Solving Generalized Small Inverse Problems

Noboru Kunihiro

The University of Tokyo, Japan
kunihiro@k.u-tokyo.ac.jp

Abstract. We introduce a "generalized small inverse problem (GSIP)" and present an algorithm for solving this problem. GSIP is formulated as finding small solutions of $f(x_0, x_1, \ldots, x_n) = x_0 h(x_1, \ldots, x_n) + C = 0 (\bmod\ M)$ for an n-variate polynomial h, non-zero integers C and M. Our algorithm is based on lattice-based Coppersmith technique. We provide a strategy for construction of a lattice basis for solving $f = 0$, which are systematically transformed from a lattice basis for solving $h = 0$. Then, we derive an upper bound such that the target problem can be solved in polynomial time in $\log M$ in an explicit form. Since GSIPs include some RSA-related problems, our algorithm is applicable to them. For example, the small key attacks by Boneh and Durfee are re-found automatically. This is a full version of [13].

Keywords: LLL algorithm, small inverse problem, RSA. lattice-based cryptanalysis

1 Introduction

Since the seminal work of Coppersmith [3–5], many cryptanalysis have been proposed by using his technique which is based on LLL algorithm. The first typical application is a small secret exponent attack on RSA proposed by Boneh and Durfee [2]. The second is a proof of deterministic polynomial time equivalence between computing the RSA secret key and factoring [6, 16].

In RSA [18], the small secret exponent d is commonly used to speed up the decryption or signature generation. In 1990, Wiener showed that when $d \leq \frac{1}{3} N^{1/4}$, the RSA moduli N can be factored in polynomial time [20]. Then, in 1999, Boneh and Durfee [2] improved the Wiener's bound to $d \leq N^{0.284}$. Furthermore, they proved that N can be factored in polynomial time when $d \leq N^{0.292}$. In their attack, lattice reduction algorithms such as LLL algorithm [14] play an important role. Let us briefly describe their attack. First, they reduce small secret exponent attack to solving a bivariate modular equation:

$$x(A + y) = 1 \ (\bmod\ e),$$

where A is a given integer and the solution $(x, y) = (\bar{x}, \bar{y})$ satisfies $|\bar{x}| < e^{\delta}$ and $|\bar{y}| < e^{1/2}$. They referred this problem as "small inverse problem." Then, they proposed a polynomial time algorithm for solving this problem. They obtained the condition on δ such that the algorithm outputs the solution. This leads to the weaker bound: $d \le N^{0.284}$ and the stronger bound: $d \le N^{0.292}$. By extending their (weaker) algorithm, Durfee and Nguyen showed cryptanalysis on some variants of RSA with short secret exponent [7]. They proposed an algorithm for solving trivariate modular equation $f(x, y, z) = x(A + y + z) + 1 = 0 \pmod{e}$ with constraint $yz = N$ in their analysis. It is crucial in their algorithm how to handle the constraint $yz = N$. To do so, they introduced so-called "Durfee-Nguyen technique."

May (and Coron-May) proved that if the RSA secret key d is revealed, the RSA moduli N can be factored in *deterministic* polynomial time [6, 16]. We will focus on the Coron-May's proof [6] rather than May's original proof [16]. Consider a univariate modular equation: $h(y) \equiv A + y = 0 \pmod{S}$, where S is an unknown divisor of a known positive integer U and A is a known positive integer. They showed a deterministic polynomial time algorithm which solves the equation for $S \le U^{1/2}$ to prove that (balanced) RSA moduli N can be factored deterministically when d is revealed. They extended their result to the unbalanced RSA case [6]. They showed the condition that the bivariate modular equation: $h(y, z) \equiv A + y + z = 0 \pmod{S}$ with constraint $yz = N$, where S and U are in the same setting as the balanced RSA.

1.1 Our Contribution

In this paper, we introduce "generalized small inverse problem (GSIP)" for an $n+1$-variate equation. Let f be an $n+1$-variate polynomial by

$$f(x_0, x_1, \ldots, x_n) = x_0 h(x_1, \ldots, x_n) + C$$

for an n-variate polynomial h and a non-zero integer C. Let M be a positive integer whose prime factors are unknown. Suppose that the solution of $f = 0 \pmod{M}$ satisfies $|\bar{x_0}| < X_0, |\bar{x_1}| < X_1, \ldots, |\bar{x_n}| < X_n$ for fixed positive integers $X_0, X_1, \ldots, X_n$. Then, one wants to find the solution: $(x_0, x_1, \ldots, x_n) = (\bar{x_0}, \bar{x_1}, \ldots, \bar{x_n})$. Some cases may have constraints between variables $x_1, \ldots, x_n$. When $C = 1$, the problem can be viewed as follows: given a function $h(x_1, x_2, \ldots, x_n)$, find small elements $(\bar{x_1}, \ldots, \bar{x_n})$ such that the inverse of $-h(\bar{x_1}, \bar{x_2}, \ldots, \bar{x_n})$ modulo M is "small". So, we call this problem as generalized small inverse problem. Classical "small inverse problem" [2] corresponds to $n = 1$, $h(x_1) = A + x_1$ and $C = 1$, where A is a given integer. GSIP is not only a natural extension of classical small inverse problem, but also is applicable to many RSA-related cryptanalysis. In our paper, we are concerned with only modular equations not integer equations.

Second, we propose a polynomial time algorithm for solving this problem. Our algorithm is based on Coppersmith's approach [3] and has the following property in the lattice basis construction:

1. First, construct a lattice basis for solving $h(x_1, \ldots, x_n) = 0 \pmod{p}$, where p is an unknown divisor of known integer N.
2. Then, construct a lattice basis for $f(x_0, \ldots, x_n) = 0 \pmod{M}$ by employing a lattice basis for h.

We introduce 4 restrictions for a lattice in solving $h = 0$. Since many methods in the literature hold these restrictions, they are not too strong restrictions. Then, we propose a simple but effective compiler which transforms a lattice basis for $h = 0$ to that for $f = x_0h + C = 0$ **(Compiler)**. Our compiler works if a lattice for $h = 0$ holds the 4 restrictions. It gives a good insight in construction of a lattice basis for f.

Our compiler is applicable to many kinds of cryptanalysis. For example, we can re-find Boneh-Durfee's small secret exponent attack on RSA [2] by using our compiler and the lattice employed in the proof for deterministic polynomial time equivalence [6]. That is, our compiler builds a bridge between these two works. It is the first time to point out this kind of connection as far as we know. Our compiler is especially effective when one needs to construct a special type of lattice. Suppose that some variables have constraint, ex. $yz = N$. In this case, it is well known that Durfee-Nguyen technique is effective [7]. If one can construct a good lattice for n-variate equation: $h = 0$ built Durfee-Nguyen technique into, one has also a good lattice for $n + 1$-variate equation: f built Durfee-Nguyen technique into. In general, the more variables are involved, the harder the construction of a good lattice is. If one uses our compiler, one just constructs a lattice basis for h not for f. Hence, one can more easily construct a good lattice basis for f.

Next, we obtain the upper bound of the solution such that the equation: $f(x_0, x_1, \ldots, x_n) = 0 \pmod{M}$ is solvable in polynomial time in $\log M$ (but not in n) (Lemma 5 and Theorem 2). That means, letting the solution be $(\bar{x_0}, \ldots, \bar{x_n})$ and positive integers $X_0, \ldots, X_n$, when $|\bar{x_i}| < X_i$ for each i, one can solve the problem in polynomial time. In deriving X_i, one needs not tedious computation. In particular, when $X_1, \ldots, X_n$ are fixed, one can easily obtain the upper bound of solution X_0.

In Boneh-Durfee's [2] and Durfee-Nguyen's analyses [7], tedious computations are needed. Furthermore, their computations are not applicable to the other kind of attacks. We generalize this kind of calculation to obtain the evaluation formula, which is easy to use and covers many kind of cryptanalysis including Boneh-Durfee's. Hence, we provide another type of "toolkit" for (especially RSA-related) cryptanalysis from that of Blömer-May [1].

