Practical Cryptanalysis of a Pulblic-key Encryption Scheme Based on Non-linear Indeterminate Equations at SAC 2017

Keita Xagawa

3-9-11, Midori-cho Musashino-shi, Tokyo 180-8585 Japan
xagawa.keita@lab.ntt.co.jp

Abstract. We investigate the security of a public-key encryption scheme, the Indeterminate Equation Cryptosystem (IEC), introduced by Akiyama, Goto, Okumura, Takagi, Nuida, and Hanaoka at SAC 2017 as post-quantum cryptography. They gave two parameter sets PS1 $(n, p, \deg X, q) = (80, 3, 1, 921601)$ and PS2 $(n, p, \deg X, q) = (80, 3, 2, 58982400019)$.
The paper gives practical key-recovery and message-recovery attacks against those parameter sets of IEC through lattice basis-reduction algorithms. We exploit the fact that $n = 80$ is composite and adopt the idea of Gentry's attack against NTRU-Composite (EUROCRYPT2001) to this setting. The summary of our attacks follows:

- On PS1, we recover 84 private keys from 100 public keys in 30–40 seconds per key.
- On PS1, we recover partial information of all message from 100 ciphertexts in a second per ciphertext.
- On PS2, we recover partial information of all message from 100 ciphertexts in 30 seconds per ciphertext.

Moreover, we also give message-recovery and distinguishing attacks against the parameter sets with prime n, say, $n = 83$. We exploit another subring to reduce the dimension of lattices in our lattice-based attacks and our attack succeeds in the case of $\deg X = 2$.

- For PS2' $(n, p, \deg X, q) = (83, 3, 2, 68339982247)$, we recover 7 messages from 10 random ciphertexts within 61,000 seconds $\approx$ 17 hours per ciphertext.
- Even for larger n, we can find short vector from lattices to break the underlying assumption of IEC. In our experiment, we can found such vector within 330,000 seconds $\approx$ 4 days for $n = 113$.

keywords: Public-Key Encryption, Indeterminate Equations Cryptosystem, Post-quantum cryptography.

1 Introduction

Algebraic-Surface Cryptosystem (ASC) is a public-key cryptosystem based on the section-finding problem. Let $R_{n,q} := \mathbb{Z}_q[t]/(t^n - 1)$ and consider $R_{n,q}[x, y]$. The section-finding problem over $R_{n,q}[x, y]$ is, given an algebraic surface $X(x, y) = 0$, finding the section $u = (u_x, u_y) \in R_{n,q}^2$ such that $X(u_x, u_y) = 0$ [AG06,AGM09]. Recently, the new version of ASC, the IEC encryption scheme, was proposed by Akiyama, Goto, Okumura, Takagi, Nuida, and Hanaoka at SAC 2017 [AGO+18], where IEC stands for Indeterminate Equation Cryptosystem. The authors investigate the security of IECs by considering the lattice-based attacks and define two sets of parameter values, PS1 $(n, p, \deg X, q) = (80, 3, 1, 921601)$ and PS2 $(n, p, \deg X, q) = (80, 3, 2, 58982400019)$.

1.1 Our Contribution

We give practical-time lattice-based attacks against the IECs.

Our first attack is combining the original lattice-based attack with Gentry's attack [Gen01] against NTRU Composite [Sil01]. Let d be a non-trivial divisor of n, say, 40. We can consider the subring $R_{d,q}[x, y]$ instead of $R_{n,q}[x, y]$. This modification allows us to employ a smaller lattice than that in the original lattice-based attacks. Our attack succeeds as follows:

- On PS1, we mount a key-recovery attack. Our attack finds 84 secret keys from 100 random keys. The attack took approximately 30 seconds per key.
- On PS1, we mound a partial-message-recovery attack. Our attack finds partial messages of all 100 pairs of random public key and ciphertext. The attack took approximately 0.5 seconds per try.
- On PS2, we mound a partial-message-recovery attack. Our attack finds partial messages of all 100 pairs of random public key and ciphertext. The attack took approximately 30 seconds per try.

