(p/X, q/Y)$.

Remark 4. $h'(x)$ is not identically zero since $h(x, y)$is not identically zero.

How to construct t_{ijk}. Let

$$f(x,y)=(x-1)(y-1).$$

Note that $f(p,q)=(p-1)(q-1)$ is the modulus of Eq.(2). Let

$$U=ed-1,\ S=(p-1)(q-1).$$

Define

$$g_{ijk}(x,y)=x^i y^j f(x,y)^k U^{m-k}.$$

Then it is easy to see that

$$g_{ijk}(p,q)=p^i q^j f(p,q)^k U^{m-k}=0 \bmod S^m$$

for any (i,j,k). In $g_{ijk}(x,y)$, we will replace each occurrence of $x^r y$ by N because $N=p^r q$ (based on the Durfee-Nguyen technique [5]). Therefore, the resulting $g_{ijk}(x,y)$ contains monomials of the form x^a, y^b and $xy^{c_1}, x^2y^{c_2}, \ldots, x^{r-1}y^{c_{r-1}}$ for some $a, b, c_1, \ldots$ and c_{r-1}.

Construct a list of polynomials $G=(g_{ijk})$ as follows, where s, t will be determined later.

```
$G \leftarrow \emptyset$
for $k=0,\cdots,m-1$, do;
    append $g_{0,0,k}$ and $g_{1,0,k}$ into $G$ in this order.
    for $i=r-1,\cdots,1$, do;  append $g_{i,1,k}$ to $G$.
for $i=0,\cdots,s$, do;  append $g_{i,0,m}$ to $G$.
for $j=1,\cdots,t$, do;
    for $i=r-1,\cdots,0$, do;  append $g_{i,j,m}$ to $G$.
return $G$.
```

Express each g_{ijk} as follows, where the leading monomial appears in the right most term of the right hand side. (For more details, see Appendix A.)

$$\begin{array}{rcl}
g_{0,0,0}(x,y) & = & U^m \\
g_{1,0,0}(x,y) & = & *** + xU^m \\
g_{r-1,1,0}(x,y) & = & *** + x^{r-1}yU^m \\
 & \vdots & \\
g_{1,1,0}(x,y) & = & *** + xyU^m \\
\hline
 & \vdots &
\end{array}$$

– –

$$\begin{aligned}
g_{0,0,m-1}(x,y) &= *** + y^{m-1}U \\
g_{1,0,m-1}(x,y) &= *** + x^{m}UX^{m} \\
g_{r-1,1,m-1}(x,y) &= *** + x^{r-1}y^{m}U \\
&\vdots \\
g_{1,1,m-1}(x,y) &= *** + xy^{m}U
\end{aligned}$$

– –

$$\begin{aligned}
g_{0,0,m}(x,y) &= *** + y^{m} \\
g_{1,0,m}(x,y) &= *** + x^{m+1} \\
&\vdots \\
g_{s,0,m}(x,y) &= *** + x^{m+s}
\end{aligned}$$

– –

$$\begin{aligned}
g_{r-1,1,m}(x,y) &= *** + x^{r-1}y^{m+1} \\
&\vdots \\
g_{0,1,m}(x,y) &= *** + y^{m+1}
\end{aligned}$$

– –

$$\vdots$$

– –

$$\begin{aligned}
g_{r-1,t,m}(x,y) &= *** + x^{r-1}y^{m+1} \\
&\vdots \\
g_{0,t,m}(x,y) &= *** + y^{m+t}
\end{aligned}$$

Next define

$$t_{ijk}(x,y) = g_{ijk}(p_0X + x, q_0Y + y). \qquad (5)$$

It is easy to see that

$$t_{ijk}(x_0,y_0) = g_{ijk}(p_0X + x_0, q_0Y + y_0) \equiv g_{ijk}(p,q) \equiv 0(\mathrm{mod} S^m).$$

We have now finished Step 1.