Our Strategies vs. General Strategies for Construction of Lattice Basis It is well known that the shape of Newton polytope of a polynomial to be solved is important. This is suggested by Coppersmith [4] and fully explained by Blömer and May in the case of bivariate integer equation [1]. For general polynomials, Jochemsz and May proposed general methods for construction of optimal lattice basis [11]. Although their method is general and effective, it cannot handle constrained variables case. Actually, when Durfee-Nguyen technique is involved, their method could not generate a good lattice. Using our compiler, Durfee-Nguyen technique is automatically involved in constructing the lattice for f if it is involved in the lattice for h. Our compiler is especially effective for specific type of equations and is applicable to many kinds of RSA-related cryptanalysis.

1.2 Organization

Section 2 gives preliminaries. In Section 3, we show how to solve the "generalized small inverse problems." First, we introduce 4 restrictions for a lattice in solving $h = 0$. Then, we give a compiler which transforms a lattice basis for $h(x_1, \ldots, x_n) = 0$ into that for $f(x_0, x_1, \ldots, x_n) = x_0 h(x_1, \ldots, x_n) + C = 0$. In Section 4, we evaluate the volume of lattice for f and derive the condition among upper bounds of solutions. In Section 5, we argue application of our compiler to GSIP and give details of an application: the small secret exponent attack to RSA, which shows the effectiveness of our compiler. Section 6 concludes the paper. Some of proofs are given in Appendix A. Some of examples are given in Appendix B.

2 Preliminaries

2.1 Small Secret Exponent Attack on RSA [2]

Let (N, e) be a public key in RSA cryptosystem, where $N = pq$ is the product of two distinct primes. For simplicity, we assume that $\gcd(p -$

$1, q-1) = 2$. A secret key d satisfies that $ed = 1 \bmod (p-1)(q-1)/2$. Hence, there exists an integer k such that $ed + k((N+1)/2 - (p+q)/2) = 1$. Writing $s = -(p+q)/2$ and $A = (N+1)/2$, we have $k(A+s) = 1 \pmod{e}$.

We set $f(x,y) = x(A+y)+1$. If one can solve a bivariate modular equation: $f(x,y) = x(A+y)+1 = 0 \pmod{e}$, one has k and s and knows the prime factors p and q of N. Suppose that the secret key satisfies $d \leq N^{\delta}$. Further assume that $e \approx N$. To summarize, the secret key will be recovered by finding the solution $(x,y) = (\bar{x}, \bar{y})$ of the equation: $x(A+y) = 1 \pmod{e}$, where $x \leq e^{\delta}$ and $|y| \leq e^{1/2}$. They referred this as the *small inverse problem*.

Boneh and Durfee gave an algorithm for solving this problem and obtained the condition on δ so that the algorithm works in polynomial time. Concretely, they showed that if $d \leq N^{0.284}$, N can be factored in polynomial time. Furthermore, they improved the bound to $d \leq N^{0.292}$.

2.2 LLL Algorithm and Howgrave-Graham's Lemma

For a vector $\boldsymbol{b}$, $||\boldsymbol{b}||$ denotes the Euclidean norm of $\boldsymbol{b}$. For a n-variate polynomial $h(x_1, \ldots, x_n) = \sum h_{j_1,\ldots,j_n} x_1^{j_1} \cdots x_n^{j_n}$, define the norm of a polynomial as $||h(x_1,\ldots,x_n)|| = \sqrt{\sum h_{j_1,\ldots,j_n}^2}$. That is, $||h(x_1,\ldots,x_n)||$ denotes the Euclidean norm of the vector which consists of coefficients of $h(x_1,\ldots,x_n)$.

Let $B = \{a_{ij}\}$ be a $w \times w'$ matrix of integers. The rows of B 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: $L = \{\sum_{i=1}^{w} n_i \boldsymbol{a_i} | n_i \in \mathbb{Z}\}$. The volume of lattice is defined by vol $(L) = \sqrt{\det(B\,{}^tB)}$, where ${}^tB$ is a transposed matrix of B. In particular, vol $(L) = |\det(B)|$ if B is full-rank.

LLL algorithm outputs short vectors in the lattice L.

Proposition 1 (LLL). *Let $B = \{a_{ij}\}$ be a non-singular $w \times w'$ matrix of integers. The rows of B generates a lattice L. Given B, the LLL algorithm outputs a reduced basis $\{\boldsymbol{b_1}, \ldots, \boldsymbol{b_w}\}$ with*

$$||\boldsymbol{b_i}|| \leq 2^{w(w-1)/(4(w+1-i))} (\text{vol } (L))^{1/(w+1-i)}$$

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

The following lemma is used when a modular equation is reduced into integer equation.

Lemma 1 (Howgrave-Graham [8]). *Let $\hat{h}(x_1,\ldots,x_n) \in \mathbb{Z}[x_1,\ldots,x_n]$ be a polynomial, which is a sum of at most w' monomials. Let m and ϕ be positive integers and $X_1,\ldots,X_n$ be some positive integers. Suppose that*

1. *$\hat{h}(\bar{x_1},\ldots,\bar{x_n}) = 0 \bmod \phi^m$, where $|\bar{x_1}| < X_1,\ldots |\bar{x_n}| < X_n$ and*
2. *$||\hat{h}(x_1X_1,\ldots,x_nX_n)|| < \phi^m/\sqrt{w'}$.*

Then $\hat{h}(\bar{x_1},\ldots,\bar{x_n}) = 0$ holds over integers.

3 How to Solve Generalized Small Inverse Problem

For a polynomial $h(x_1,\ldots,x_n)$, consider the following two problems: (I) Given $N(=pq)$, find a small solution of $h(x_1,\ldots,x_n) = 0(\bmod\ p)$. (II) Given M, find a small solution of $x_0h(x_1,\ldots,x_n) + C = 0(\bmod\ M)$. Problem (II) corresponds to a generalized small inverse problem. We will show a compiler which transforms a lattice basis for (I) to that for (II).

3.1 Lattice-Based Algorithm for (I)

The problem (I) can be solved by combining the LLL algorithm and Lemma 1 as follows. Let $X_1,\ldots,X_n$ be positive integers of Lemma 1. Define a polynomial as

$$h_{[j_1,\ldots,j_n,k]}(x_1,\ldots,x_n) := x_1^{j_1}\cdots x_n^{j_n}h(x_1,\ldots,x_n)^k$$

for non-negative integers $j_1,\ldots,j_n,k$. Let u be a non-negative integer. Using $h_{[j_1,\ldots,j_n,k]}$, we define a shift-polynomial

$$h^{(u)}_{[j_1,\ldots,j_n,k]}(x_1,\ldots,x_n) := h_{[j_1,\ldots,j_n,k]}(x_1,\ldots,x_n)N^{u-k}. \qquad (1)$$

Let a solution of $h = 0 \ (\bmod\ p)$ be $(x_1,\ldots,x_n) = (\bar{x_1},\ldots,\bar{x_n})$. It is easy to see that

$$h^{(u)}_{[j_1,\ldots,j_n,k]}(\bar{x_1},\ldots,\bar{x_n}) = 0 \ (\bmod\ p^u)$$

for any $(j_1,\ldots,j_n,k)$.

Fix a set $\mathcal{H}^{(u)}$ of $[j_1,\ldots,j_n,k]$ for each u. We construct a lattice $L^{(u)}_h$ spanned by a set of the coefficient vector of $h^{(u)}_{[j_1,\ldots,j_n,k]}(x_1X_1,\ldots,x_nX_n)$ for $[j_1,\ldots,j_n,k] \in \mathcal{H}^{(u)}$. Then, we apply the LLL algorithm to this lattice. The LLL algorithm yields small vectors of this lattice. Finally, we can obtain polynomial $\hat{h}$ satisfying the condition of Lemma 1 from this small vector. How to choose $\mathcal{H}^{(u)}$ for each u depends on $h(x_1,\ldots,x_n)$.