We exploit another class of subring $R_{n,q}[x]$ of $R_{n,q}[x, y]$ to reduce the dimension of lattices in our lattice-based attacks. Our attack succeeds in the case of $\deg X = 2$ as follows:

– For $(n, p, \deg X) = (83, 3, 2)$, we recover 7 messages out of 10 random ciphertexts in 61,000 seconds $\approx$ 17 hours per ciphertext.
– Even for larger n, we can find short vector which enables us to break the underlying assumption of IEC. We can find such vector for $n = 113$ within 330,000 seconds $\approx$ 4 days.

Responsible Disclosure Process: We already notified the authors of our attacks before making this paper public. We informed them by email on September 28th with key-recovery attack on PS1, October 2nd with partial-message-recovery attack on PS1 and PS2, October 17th with message-recovery attack on $(n, p, \deg X) = (83, 3, 2)$, and November 2nd with distinguishing attack on variant of PS2 with $n \geq 83$. The authors reported that they have changed parameter values and they run their experiments further. We publish this paper after Akiyama et al. published their revised paper and their NIST PQC submission [AGO+17b,AGO+17a].

1.2 Organization

We define notations and review lattices in section 2. We review the IEC scheme in section 3 and the original lattice-based attacks in section 4. We recall Gentry's attack in section 5. We combine them in section 6 and give new attacks in section 7. The experimental results are reported in section 8.

2 Preliminaries

Notations: The security parameter is denoted by κ.

For a positive integer q, we define $\mathbb{Z}_q := \mathbb{Z}/(q\mathbb{Z})$ and $\mathbb{Z}_q^+ := \{0, 1, \ldots, q-1\}$. For a positive integer n, we define $R_n := \mathbb{Z}[t]/(t^n - 1)$. For two positive integers n and q, we define $R_{n,q} := \mathbb{Z}_q[t]/(t^n - 1)$. We also define a subset $R_{n,q,p}$ of $R_{n,q}$ as a set of all $\mathbb{Z}_p$-coefficient polynomials in $R_{n,q}$, that is,

$$R_{n,q,p} := \left\{ f = \sum_{i=0}^{n-1} f_i t^i \in R_{n,q} \;\middle|\; f_i \in \{0, 1, \ldots, p-1\} \subset \mathbb{Z}_q \right\}.$$

Let R be a ring and consider $R[x, y]$. For R and a set of indices $\Gamma \subseteq \mathbb{Z}^2_{\geq 0}$, we define

$$\mathfrak{F}(\Gamma, R) := \left\{ f \in R[x, y] \mid f = \textstyle\sum_{(i,j)\in\Gamma} a_{ij} x^i y^j \right\},$$

a set of all polynomials in $R[x, y]$ which only consists of $x^i y^j$ terms for $(i, j) \in \Gamma$. We will refer Γ as the term set. (Those notations are borrowed from [AGO+18].) We define *the total degree* of $f(x, y) \in R[x, y]$ as the maximum of the sums of the exponents of the variables in the term $a_{ij} x^i y^j$.

Polynomials: We review the notations which bridge polynomials in R_n and n-dimensional vectors (and matrices). For integers n and q, let us define two functions:

$$\mathrm{vec}_n \colon R_{n,q} \to \mathbb{Z}^n \colon f = f_0 + f_1 t + \cdots + f_{n-1} t^{n-1} \mapsto (f_0, f_1, \ldots, f_{n-1})$$

$$\mathrm{Rot}_n \colon R_{n,q} \to \mathbb{Z}^{n\times n} \colon f \mapsto= \{f_{j-i \bmod n}\}_{i,j=0,\ldots,n-1} = \begin{pmatrix} \mathrm{vec}_n(f) \\ \mathrm{vec}_n(tf) \\ \mathrm{vec}_n(t^2 f) \\ \vdots \\ \mathrm{vec}_n(t^{n-1} f) \end{pmatrix}.$$