How to compute det T. Let M be the coefficient matrix of $\{g_{ijk}(xX,yY)\}$ and T be the coefficient matrix of $\{t_{ijk}(xX,yY)\}$. Tables 1 and 2 show small examples.

We want to apply Proposition 1 to T, where we need to know $\det T$. However, computing $\det T$ is not easy because T is not lower triangular. (See from Table 1.) This is the big difference from the previous works [4, 9]. We prove the following lemma based on Lemma 2.

Table 1. Example of T for $r = 2, m = 3, s = 2, t = 2$.

	1	x	xy	y	x^2	xy^2	y^2	x^3	xy^3	y^3
$t_{000}(xX, yY)$	U^3									
$t_{100}(xX, yY)$	$*$	U^3X								
$t_{110}(xX, yY)$	$*$	$*$	U^3XY	U^3XYp_0						
$t_{001}(xX, yY)$	$*$	$*$	U^2XY	$U^2(p_0X - 1)Y$						
$t_{101}(xX, yY)$	$*$	$*$	$*$	$*$	$-U^2X^2$					
$t_{111}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$-U^2XY^2$	$-U^2XY^2p_0$			
$t_{002}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$-2UXY^2$	$UY^2(1 - 2p_0X)$			
$t_{102}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	UX^3		
$t_{112}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	UXY^3	UXY^3p_0
$t_{003}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$3XY^3$	$Y^3(3p_0X - 1)$

	$1 \cdots y^3$	x^4	x^5	xy^4	y^4	xy^5	y^5
$t_{103}(xX, yY)$	$*$	$-X^4$					
$t_{203}(xX, yY)$	$*$	$*$	$-X^5$				
$t_{113}(xX, yY)$	$*$	$*$	$*$	$-XY^4$	$-p_0XY^4$		
$t_{013}(xX, yY)$	$*$	$*$	$*$	$3XY^4$	$(3p_0X - 1)Y^4$		
$t_{123}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$-XY^5$	$-p_0XY^5$
$t_{023}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$-4XY^5$	$(1 - 4p_0X)Y^5$

Table 2. Example of M for $r = 2, m = 3, s = 2, t = 2$.

	1	x	xy	y	x^2	xy^2	y^2	x^3	xy^3	y^3	x^4	x^5	xy^4	y^4	xy^5	y^5
$g_{000}(xX, yY)$	U^3															
$g_{100}(xX, yY)$	$*$	U^3X														
$g_{110}(xX, yY)$	$*$	$*$	U^3XY													
$g_{001}(xX, yY)$	$*$	$*$	$*$	$-U^2Y$												
$g_{101}(xX, yY)$	$*$	$*$	$*$	$*$	$-U^2X^2$											
$g_{111}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$-U^2XY^2$										
$g_{002}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	UY^2									
$g_{102}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	UX^3								
$g_{112}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	UXY^3							
$g_{003}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-Y^3$						
$g_{103}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-X^4$					
$g_{203}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-X^5$				
$g_{113}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-XY^4$			
$g_{013}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-Y^4$		
$g_{123}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-XY^5$	
$g_{023}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-Y^5$

Lemma 3. *It holds that*

$$\det T = \det M. \tag{6}$$

Proof. For $1 \leq j \leq m+t$, define r polynomials $f_{1,j}, \cdots, f_{r,j}$ of degree $r-1$ as follows.

- For $1 \leq j \leq m$,
  - $f_{a,j}(x)$ is the coefficient of y^j in $g_{r-a,1,j-1}(xX, yY)$ for $1 \leq a \leq r-1$.
  - $f_{r,j}(x)$ is the coefficient of y^j in $g_{0,0,j}(xX, yY)$.
- For $m+1 \leq j \leq m+t$,
  - $f_{a,j}(x)$ is the coefficient of y^j in $g_{r-a,j-m,m}(xX, yY)$ for $1 \leq a \leq r$.

Similarly, define r polynomials $e_{1,j}, \cdots, e_{r,j}$ of degree $r-1$ as follows.