First, we define the set $M(h_{[j_1,\ldots,j_n,k]})$ of monomials

$$M(h_{[j_1,\ldots,j_n,k]}) \equiv \{x_1^{i_1}\cdots x_n^{i_n} | x_1^{i_1}\cdots x_n^{i_n} \text{ is a monomial of } h_{[j_1,\ldots,j_n,k]}(x_1,\ldots,x_n)\}.$$

Next, we define the set $M(\mathcal{H}^{(u)})$ of monomials

$$M(\mathcal{H}^{(u)}) \equiv \bigcup_{[j_1,\ldots,j_n,k]\in\mathcal{H}^{(u)}} M(h_{[j_1,\ldots,j_n,k]}).$$

We will introduce 4 restrictions for a lattice in solving $h = 0$ and consider only a set $\mathcal{H}^{(u)}$ of $[j_1,\ldots,j_n,k]$ for each u which holds 4 restrictions.

Restriction 1 For any positive integer u, there exist two sets $\mathcal{A} = \{[j_{1i},\ldots,j_{ni}]\}_{1\le i\le\#\mathcal{A}}$ and $\mathcal{B} = \{[j_{1i}^*,\ldots,j_{ni}^*]\}_{1\le i\le\#\mathcal{B}}$ such that $\mathcal{A}\subseteq\mathcal{B}$ and $\mathcal{H}^{(u)}$ is given by

$$\mathcal{H}^{(u)} = \bigcup_{k=0}^{u-1}\{[j_{1i},\ldots,j_{ni},k]\}_{1\le i\le\#\mathcal{A}} \cup \{[j_{1i}^*,\ldots,j_{ni}^*,u]\}_{1\le i\le\#\mathcal{B}}. \quad (2)$$

We call $(\mathcal{A},\mathcal{B})$ a *generator.*

Restriction 2 For any u, $L_h^{(u)}$ is full rank.

Restriction 3 A generator $\mathcal{B}$ is parametrized by some optimizing parameters $\boldsymbol{t} = (t_1,\ldots,t_k)$. If needed, we use notation: $\mathcal{B}(\boldsymbol{t})$.

Restriction 4 The volume of $L_h^{(u)}$ does not depend on coefficients of h. That is, it is given by

$$\text{vol } L_h^{(u)} = N^{\gamma_U} X_1^{\gamma_1} X_2^{\gamma_2}\cdots X_n^{\gamma_n}. \quad (3)$$

Let w be the dimension of the lattice. Here, $\gamma_U,\gamma_1,\ldots,\gamma_n$ and w are functions of u and $\boldsymbol{t}$. Moreover, each total degree of $\gamma_U,\gamma_1,\ldots,\gamma_n$ and uw is 2. If needed, we use vol $L_h^{(u;\boldsymbol{t})}$, $\gamma_U(u;\boldsymbol{t}), \gamma_i(u;\boldsymbol{t})$ for $1\le i\le n$.

Lattices derived in many previous method [6, 9, 12, 17] holds Restrictions 1–4 as described in Table 1.

Restriction 1 implies that if $[j_1,\ldots,j_n,k]\in\mathcal{H}^{(u)}$ and $k\ge 1$, then $[j_1,\ldots,j_n,k-1]\in\mathcal{H}^{(u-1)}$, which is crucial for our compiler. For convenience, we use the following notation: for a set $\mathcal{A}$ and $k\in\mathbb{Z}_{\ge0}$, a set $[\mathcal{A},k]$ is defined by $\{[j_1,\ldots,j_n,k] | [j_1,\ldots,j_n]\in\mathcal{A}\}$. If this notation is used, we can rewrite Eq. (2) as

$$\mathcal{H}^{(u)} = \bigcup_{k=0}^{u-1}[\mathcal{A},k]\cup[\mathcal{B},u].$$

Restriction 2 implies that $\#\mathcal{H}^{(u)} = \#M(\mathcal{H}^{(u)})$. The polynomial order of $\mathcal{H}^{(u)}$ and monomial order of $M(\mathcal{H}^{(u)})$ should be adequately defined so as to be linearly ordered. Let $B_h^{(u)}(\mathcal{A}, \mathcal{B})$ denote a $\#\mathcal{H}^{(u)} \times \#\mathcal{H}^{(u)}$ square matrix, where each row of $B_h^{(u)}(\mathcal{A}, \mathcal{B})$ is the coefficient vector of $h^{(u)}_{[j_1,\ldots,j_n,k]}(x_1X_1, \ldots, x_nX_n)$ when $\mathcal{A}$ and $\mathcal{B}$ is used as a generator. If $\mathcal{A}$ and $\mathcal{B}$ are clear from the context, we often omit $\mathcal{A}, \mathcal{B}$ and simply write $B_h^{(u)}$. Since $L_h^{(u)}$ is full-rank, vol $L_h^{(u)} = |\det B_h^{(u)}|$.

3.2 How to Solve (II)

We show how to solve the problem (II). First, we overview our algorithm and then focus on Step 1-2.

Input: $n+1$-variate equation $f(x_0, x_1, \ldots, x_n) = x_0h(x_1, \ldots, x_n) + C = 0 \pmod M$ with small roots

Output: All small roots $(\bar{x_0}, \ldots, \bar{x_n})$ of $f(x_0, x_1, \ldots, x_n) = 0 \pmod M$

Step1 Construct a lattice for f.

- **Step1-1** Construct a lattice $L_h^{(u)}$ for h or choose a generator $\mathcal{A}$ and $\mathcal{B}$ for h.
- **Step1-2** Construct a lattice L_f for f by employing the lattice for $L_h^{(u)}$ or $\mathcal{A}$ and $\mathcal{B}$.

Step2 Run LLL algorithm for input L_f to obtain $n+1$ polynomials $r_1, r_2, \ldots, r_{n+1} \in \mathbb{Z}[x_0, x_1, \ldots, x_n]$ over the integers, where they are non-zero integer combination of $f_{[i,j_1,\ldots,j_n,k]}(x_0X_0, x_1X_1, \ldots, x_nX_n)$ with small coefficients.

Step3 Compute a resultant for r_i to obtain a univariate integer equation. Then, solve the equation by using standard technique.

We point out some remarks. Our algorithm cannot always guarantee to output correct solutions. So, our algorithm is heuristic. We assume the following as same as [11].

Assumption 1 *The resultant computations for polynomials r_i yield non-zero polynomials.*

Experiments are needed for specific cases to justify the assumption.

We move on to the discussion of Step 1-2. Letting m be a positive integer, we define shift-polynomials for $f(x_0, x_1, \ldots, x_n)$ as

$$f_{[i,j_1,\ldots,j_n,k]}(x_0, x_1, \ldots, x_n) := x_0^i x_1^{j_1} \cdots x_n^{j_n} f(x_0, x_1, \ldots, x_n)^k M^{m-k}.$$

Let a solution of $f = 0 \pmod{M}$ be $(x_0, \ldots, x_n) = (\bar{x_0}, \ldots, \bar{x_n})$. It is easy to see that

$$f_{[i,j_1,\ldots,j_n,k]}(\bar{x_0}, \ldots, \bar{x_n}) = 0 \pmod{M^m}$$

for any $(i, j_1, \ldots, j_n, k)$.

Let $\mathcal{F}$ be a set of indexes $[i, j_1, \ldots, j_n, k]$. We construct the lattice L_f spanned by the coefficient vectors of $f_{[i,j_1,\ldots,j_n,k]}(x_0X_0, \ldots, x_nX_n)$ with $[i, j_1, \ldots, j_n, k] \in \mathcal{F}$. How does one choose a set of indexes $\mathcal{F}$? This is a difficult problem. The choice of $\mathcal{F}$ determines the performance of the algorithm. Indeed, the volume of the lattice derived by $\mathcal{F}$ should be small. Moreover, one must calculate or estimate the volume of lattice. If $\mathcal{F}$ is badly chosen, it might be difficult to calculate (or even though estimate) its volume. So, one must choose in a clever way the set $\mathcal{F}$. We overcome this problem by employing a lattice basis for solving $h = 0$. We propose the following compiler, which transforms a set of shift-polynomial for $h = 0$ into that for $f = 0$. In explanation, we use a notation: a set $[k_1, \mathcal{A}, k_2]$ is defined by $[k_1, \mathcal{A}, k_2] = \{[k_1, j_1, \ldots, j_n, k_2] | [j_1, \ldots, j_n] \in \mathcal{A}\}$.
Compiler Fix a positive integer m. By using generators $\mathcal{A}$ and $\mathcal{B}$ for $h = 0$, we construct a set $\mathcal{F}$ of shift-polynomials as follows. First, we set

$$\mathcal{F}^{(u)} \equiv \bigcup_{k=0}^{u-1} [u-k, \mathcal{A}, k] \cup [0, \mathcal{B}, u].$$

Then, we set

$$\mathcal{F} \equiv \bigcup_{u=0}^{m} \mathcal{F}^{(u)} = \bigcup_{u=0}^{m} \left\{ \bigcup_{k=0}^{u-1} [u-k, \mathcal{A}, k] \cup [0, \mathcal{B}, u] \right\}.$$

$\mathcal{F}$ is explicitly given by

$$\mathcal{F} = \bigcup_{u=0}^{m} \left\{ \bigcup_{k=0}^{u-1} \{[u-k, j_{1i}, \ldots, j_{ni}, k]\}_{1 \le i \le \#\mathcal{A}} \cup \{[0, j^*_{1i}, \ldots, j^*_{ni}, u]\}_{1 \le i \le \#\mathcal{B}} \right\}.$$

Obviously, $\#\mathcal{F}^{(u)} = \#\mathcal{H}^{(u)}$. If we define polynomial and monomial orders as follows, the polynomial set $\mathcal{F}$ and the monomial order are linearly ordered.

monomial order: We define $\prec$ as $x_0^u x_1^{j_1} \cdots x_n^{j_n} \prec x_0^{u'} x_1^{j'_1} \cdots x_n^{j'_n}$

$$\text{if } \begin{cases} u < u' \text{ or} \\ u = u' \text{ and } x_1^{j_1} \cdots x_n^{j_n} \prec x_1^{j'_1} \cdots x_n^{j'_n} \text{ in } M(\mathcal{H}^{(u)}). \end{cases}$$

polynomial order We define $\prec$ as $[i, j_1, \ldots, j_n, k] \prec [i', j'_1, \ldots, j'_n, k']$

$$\text{if } \begin{cases} i + k < i' + k' \text{ or} \\ i + k = i' + k' \text{ and } [j_1, \ldots, j_n, k] \prec [j'_1, \ldots, j'_n, k'] \text{ in } \mathcal{H}^{(i+k)} \end{cases}$$

Informally, letting $f' \in \mathcal{F}^{(u')}$ and $f'' \in \mathcal{F}^{(u'')}$, $f' \prec f''$ if $u' < u''$.

Theorem 1. *Suppose that $\mathcal{F}$ is set by our Compiler and $\mathcal{H}^{(u)}$ holding 4 restrictions. Let B be a matrix, where each row of B is the coefficient vectors of $f_{[u-k,j_1,\ldots,j_n,k]}(x_0X_0, \ldots, x_nX_n)$ according to the order of $\mathcal{F}$. Then, the matrix B is square and blocked lower triangular.*

For Theorem 1, B is written as

$$B = \begin{pmatrix} B_0 & & \mathbf{0} \\ & B_1 & \\ \vdots & & \ddots \\ * & & \cdots B_m \end{pmatrix},$$

where each B_u is a $\#\mathcal{H}^{(u)} \times \#\mathcal{H}^{(u)}$ matrix for $0 \le u \le m$. Note that B_u corresponds to $\#\mathcal{H}^{(u)}$ polynomials $\{f_{[i,j_1,\ldots,j_n,k]} | [i, j_1, \ldots, j_n, k] \in \mathcal{F}^{(u)}\}$ and $\#\mathcal{H}^{(u)}$ monomials which are divisible by x_0^u. The determinant of B is simply given by $\det B = \det B_0 \det B_1 \cdots \det B_m$.

The application to small secret exponent attack will be given in Section 5.1. Other examples are given in Section 5 and Appendix B.

3.3 Proof of Theorem 1

We define the set of monomials as

$M(f_{[u-k,j_1,\ldots,j_n,k]}) \equiv \{x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n} | x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n} \text{ is a monomial of } f_{[u-k,j_1,\ldots,j_n,k]}\}$

and

$$M(\mathcal{F}^{(u)}) \equiv \bigcup_{\boldsymbol{J} \in \mathcal{F}^{(u)}} M(f_{\boldsymbol{J}}).$$

We use the notation: $x_0^{i_0}\mathcal{M} \equiv \{x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n} | x_1^{i_1}\cdots x_n^{i_n} \in \mathcal{M}\}$ for $\mathcal{M} = \{x_1^{i_1}\cdots x_n^{i_n}\}$.

First, we show the following two lemmas.

Lemma 2. *If $[u-k, j_1, \ldots, j_n, k] \in \mathcal{F}$ for $k \ge 1$,it holds that*

$$M(f_{[u-k,j_1,\ldots,j_n,k-1]}) \subset M(f_{[u-k,j_1,\ldots,j_n,k]}).$$

Furthermore, it holds that for $k \ge 1$, $M(f_{[u-k,j_1,\ldots,j_n,k]}) \backslash M(f_{[u-k,j_1,\ldots,j_n,k-1]})$ $= x_0^u\{x_1^{i_1}\cdots x_n^{i_n} | x_1^{i_1}\cdots x_n^{i_n}$ is a monomial of $h_{[j_1,\ldots,j_n,k]}\}$.

Lemma 3. *It holds that*

$$M(\mathcal{F}^{(0)}) \subset M(\mathcal{F}^{(1)}) \subset \cdots \subset M(\mathcal{F}^{(m)}).$$

Furthermore, it holds that

$$M(\mathcal{F}^{(u)}) \setminus M(\mathcal{F}^{(u-1)}) = x_0^u M(\mathcal{H}^{(u)}).$$

Proof (of Lemma 2). For $k \geq 1$, if $[u-k, j_1, \ldots, j_n, k] \in \mathcal{F}^{(u)}$, then $[u-k, j_1, \ldots, j_n, k-1] \in \mathcal{F}^{(u-1)}$. The expansion of $f_{[u-k,j_1,\ldots,j_n,k]}(x_0, x_1, \ldots, x_n)$ is given by

$$\begin{aligned} f_{[u-k,j_1,\ldots,j_n,k]}(x_0, x_1, \ldots, x_n) &= x_0^{u-k} x_1^{j_1} \cdots x_n^{j_n} (x_0 h + C)^k M^{m-k} \\ &= x_0^u h_{[j_1,\ldots,j_n,k]} M^{m-k} + \sum_{i=1}^{k} \binom{k}{i} C^i M^{m-k} x_0^{u-i} h_{[j_1,\ldots,j_n,k-i]}. \end{aligned}$$

The expansion of $f_{[u-k,j_1,\ldots,j_n,k-1]}(x_0, x_1, \ldots, x_n)$ is given by

$$\begin{aligned} f_{[u-k,j_1,\ldots,j_n,k-1]}(x_0, x_1, \ldots, x_n) &= x_0^{u-k+1} x_1^{j_1} \cdots x_n^{j_n} (x_0 h + C)^{k-1} M^{m-k+1} \\ &= \sum_{i=1}^{k} \binom{k-1}{i-1} C^{i-1} M^{m-k+1} x_0^{u-i} h_{[j_1,\ldots,j_n,k-i]}. \end{aligned}$$

Then, we have the lemma. □

For Lemma 3, the number of monomials firstly appearing in $\mathcal{F}^{(u)}$ is $\#(M(\mathcal{F}^{(u)}) \setminus M(\mathcal{F}^{(u-1)})) = \#M(\mathcal{H}^{(u)})$. For the construction of our Compiler, the number of polynomials in $\mathcal{F}^{(u)}$ is $\#\mathcal{F}^{(u)} = \#\mathcal{H}^{(u)}$. Restriction 2 implies that $\#\mathcal{H}^{(u)} = \#M(\mathcal{H}^{(u)})$. Then, $\#(M(\mathcal{F}^{(u)}) \setminus M(\mathcal{F}^{(u-1)})) = \#\mathcal{F}^{(u)}$. This implies that B is blocked lower triangular. □

3.4 Small Example of our Compiler

We show a small example which shows how our Compiler works. Let $h(y)$ be a univariate monic polynomial with degree 1: $h(y) = A + y$. In this case, a target equation is $f(x, y) = xh(y) + C = x(A + y) + C = 0 \pmod{M}$.