- For $1 \leq j \leq m$,
  - $e_{a,j}(x)$ is the coefficient of y^j in $t_{r-a,1,j-1}(xX, yY)$ for $1 \leq a \leq r-1$.
  - $e_{r,j}(x)$ is the coefficient of y^j in $t_{0,0,j}(xX, yY)$.
- For $m+1 \leq j \leq m+t$,
  - $e_{a,j}(x)$ is the coefficient of y^j in $t_{r-a,j-m,m}(xX, yY)$ for $1 \leq a \leq r$.

Let M_j be the $r \times r$ coefficient matrix of $f_{1,j}, \cdots, f_{r,j}$, and T_j be the $r \times r$ coefficient matrix of $e_{1,j}, \cdots, e_{r,j}$.

For example, T_1, M_1, T_2 and M_2 of Table 1 and 2 are as follows.

$$T_1 = \begin{pmatrix} U^3XY, & U^3XYp_0 \\ U^2XY, & U^2Y(p_0X-1) \end{pmatrix}, M_1 = \begin{pmatrix} U^3XY, & 0 \\ U^2XY, & -U^2Y \end{pmatrix}.$$

$$T_2 = \begin{pmatrix} -U^2XY^2, & -U^2XY^2p_0 \\ -2UXY^2, & UY^2(1-2p_0X) \end{pmatrix}, M_2 = \begin{pmatrix} -U^2XY, & 0 \\ -2UXY^2, & UY^2 \end{pmatrix}.$$

From Eq.(5), we obtain that

$$t_{ijk}(xX, yY) = g_{ijk}(p_0X + xX, q_0Y + yY) = g_{ijk}(X(x+p_0), Y(y+q_0)).$$

Hence, it is easy to see that $e_{i,j}(x) = f_{i,j}(x+p_0)$ because y^j is the highest term in $g_{r-a,1,j-1}(xX, yY)$, $g_{0,0,j}(xX, yY)$ and $g_{r-a,j-m,m}(xX, yY)$. Therefore, from Lemma 2, we obtain that $\det T_j = \det M_j$ for $1 \leq j \leq m+t$. Consequently, we can see that $\det T = \det M$. □

Since M is a triangular matrix, we can compute $\det M$ easily as follows[5].

$$\det M = U^{(r+1)m(m+1)/2} \cdot X^{(m+s)(m+s+1)/2 + r(r-1)(m+t)/2} \cdot Y^{r(m+t)(m+t+1)/2}$$

[5] In what follows, we omit the sign of $\det M$.

Applying LLL. Note that T and M are $w \times w$ matrices, where

$$w = (r+1)m + (s+1) + rt = (r+1)m + s + rt + 1.$$

Now by applying LLL algorithm to T, we can obtain

$$h(x,y) = \sum a_{ijk} t_{ijk}(x,y)$$

such that

$$||h(xX,yY)|| \le 2^{(w-1)/4} (\det M)^{1/w}$$

for some integers a_{ijk}. From the definition of $t_{ijk}(x,y)$, it holds that

$$h(x_0,y_0) = \sum a_{ijk} t_{ijk}(x_0,y_0) = 0 \bmod S^m.$$

Therefore, if $||h(xX,yY)|| < S^m/\sqrt{w}$, then from Howgrave-Graham's lemma, we have $h(x_0,y_0) = 0$ over integers. Therefore, it is sufficient to show that

$$2^{(w-1)/4} (\det M)^{1/w} < \frac{S^m}{\sqrt{w}}. \tag{7}$$

Since p and q are the same bit length, it satisfies that $S = (p-1)(q-1) > pq/2 > \max(p^2,q^2)/4 > N^{2/(r+1)}/4$. Using the inequality $\sqrt{w} \le 2^{(w-1)/2}$, we obtain the following sufficient condition:

$$\det M < N^{\frac{2mw}{r+1}} 2^{-(2mw + \frac{3}{4}w(w-1))}. \tag{8}$$

How to determine X and Y By setting $X = Y$ and $s = t$, $\det M$ can be simplified as

$$\det M = U^{(r+1)m(m+1)/2} \cdot X^{(r+1)(m+s)(m+s+1)/2 + r(r-1)(m+s)/2}. \tag{9}$$

The dimension of the lattice is given as $w = (r+1)(m+s)+1$.