Let $h_{[j,k]}^{(u)}(y) := y^j h(y)^k N^{u-k}$. Suppose that we use a generator $\mathcal{A} = \{[0]\}$ and $\mathcal{B} = \{[0], [1], [2]\}$. Then, $\mathcal{H}^{(0)} = \{[0, 0], [1, 0], [2, 0]\}$ and $\mathcal{H}^{(1)} = \{[0, 0], [0, 1], [1, 1], [2, 1]\}$. Corresponding matrixes $B_h^{(0)}$ and $B_h^{(1)}$ are given as follows.

$$B_h^{(0)} = \begin{array}{l|lll} & 1 & y & y^2 \\ \hline h_{[0,0]}^{(0)}(=1) & 1 & 0 & 0 \\ h_{[1,0]}^{(0)}(=Yy) & 0 & Y & 0 \\ h_{[2,0]}^{(0)}(=Y^2y^2) & 0 & 0 & Y^2 \end{array}, \quad B_h^{(1)} = \begin{array}{l|llll} & 1 & y & y^2 & y^3 \\ \hline h_{[0,0]}^{(1)}(=N) & N & 0 & 0 & 0 \\ h_{[0,1]}^{(1)}(=A+Yy) & A & Y & 0 & 0 \\ h_{[1,1]}^{(1)}(=AYy+Y^2y^2) & 0 & AY & Y^2 & 0 \\ h_{[2,1]}^{(1)}(=AY^2y^2+Y^3y^3) & 0 & 0 & AY^2 & Y^3 \end{array}$$

For example, $M(h_{[1,1]}) = \{y, y^2\}$ and $M(\mathcal{H}^{(1)}) = \{1, y, y^2, y^3\}$.

For a positive integer m, let $f_{[i,j,k]}(x, y) := x^i y^j f(x, y)^k M^{m-k}$. In the example, we fix $m = 1$. Applying our compiler, we obtain $\mathcal{F}$ of $f_{[i,j,k]}$ for solving $f(x, y) = xh(y) + C = 0 \pmod{M}$ as follows:

$$\mathcal{F} = \{[0, 0, 0], [0, 1, 0], [0, 2, 0], [1, 0, 0], [0, 0, 1], [0, 1, 1], [0, 2, 1]\}.$$

A matrix B generated by $\mathcal{F}$ is given as follows.

$$B = \begin{array}{l|ccc|cccc}
 & 1 & y & y^2 & x & xy & xy^2 & xy^3 \\
\hline
f_{[0,0,0]}(= M) & M & 0 & 0 & 0 & 0 & 0 & 0 \\
f_{[0,1,0]}(= YMy) & 0 & YM & 0 & 0 & 0 & 0 & 0 \\
f_{[0,2,0]}(= Y^2My^2) & 0 & 0 & Y^2M & 0 & 0 & 0 & 0 \\
\hline
f_{[1,0,0]}(= XMx) & 0 & 0 & 0 & XM & 0 & 0 & 0 \\
f_{[0,0,1]}(= C + AXx + XYxy) & C & 0 & 0 & AX & XY & 0 & 0 \\
f_{[0,1,1]}(= CYy + AXYxy + XY^2xy^2) & 0 & CY & 0 & 0 & AXY & XY^2 & 0 \\
f_{[0,2,1]}(= CY^2y^2 + AXY^2xy^2 + XY^3xy^3) & 0 & 0 & CY^2 & 0 & 0 & AXY^2 & XY^3
\end{array}$$

Columns and rows are ordered by polynomial and monomial orders in $\mathcal{F}$. The determinant of B is given by the product of diagonal elements. So, $\det B = M^4X^4Y^9$.

4 Deriving a Condition for Solving GSIP

In the previous section, we show how to choose a set $\mathcal{F}$. The next thing to do is evaluation of a volume of the lattice L_f or the determinant of the corresponding matrix B. Then, we will derive the condition for solving the problem by combining the value of determinant and Lemma 1.

First, we derive a determinant of matrix B (or a volume of L_f) obtained by our compiler.

Lemma 4. *Let $B_h^{(u;\boldsymbol{t})}$ be the corresponding matrix for h and $w(u; \boldsymbol{t})$ be the dimension of the lattice. Then, the determinant of B derived by our Compiler is given by*

$$\det B = M^{mW} \left(\frac{X_0}{M}\right)^{\sum_{u=0}^{m} uw(u;\boldsymbol{t})} \prod_{u=0}^{m} \det B_h^{(u;\boldsymbol{t})}(M), \tag{4}$$

where $W(= \sum_{u=0}^{m} w(u; \boldsymbol{t}))$ is the rank of B.

Next, we derive a condition that we can find all solutions of $f = 0 \pmod{M}$.

Lemma 5. *Suppose that the determinant of $B_h^{(u;\boldsymbol{t})}$ is given as the same as Lemma 4. Under Assumption 1, we can find all solutions of the equation $f = 0 \pmod M$ with $|x_0| < X_0, |x_1| < X_1, \ldots, |x_n| < X_n$ if*

$$\prod_{u=0}^{m} \det B_h^{(u;\boldsymbol{t})}(M) < \left(\frac{M}{X_0}\right)^{\sum uw(u;\boldsymbol{t})} = \prod_{u=0}^{m} \left(\frac{M}{X_0}\right)^{uw(u;\boldsymbol{t})}. \tag{5}$$

The time complexity is polynomial in $\log M$ and 2^n.

In case of Maximizing X_0 In many cryptanalysis, all the task is to maximize X_0 for fixed $X_1, X_2, \ldots, X_n$. Hereafter, we focus on this situation. We introduce an operator: $I : m^k \to \frac{1}{k+1} m^{k+1}$. Obviously, the operator I is homomorphic. Hence, we can write $\sum_{u=0}^{m} uw(u;\boldsymbol{t}) = I(mw(m;\boldsymbol{t}))$ and $\sum_{u=0}^{m} \gamma_i(u;\boldsymbol{t}) = I(\gamma_i(m;\boldsymbol{t}))$. We rewrite Eq. (5) by using the operator I as: $(X_0/M)^{I(mw(m;\boldsymbol{t}))} < M^{-I(\gamma_U(m;\boldsymbol{t}))} X_1^{-I(\gamma_1(m;\boldsymbol{t})} \cdots X_n^{-I(\gamma_n(m;\boldsymbol{t}))}$. Hence, we have

$$X_0 < M/(M^{I(\gamma_U(m;\boldsymbol{t}))} X_1^{I(\gamma_1(m;\boldsymbol{t})} \cdots X_n^{I(\gamma_n(m;\boldsymbol{t}))})^{1/I(mw(m;\boldsymbol{t}))}.$$

Let A_i be a fixed positive number such that $X_i = M^{A_i}$ for $1 \le i \le n$. We can simplify the above as $X_0 < M/M^{(I(\gamma_U(m;\boldsymbol{t})) + \sum_{i=1}^{n} A_i I(\gamma_i(m;\boldsymbol{t})))/I(mw(m;\boldsymbol{t}))}$. Setting

$$\begin{aligned} l(m;\boldsymbol{t}) &\equiv \frac{I(\gamma_U(m;\boldsymbol{t})) + \sum_{i=1}^{n} A_i I(\gamma_i(m;\boldsymbol{t}))}{I(mw(m;\boldsymbol{t}))} \\ &= \frac{I(\gamma_U(m;\boldsymbol{t}) + \sum_{i=1}^{n} A_i \gamma_i(m;\boldsymbol{t}))}{I(mw(m;\boldsymbol{t}))}, \end{aligned} \tag{6}$$

we have $X_0 < M^{1-l(m;\boldsymbol{t})}$. The next thing to do is to obtain $\boldsymbol{t}$ minimizing $l(m;\boldsymbol{t})$ for fixed m. The values $\boldsymbol{t}$ minimizing $l(m;\boldsymbol{t})$ is given by solving simultaneous equations:

$$\frac{\partial l(m;\boldsymbol{t})}{\partial t_1} = \frac{\partial l(m;\boldsymbol{t})}{\partial t_2} = \cdots = \frac{\partial l(m;\boldsymbol{t})}{\partial t_k} = 0.$$

Let $\boldsymbol{t'}$ be the solution of the above equations if it exists. If we ignore small terms[1], each $I(\gamma_U(m;\boldsymbol{t}))$, $I(\gamma_1(m;\boldsymbol{t})), \ldots I(\gamma_n(m;\boldsymbol{t})), I(mw(m;\boldsymbol{t})))$ consists of one term with the same total degree 3. Hence, each element

[1] If we don't ignore the small term, we can obtain the optimal value of m. But, we need tedious computation in general. For small secret exponent attack case, Boneh-Durfee gave the details analysis [2].

t'_i of $\boldsymbol{t'}$ is represented by $t'_i = \tau'_i m$ for positive integers τ_i's. Letting $\boldsymbol{\tau'} = (\tau'_1, \dots, \tau'_n)$, we have the condition for X_0:

$$X_0 \le \max_{\boldsymbol{t}} M^{1-l(m;\boldsymbol{t})} = M^{1-\min_{\boldsymbol{t}} l(m;\boldsymbol{t})} = M^{1-l(m;m\boldsymbol{\tau'})}, \tag{7}$$

which does not depend on m.

Next, we will analyze the most simple case, that is, $\mathcal{B}$ is parametrized by one parameter. In this case, we have an explicit formula of the upper bound of X_0.

Theorem 2. *Suppose that a lattice for $h = 0$ holds Restrictions 1–4 holds and $\mathcal{B}$ is parametrized by one parameter t. For given positive integers $A_1, \dots, A_n$, we set $a_2m^2 + a_1mt + a_0t^2 \equiv \gamma_U(m;t) + \sum_{i=1}^n A_i\gamma_i(m;t)$ and $w(m;t) = b_2m + b_1t$. Suppose that $a_1b_2 < a_2b_1$. Under Assumption 1, we can find all solutions of equation: $f = 0 \pmod{M}$ with $|x_0| < X_0, |x_1| < M^{A_1}, \dots, |x_n| < M^{A_n}$ if $X_0 < M^{1-\frac{4a_0}{b_1}c'-\frac{a_1}{b_1}}$, where $c' = (\sqrt{4a_0^2b_2^2 - 3a_0a_1b_1b_2 + 3a_0a_2b_1^2} - 2a_0b_2)/(3a_0b_1)$. In particular, if $b_1 = b_2$, we simply have the condition as*

$$X_0 < M^{1-\frac{4\sqrt{4a_0^2-3a_0a_1+3a_0a_2}-8a_0+3a_1}{3b_1}}. \tag{8}$$

Time complexity is in polynomial in $\log M$ and 2^n.

Remark 1. Eqs. (5) and (8) do not depend on the constant C.

5 Application of our Compiler to RSA-Related Cryptanalysis

We show several examples of GSIP and argue applications of our compiler to them. Table 1 summarizes some example of GSIP in the literature. "Constraint" shows what kind of constraint variables have in both of solving $f = 0$ and $h = 0$. "$\mathcal{A}$" and "$\mathcal{B}$" show what kind of generators we use in both of solving $f = 0$ and $h = 0$.

We give more explanation for each cases and give details of Case 1. More examples are given in Appendix B.

Case 1 Consider the small secret exponent attack on RSA by Boneh and Durfee [2]. In their attack, they handled $f(x, y) = x(y + A) + 1 = 0 \pmod{e}$. Hence, this problem corresponds to $h(y) = y + A$ and $C = 1$. By using our compiler, the lattice basis for $f(x) = 0$ is automatically obtained. Then, one can easily obtain the bound: $d \le N^{0.284}$. We'll discuss the details later.

Table 1. Examples of GSIP

	Case 1	Case 1'	Case 2	Case 3
	Boneh-Durfee [2]	May [15]	Durfee-Nguyen [7]	Itoh et al. [10]
$f = xh + C$	$x(A+y)+1$	$x(y-N)+N$	$x(A+y+z)+1$	$x(y-1)(z-1)+1$
Constraint	-	-	$yz = N$	$y^r z = N$
	Howgrave-Graham [9]		Coron-May [6]	Kunihiro-Kurosawa [12]
h	$y+A$	$y-N$	$A+y+z$	$(y-1)(z-1)$
$\mathcal{A}$	$\{[0]\}$		$\{[0,0],[1,0]\}$	$\{[0,0],[1,0]\} \cup \bigcup_{i=1}^{r-1}\{[i,1]\}$
$\mathcal{B}$	$\cup_{i=0}^{t}\{[i]\}$		$\cup_{i=0}^{t_1}\{[i,0]\} \cup \bigcup_{j=1}^{t_2}\{[0,j]\}$	$\cup_{i=0}^{t_1}\{[i,0]\} \cup \bigcup_{k=0}^{r-1}\bigcup_{j=1}^{t_2}\{[k,j]\}$

Case 1' Consider the small CRT exponent attack on unbalanced RSA by May [15]. In his attack, he handled $f(x,y) = x(y-N)+N = 0 \pmod{e}$. Hence, this problem corresponds to $h(y) = y - N$ and $C = N$. By using our compiler, the lattice basis for $f(x) = 0$ is automatically obtained. Furthermore, one can easily obtain the bound $d_p \le e^{1-2(\sqrt{\beta^2+3\beta}+\beta)/3}$, where $q < e^{\beta}$.

Case 2 Consider cryptanalysis on some variants of RSA with small secret exponent by Durfee-Nguyen [7]. In their attack, they handled the trivariate modular equation: $f(x,y,z) = x(A+y+z)+1 = 0 \pmod{e}$ with constraint $yz = N$. Hence, this problem corresponds to $h(y,z) = A + y + z$ and $C = 1$. By using our compiler, the lattice basis for Durfee-Nguyen's attack is automatically obtained.

Case 3 Consider the small secret exponent attacks on Takagi's variant of RSA [19] by Itoh et al. [10]. This attack can be obtained by our compiler and a lattice basis used in proving a deterministic polynomial equivalence between factoring and computing the secret exponent in that scheme [12]. Note that since Durfee-Nguyen technique is adequately involved in a lattice basis for h, we can easily obtain that for f. One can easily obtain the bound: $d < N^{(7-2\sqrt{7})/3(r+1)}$.

5.1 Case 1: Transforming Howgrave-Graham's Lattice Basis to Boneh-Durfee's Lattice Basis

Next, we move on to an actual cryptanalysis. We show that our compiler builds a bridge between a lattice basis in [9] and that in [2]. We simply write x, y, X, Y instead of x_0, x_1, X_0 and X_1.

Howgrave-Graham [9] provided an algorithm[2] for solving $h(y) = A + y = 0 \pmod{S}$ for integers A and S, which is an unknown divisor of an known integer U. Set shift-polynomials as $h^{(u)}_{[j,k]}(y) := h_{[j,k]}N^{u-k} = y^j h(y)^k N^{u-k}$. In his paper, he chose the set of the indexes of shift-polynomials as $\mathcal{H}^{(u)} = \bigcup_{k=0}^{u-1}\{[0,k]\} \cup \bigcup_{i=0}^{t}\{[i,u]\}$. We set a polynomial order by this. Note that a generator is given by $\mathcal{A} = \{[0]\}$ and $\mathcal{B} = \cup_{i=0}^{t}\{[i]\}$. Hence, $\mathcal{H}^{(u)}$ holds Restrictions 1–3.