Since it holds that $U \le N^{\frac{4}{r+1}}$ from our assumption, we obtain

$$\begin{aligned} \det M &\le N^{\frac{(r+1)m(m+1)}{2} \cdot \frac{4}{r+1}} \cdot X^{(r+1)(m+s)(m+s+1)/2 + r(r-1)(m+s)/2} \\ &= N^{2m(m+1)} \cdot X^{(r+1)(m+s)(m+s+1)/2 + r(r-1)(m+s)/2}. \end{aligned} \tag{10}$$

From inequalities (8) and (10) we obtain

$$\begin{aligned} N^{2m(m+1)} \cdot X^{(r+1)(m+s)(m+s+1)/2 + r(r-1)(m+s)/2} &\le N^{2mw/(r+1)} \cdot 2^{-(2mw + \frac{3}{4}w(w-1))} \\ X^{(r+1)(m+s)(m+s+1)/2 + r(r-1)(m+s)/2} &\le N^{2m(s + \frac{1}{r+1} - 1)} 2^{-(2mw + \frac{3}{4}w(w-1))}. \end{aligned}$$

The above inequality can be transformed into

$$X \leq N^{2m\frac{s+1/(r+1)-1}{(r+1)(m+s)(m+s+1)/2+r(r-1)(m+s)/2}} \cdot 2^{-\frac{2mw+3w(w-1)/4}{(r+1)(m+s)(m+s+1)/2+r(r-1)(m+s)/2}}. \tag{11}$$

Letting

$$\gamma(m,s;r) = 2m\frac{s+1/(r+1)-1}{(r+1)(m+s)(m+s+1)/2+r(r-1)(m+s)/2}$$

and

$$\delta(m,s;r) = \frac{2mw+3w(w-1)/4}{(r+1)(m+s)(m+s+1)/2+r(r-1)(m+s)/2},$$

we can express inequality (11) as

$$X \leq N^{\gamma(m,s;r)}2^{-\delta(m,s;r)}. \tag{12}$$

The next thing to do is to find s which maximize $\gamma(m,s;r)$ for a fixed m to maximize the bound X on x_0. Such s is given by $s = m$. In this setting, $\gamma(m,m;r)$ is calculated as follows.

$$\gamma(m,m;r) = 2\frac{m+1/(r+1)-1}{2(r+1)m+r^2+1} = \frac{1}{r+1} - \frac{r+1}{2(r+1)m+r^2+1}$$

$\delta(m,m;r)$ is calculated as

$$\begin{aligned}\delta(m,m;r) &= \frac{2mw+3w(w-1)/4}{(r+1)(m+m)(m+m+1)/2+r(r-1)(m+m)/2} \\ &= \frac{(2(r+1)m+1)(\frac{3}{2}(r+1)+2)}{2(r+1)m+1+r^2} < \frac{1}{2}(3r+5).\end{aligned}$$

From the above discussion, we obtain

$$X \leq 2^{-\frac{(3r+5)}{2}}N^{\frac{1}{r+1}-\frac{r+1}{2(r+1)m+r^2+1}}. \tag{13}$$

Total Computational time Taking the largest integer X of inequality (13), we obtain

$$\frac{p}{X} < \frac{2N^{1/(r+1)}}{X} \leq N^{\frac{r+1}{2(r+1)m+r^2+1}} \cdot 2^{\frac{3r+7}{2}}. \tag{14}$$

By setting $m = \lfloor \log N \rfloor$, we obtain

$$\frac{p}{X} < O(1) \cdot 2^{\frac{3r+7}{2}}. \tag{15}$$

Hence, the number of repetition for selection of p_0 is upper bounded by a polynomial of 2^r.