Let $f(x,y) = x(A+y)+1$. We argue a lattice basis construction for f. Since $f(x,y) = xh(y)+1$, we can employ our Compiler to construct a lattice basis for f. For a positive integer m, we define shift-polynomials for f as $f_{[i,j,k]}(x,y) = x^i y^j f(x,y)^k M^{m-k}$. By our Compiler and Howgrave-Graham's lattice basis, we have a set $\mathcal{F}$ as

$$\mathcal{F} = \bigcup_{u=0}^{m}\left\{\bigcup_{k=0}^{u-1}\{[u-k,0,k]\} \cup \bigcup_{i=0}^{t}\{[0,i,u]\}\right\}$$

for fixed t. We have explicitly

$$\begin{aligned}\mathcal{F} = \{&[0,0,0],[0,1,0],\dots,[0,t,0],[1,0,0],[0,0,1],[0,1,1],\dots,[0,t,1],\\ &[m,0,0],[m-1,0,1],\dots,[0,0,m],[0,1,m],[0,2,m],\dots,[0,t,m]\}.\end{aligned}$$

As you can easily verify, Boneh-Durfee's set of shift-polynomials [2] and ours are completely the same as a set (but, a polynomial order is different). Then, they are the same as a lattice basis. So, we obtain the *same* lattice with Boneh-Durfee's by using our compiler and Howgrave-Graham's lattice basis [9].

Next, according to the discussion in Section 4, we will re-derive the bound of the secret key d. In [6], γ_U and γ_Y are given as $\gamma_U(u;t) = u(u+1)/2$ and $\gamma_Y(u;t) = (u+t)(u+t+1)/2$. And, the dimension is given by $w(u;t) = u+t+1$ and $A_Y = \log_e Y = 1/2$. In this case, we can obtain the same bound very easily. Since $\deg(\gamma_U(u;t)) = \deg(\gamma_Y(u;t)) = 2$ and $\deg(w(u;t)) = 1$, Restriction 4 holds. Then, we can use Theorem 2. By ignoring small terms, we have $a_0 = 1/4, a_1 = 1/2, a_2 = 3/4, b_1 = b_2 = 1$. By plugging these values into Eq. (8), one can easily obtain the bound

$$X < e^{1-\frac{4\sqrt{4\cdot1-3\cdot1\cdot2+3\cdot1\cdot3}-8\cdot1+3\cdot2}{3\cdot4}} = e^{(7-2\sqrt{7})/6} \approx N^{0.284},$$

which is exactly same as the Boneh-Durfee's weaker bound.

[2] By employing his algorithm, Coron and May gave the deterministic polynomial time algorithm for factoring the RSA modulus under the condition that the secret key d is given [6].

5.2 Case 1'

By using the same lattice basis as Case1, we re-derive the small CRT-exponent attack [15]. By just replacing $A_Y = \beta$, we can derive the condition: $d_p < e^{1-2(\sqrt{\beta^2+3\beta}+\beta)/3}$, where $q < e^{\beta}$.

6 Concluding Remarks and Open Problems

We note that our conversion is not enough. As shown in Sec. 5.1, our approach just achieves the Boneh-Durfee's *weaker bound*. We need more analysis to achieve the stronger bound: $d \leq N^{0.292}$. Actually, Boneh and Durfee [2] deleted some *bad* lattice bases and introduced the concept *Geometrically Progressive Matrix* to evaluate the upper bound of the determinant of the lattice. By these efforts, they achieved the stronger bound $d \leq N^{0.292}$. We need to develop a general theory including such an improvement.

Acknowledgement

The author thanks Kaoru Kurosawa for helpful discussions.

References

1. J. Blömer and A. May, "A Tool Kit for Finding Small Roots of Bivariate Polynomials over the Integers," in Proc. of Eurocrypt2005, LNCS 3494, pp. 251–267, 2005.
2. D.Boneh and G.Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$," IEEE Transactions on Information Theory 46(4): 1339 (2000). (Firstly appeared in Eurocrypt'99).
3. D. Coppersmith, "Finding a Small Root of a Univariate Modular Equation,"in Proc. of Eurocrypt'96, LNCS 1070, pp. 155–165, 1996.
4. D. Coppersmith, "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known," in Proc. of Eurocrypt'96, LNCS 1070, pp. 178–189, 1996.
5. D. Coppersmith, "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities," J. Cryptology 10(4): 233-260, 1997.
6. J.S. Coron and A.May, "Deterministic Polynomial Time Equivalence of Computing the RSA Secret Key and Factoring," Journal of Cryptology, Vol. 20, No. 1, pp. 39–50, 2007. (IACR ePrint Archive: Report 2004/208, 2004.)
7. G. Durfee and P. Nguyen, "Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt'99," in Proc. of Asiacrypt2000, LNCS 1976, pp. 14–29, 2000.
8. N. Howgrave-Graham, "Finding Small Roots of Univariate Modular Equations Revisited," IMA Int. Conf., pp.131–142 (1997)
9. N. Howgrave-Graham, "Approximate Integer Common Divisors," in Proc. of Cryptography and Lattice Conference (CaLC2001), LNCS 2146, pp. 51–66, 2001.

10. K. Itoh, N. Kunihiro and K. Kurosawa, “Small Secret Key Attack on a Variant of RSA (due to Takagi),” In Proc. of CT-RSA2008, LNCS4964, pp. 387–406, 2008.
11. E. Jochemsz and A. May, “A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants,” In Proc. of Asiacrypt2006, LNCS4284, pp. 267–282, 2006.
12. N. Kunihiro and K. Kurosawa, “Deterministic Polynomial Time Equivalence between Factoring and Key-Recovery Attack on Takagi’s RSA,” In Proc. of PKC2007, LNCS4450, pp. 412-425, 2007.
13. N. Kunihiro, “Solving Generalized Small Inverse Problems,” to appear in Proc. of ACISP2010.
14. A.K. Lenstra, H.W. Lenstra, L. Lovász, “Factoring polynomials with rational coefficients,” Mathematische Annalen 261, pp.515–534, 1982.
15. A. May, “Cryptanalysis of Unbalanced RSA with Small CRT-Exponent,” in Proc. of Crypto2002, LNCS 2442, pp. 242–256, 2002.
16. A. May, “Computing the RSA Secret Key Is Deterministic Polynomial Time Equivalent to Factoring,” in Proc. of Crypto2004, LNCS 3152, pp. 213–219, 2004.
17. A. May, Chapter3.2 “The univariate case,” in “New RSA Vulnerabilities Using Lattice Reduction Methods,” Ph.D thesis, University of Paderborn, 2003.
18. 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.
19. T. Takagi, “Fast RSA-Type Cryptosystem Modulo $p^k q$, ” in Proc. of Crypto’98, LNCS 1462, pp.318–326, 1998.
20. M. Wiener, “Cryptanalysis of Short RSA Secret Exponents,” IEEE Transactions on Information Theory, Vol. 36, pp. 553–558, 1990.

A Proofs

A.1 Proof of Lemma 4

The determinant of the submatrix B_u is given by

$$\det B_u = M^{(m-u)w} X_0^{uw} \det B_h^{(u)}(M) = M^{mw} \left(\frac{X_0}{M}\right)^{uw} \det B_h^{(u)}(M).$$

Since the determinant $\det B$ for f is given by $\det B = \prod_{u=0}^{m} \det B_u$, we have the lemma. $\square$

A.2 Proof of Lemma 5

For Lemma 1, if the norm of $\boldsymbol{b_{n+1}}$ is less than $M^m/\sqrt{w}$, we can reduce the modular equations into integer equations. Combining Proposition 1, this condition can be transformed into

$$\det B < M^{mW}/\gamma, \tag{9}$$

where γ is a constant. Since this term is negligible compared to M^{mW}, we can ignore this term. By substituting Eq. (4) into Eq. (9), we have

$$(X_0/M)^{\sum uw(u)} < \prod_{u=0}^{m} (\det B_h^{(u)}(M))^{-1}. \tag{10}$$

It is important that M^{mW} in both hand sides are canceled. By transforming this inequality, we have the above condition. $\square$

A.3 Proof of Theorem 2

The function $l(m;t)$ is given by

$$l(m;t) = \frac{a_0 m t^2 + a_1 m^2 t/2 + a_2 m^3/3}{b_1 m^2 t/2 + b_2 m^3/3}. \tag{11}$$

By replacing $x = t/m$, we have

$$l(x) \equiv l(m;mx) = \frac{6a_0 x^2 + 3a_1 x + 2a_2}{3b_1 x + 2b_2}.$$

The value x minimizing $l(x)$ satisfies $3a_0b_1x^2 + 4a_0b_2x + (a_1b_2 - a_2b_1) = 0$. If $a_1b_2 - a_2b_1 < 0$, this equation has a positive solution. By solving the above equation, we have

$$x = \frac{\sqrt{4a_0^2b_2^2 - 3a_0a_1b_1b_2 + 3a_0a_2b_1^2} - 2a_0b_2}{3a_0b_1}.$$

Letting this value c' and plugging c' into Eq. (7), we have the following condition for X_0:

$$\log_M X_0 < 1 - \frac{4a_0}{b_1}c' - \frac{a_1}{b_1}.$$

In particular, if $b_1 = b_2$, we have simply

$$c' = \frac{\sqrt{4a_0^2 - 3a_0a_1 + 3a_0a_2}}{3a_0} - \frac{2}{3}.$$

B More Examples

B.1 Application to Coron-May's Lattice Basis to Durfee-Nguyen's Lattice basis

In this subsection, we simply write x, y, z, X, Y, Z instead of x_0, x_1, x_2, X_0, X_1 and X_2.

Coron and May gave the deterministic polynomial time algorithm for factoring the *unbalanced* RSA modulus under the condition that the secret key d is given [6]. In the proof, they provided an algorithm for solving $h(y,z) = A + y + z = 0 \pmod{S}$ for integers A and S, which is an unknown divisor of an known integer U. Note that y and z have relation: $yz = N$.

Set shift-polynomials as $h^{(u)}_{[j,k,l]}(y,z) := h_{[j,k,l]}N^{u-l} = y^j z^k h(y,z)^l N^{u-l}$. In their paper, they chose the set of the indexes of shift-polynomials as

$$\mathcal{H}^{(u)} = \bigcup_{k=0}^{u-1}\{[0,0,k] \cup [1,0,k]\} \cup \bigcup_{i_1=0}^{t_1}\{[i_1,0,u]\} \cup \bigcup_{i_2=0}^{t_2}\{[0,i_2,u]\}.$$

Note that a generator is given by

$$\mathcal{A} = \{[0,0],[1,0]\} \text{ and } \mathcal{B} = \cup_{i_1=0}^{t_1}\{[i_1,0]\} \cup \bigcup_{i_2=1}^{t_2}\{[0,i_2]\}.$$

Hence, $\mathcal{H}^{(u)}$ holds Restrictions 1–3.

Let $f(x,y,z) = x(A+y+z)+1$ with constraint $yz = N$. We argue a lattice basis construction for f. Since $f(x,y,z) = xh(y,z)+1$, we can employ our Compiler to construct a lattice basis for f. For a positive integer m, we define shift-polynomials for f as $f_{[i,j,k,l]}(x,y) := x^i y^j z^l f(x,y,z)^l M^{m-l}$. By our Compiler and Coron-May's lattice basis, we have a set $\mathcal{F}$ as

$$\mathcal{F} = \bigcup_{u=0}^{m}\left\{\bigcup_{k=0}^{u-1}\{[u-k,0,0,k] \cup [u-k,1,0,k]\} \cup \bigcup_{i_1=0}^{t_1}\{[0,i_1,0,u]\} \cup \bigcup_{i_2=0}^{t_2}\{[0,0,i_2,u]\}\right\}$$

for fixed t_1 and t_2. As you can easily verify, Durfee-Nguyen's set of shift-polynomials [7] and ours are completely the same as a set. Then, they are the same as a lattice basis. So, we obtain the *same* lattice with Boneh-Durfee's by using our compiler and Coron-May's lattice basis [6].

B.2 Application to Kunihiro-Kurosawa's Lattice Basis to Itoh et al.'s Lattice basis

In this subsection, we simply write x, y, z, X, Y, Z instead of x_0, x_1, x_2, X_0, X_1 and X_2.

Kunihiro and Kurosawa gave the deterministic polynomial time algorithm for factoring the RSA modulus for the Takagi's variant of RSA under the condition that the secret key d is given [12]. In the proof, they provided an algorithm for solving $h(y,z) = (y-1)(z-1) = 0 \pmod{S}$

for an integer$S$, which is an unknown divisor of an known integer U. Note that y and z have relation: $y^r z = N$.

Set shift-polynomials as $h^{(u)}_{[j,k,l]}(y,z) := h_{[j,k,l]}N^{u-l} = y^j z^k h(y,z)^l N^{u-l}$. In their paper, they chose the set of the indexes of shift-polynomials as

$$\mathcal{H}^{(u)} = \bigcup_{k=0}^{u-1}\left\{[0,0,k] \cup [1,0,k] \cup \bigcup_{i=1}^{r-1}\{[i,1,k]\}\right\}$$
$$\cup \bigcup_{i_1=0}^{t_1}\{[i_1,0,u]\} \cup \bigcup_{i_3=0}^{r-1}\bigcup_{i_2=1}^{t_2}\{[i_3,i_2,u]\}.$$

Note that a generator is given by

$$\mathcal{A} = \{[0,0],[1,0]\} \cup \bigcup_{i=1}^{r-1}\{[i,1]\} \text{ and } \mathcal{B} = \cup_{i_1=0}^{t_1}\{[i_1,0]\} \cup \bigcup_{i_3=0}^{r-1}\bigcup_{i_2=1}^{t_2}\{[i_3,i_2]\}.$$

Hence, $\mathcal{H}^{(u)}$ holds Restrictions 1–3.

Let $f(x,y,z) = x(y-1)(z-1)+1$ with constraint $y^r z = N$. We argue a lattice basis construction for f. Since $f(x,y,z) = xh(y,z)+1$, we can employ our Compiler to construct a lattice basis for f. For a positive integer m, we define shift-polynomials for f as $f_{[i,j,k,l]}(x,y) := x^i y^j z^l f(x,y,z)^l M^{m-l}$. By our Compiler and Kunihiro-Kurosawa's lattice basis, we have a set $\mathcal{F}$ as

$$\mathcal{F}^{(u)} = \bigcup_{k=0}^{u-1}\left\{[u-k,0,0,k] \cup [u-k,1,0,k] \cup \bigcup_{i=1}^{r-1}\{[u-k,i,1,k]\}\right\}$$
$$\cup \bigcup_{i_1=0}^{t_1}\{[0,i_1,0,u]\} \cup \bigcup_{i_3=0}^{r-1}\bigcup_{i_2=1}^{t_2}\{[0,i_3,i_2,u]\}.$$

and

$$\mathcal{F} = \bigcup_{u=0}^{m}\mathcal{F}^{(u)}$$

for fixed t_1 and t_2. As you can easily verify, Itoh et. al's set of shift-polynomials [10] for weaker bound: $d \le N^{(0.568/(r+1))}$ and ours are completely the same as a set. Then, they are the same as a lattice basis. So, we obtain the *same* lattice with Itoh et al.'s by using our compiler and Kunihiro-Kurosawa's lattice basis [12].

B.3 Analysis for Non-linear $h(y)$

May [17] extended the Howgrave-Graham's result [9] to analyze the case for univariate polynomial with higher-degree. By using May's lattice basis, we analyze the case for the equation $f(x, y) = xh(y) + C = 0 \pmod{M}$, where h is a univariate monic polynomial with degree $\kappa \geq 1$ and C is a non-zero integer. May [17] gave the set of indexes in the explicit form. A generator for h is given by $\mathcal{A} = \cup_{i=0}^{\kappa-1}\{[i]\}$ and $\mathcal{B} = \cup_{i=1}^{t}\{[i]\}$. The determinant of $B_h^{(u)}$ by $\det B_h^{(u)} = N^{\kappa u(u+1)/2} Y^{w(w-1)/2}$, where $w = \kappa u + t$. By the similar analysis in Section 4, we have

$$X < M^{1 - \frac{2\sqrt{(\beta\kappa)^2 + 3\beta\kappa} - \beta\kappa}{3}},$$

where $\beta = \log_M Y$. Letting $M = e$, $\beta = 1/2$ and $\kappa = 1$, one has again the Boneh-Durfee's weaker bound: $\delta < (7 - 2\sqrt{7})/6$.