The dimension of the lattice is given by $w = (r+1)m + s + rt + 1 = 2(r+1)m + 1 = O(\log N)$. The maximum entry of the lattice is given by $N^{\frac{4m}{r+1}+1}$. This implies that the logarithm of the maximum entry is given by $O(\log N^{\frac{4m}{r+1}}) = O(m \log N) = O((\log N)^2)$. Hence, the total computation cost for the bivariate polynomial $h(x, y)$ is given by the polynomial of $(\log N, 2^r)$. Note that LLL algorithm works deterministically.

The rest of our algorithm works in deterministic polynomial time of $\log N$. From the above discussion, N can be factorized in deterministic polynomial time of $\log N$ and 2^r. □

4 Concluding Remarks

We used the same approach as Coron-May[4]. But, Theorem 1 cannot be obtained trivially from[4]. We had to overcome the following two difficulties in order to prove our theorem.

1. How should we arrange the order of polynomials g_{ijk} and monomials so that M is triangular?
2. How should we calculate $\det T$? Since T is not triangular, calculation of determinant seems difficult.

First, we explain how to overcome the first difficulty. In the analysis of standard RSA [4], each occurrence of xy is replaced by N because $N = pq$. Hence only x^a or y^b appears in the resulting $g_{i,j,k}$ which makes it easy to form a triangular matrix.

On the other hand, we replace each occurrence of $x^r y$ by N because $N = p^r q$ in Takagi's RSA. Then the resulting $g_{ijk}(x, y)$ contains monomials of the form x^a, y^b and $xy^{c_1}, x^2y^{c_2}, \ldots, x^{r-1}y^{c_{r-1}}$ for some $a, b, c_1, \ldots$ and c_{r-1}. A technical difficulty is how to make a triangular matrix M from these $g_{i,j,k}$. We have given an efficient solution for this problem.

Remark 5. We can apply Blömer-May's method [1] to our problem. In this method, however, the lattice is uniquely determined by the Newton polygon of the target polynomial $f(x, y)$, and hence there is no room for replacing $x^r y$ with N. Consequently we would get a smaller range of *ed*.

Next, we explain how to overcome the second difficulty. Since the only monomials x^a and y^b appear in Coron-May's g_{ijk}, the matrix generated from t_{ijk} is naturally triangular. Hence, in Coron-May's case, the determinant of T can be easily obtained. However, in our polynomials g_{ijk}, the monomial $x^i y^j$ appears. Hence, our matrix T cannot be triangular (for example, see Table 1). By showing Lemma 3 (that is $\det T = \det M$), we overcome this problem. In the proof of Lemma 3, Lemma 2 plays an important role. Note that in proof of lemma 3, we did not use the property that M is triangular. We enjoy this property in calculating $\det M$. We believe that our new technique will be useful for many other lattice related problems.

Acknowledgment

This research was supported in part by the Grants-in-Aid No. 16500009 for Scientific Research, JSPS. We thank the anonymous reviewers for their helpful comments.

References

1. J. Blömer and A. May, "A Tool Kit for Finding Small Roots of Bivariate Polynomials over the Integers," Proc. of Eurocrypt2005, LNCS 3494, pp. 251–267, 2005.
2. D.Boneh, G.Durfee and N.Howgrave-Graham, "Factoring $N = p^r q$ for Large r," in Proc. of Crypto'99, LNCS 1666, pp. 326–337, 1999.
3. D. Coppersmith, "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities," J. Cryptology 10(4): 233-260, 1997.
4. J.S. Coron and A.May, "Deterministic Polynomial Time Equivalence of Computing the RSA Secret Key and Factoring," IACR ePrint Archive: Report 2004/208, 2004, to appear in Journal of Cryptology.
5. G. Durfee and P. Nguyen, "Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt'99," Proc. of Asiacrypt2000, LNCS 1976, pp. 14–29, 2000.
6. A. Fujioka, T. Okamoto and S. Miyaguchi, "ESIGN: An Efficient Digital Signature Implementation for Smart Cards," In Proc. of Eurocrypt'91, LNCS 547, pp.446-457, 1992.
7. N.Howgrave-Graham, "Finding Small Roots of Univariate Modular Equations Revisited," IMA Int. Conf., pp.131–142 (1997)
8. A.K. Lenstra, H.W. Lenstra, L. Lovász, "Factoring polynomials with rational coefficients," Mathematische Annalen 261, pp.515–534, 1982.
9. A. May, "Computing the RSA Secret Key Is Deterministic Polynomial Time Equivalent to Factoring," in Proc. of Crypto2004, LNCS 3152, pp. 213–219, 2004.
10. G. L. Miller, "Riemann's Hypothesis and Tests for Primality," Seventh Annual ACM Symposium on the Theory of Computing, pp. 234–239, 1975.
11. T. Okamoto and S. Uchiyama, "A New Public Key Cryptosystem as secure as factoring," in Proc. of Eurocrypt'98, LNCS 1403, pp. 310–318, 1998.
12. R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, vol. 21(2), pp. 120–126, 1978.
13. T. Takagi, "Fast RSA-Type Cryptosystem Modulo $p^k q$, " in Proc. of Crypto'98, LNCS 1462, pp.318–326, 1998.
14. T. Takagi, "A Fast RSA-Type Public-Key Primitive Modulo $p^k q$ Using Hensel Lifting," IEICE Trans. Fundamentals, Vol. 87-A, no. 1, pp. 94–101, 2004.

A Our matrix M is triangular

In this section, we describe the matrix M of Sec.3.2 more formally, and show that it is a lower triangular matrix.

We say that $g_{i,j,k}$ is the ℓth polynomial of G if it is the ℓth polynomial that is appended to G by the algorithm of Sec.3.2. For a monomial $x^a y^b$ which is included in the ℓth $g_{i,j,k}$, we say that $x^a y^b$ appears here first if it does not appear in the first $(\ell-1)$ polynomials of G. Let $d_k = (k-1)(r+1)$. (Note that each occurrence of $x^r y$ is replaced by N.)

Lemma 4. *$g_{0,0,k}$ is the $(d_k + 1)$th polynomial of G, and y^k appears here first for $1 \le k \le m$.*

Lemma 5. *$g_{1,0,k}$ is the $(d_k + 2)$th polynomial of G, and x^{k+1} appears here first for $0 \le k \le m$.*

Lemma 6. *$g_{r-i,1,k}$ is the $(d_k + i + 2)$th polynomial of G, and $x^{r-i}y^{k+1}$ appears here first for $1 \le i \le r-1$ and $0 \le k \le m-1$.*

Lemma 7. *$g_{i,0,m}$ is the (d_m+i+1)th polynomial of G, and x^{i+m} appears here first for $1 \le i \le s$.*

Lemma 8. *$g_{0,j,m}$ is the (d_m+s+j)th polynomial of G, and x^{j+m} appears here first for $1 \le j \le t$.*

Consider an expression of $g_{i,j,k}$ as follows.

- The leading monomial of $g_{0,0,k}$ is y^k for $1 \le k \le m$.
- The leading monomial of $g_{1,0,k}$ is x^{k+1} for $0 \le k \le m$.
- The leading monomial of $g_{r-i,1,k}$ is $x^{r-i}y^{k+1}$ for $1 \le i \le r-1$ and $0 \le k \le m-1$.
- The leading monomial of $g_{i,0,m}$ is x^{i+m} for $1 \le i \le s$.
- The leading monomial of $g_{0,j,m}$ is x^{j+m} for $1 \le j \le t$.

Lemma 9. *In the ℓth $g_{i,j.k}$, all the monomials other than the leading one appears in some polynomial of $G_{\ell-1}$.*

Let M be a $w \times w$ matrix such that ℓth row consists of the coefficients of the ℓth $g_{i,j,k}$ of G, where the leading monomial of each $g_{i,j,k}$ is given as above. Then it is easy to see that M is a lower triangular matrix from the above lemmas.

The proofs of the lemmas will be given in the full version.