We have

$$\mathrm{vec}_n(f) \cdot \mathrm{Rot}_n(g) = \mathrm{vec}_n(f \cdot g) \text{ and } \mathrm{Rot}_n(f) \cdot \mathrm{Rot}_n(g) = \mathrm{Rot}_n(f \cdot g)$$

Lattices: Given n-linearly independent vectors $B = \{b_0, \ldots, b_{n-1}\} \subset \mathbb{R}^m$, the lattice generated by them is the set of vectors

$$\mathcal{L}(B) = \mathbb{Z}^n \cdot B = \{\sum_{i=0}^{n-1} x_i b_i \mid x_i \in \mathbb{Z}\}.$$

The vectors B are known as a basis of the lattice. If $n = m$, we say the lattice is the full-rank. In what follows, we only consider full-rank lattices.

The determinant or volume $\mathrm{vol}(\Lambda)$ of a full-rank lattice Λ is the absolute value of the determinant of any given basis B of Λ, that is, $\mathrm{vol}(\Lambda) = |\det(B)|$. The dual of a lattice Λ, denoted by Λ^*, is the lattice consisting of the set of all vectors $z \in \mathbb{R}^m$ orthogonal to any vectors $v \in \Lambda$, that is, $\Lambda^* = \{z \in \mathbb{R}^m \mid \langle z, y\rangle = 0 \text{ for all } y \in \Lambda\}$.

We also define q-ary lattices. For $A \in \mathbb{Z}_q^{n\times m}$,

$$\Lambda_q(A) := \{z \in \mathbb{Z}^m \mid z = sA \pmod q \text{ for some } s \in \mathbb{Z}^n\}$$
$$\Lambda_q^{\perp}(A) := \{e \in \mathbb{Z}^m \mid eA^{\top} \equiv 0 \pmod q\}.$$

We have

$$\Lambda_q^{\perp}(A) = q \cdot \Lambda_q(A)^* \text{ and } \Lambda_q(A) = q \cdot \Lambda_q^{\perp}(A)^*.$$

See e.g., [GPV08, Section 5].

The basis of Λ_q is easily obtained. For example, we obtain the basis by considering a matrix $\begin{pmatrix} A \\ qI_m \end{pmatrix}$ and taking the row echelon form of the matrix.

SVP and CVP: Finally we define shortest-vector problem and closest-vector problem. The shortest-vector problem (SVP) is, given a lattice Λ, finding a non-zero vector $v \in \Lambda \setminus \{0\}$ such that $\|v\| \le \|x\|$ for any non-zero lattice vector $x \in \Lambda \setminus \{0\}$. The closet-vector problem (CVP) is, given a lattice Λ and a target vector t, finding a lattice vector $w \in \Lambda$ such that $\|w - t\| \le \|x - t\|$ for any lattice vector $x \in \Lambda$.

The Gaussian heuristic says that the m-dimensional full-rank lattice contains a short vector of length approximately

$$\gamma = \sqrt{\frac{m}{2\pi e}} \det(L)^{1/m}.$$

If our target vector v is sufficiently smaller than γ, then we expect the LLL or BKZ algorithm find the short vector v.

3 IEC Scheme

Parameters: In the IEC scheme, we will employ $X \in R[x, y]$ as a public key, $r, e \in R[x, y]$ as a random polynomials in ciphertexts. The IEC involves several parameters, (p, q, n) and $(\Gamma_X, \Gamma_r, \Gamma_{Xr})$:

1. p, q: primes and $p \ll q$
2. n: the degree of $R_{n,q} = \mathbb{Z}_q[t]/(t^n - 1)$
3. Γ_X: The term set of $X(x, y)$
4. w_X: The total degree of X
5. Γ_r: The term set of the random polynomial $r(x, y)$
6. w_r : The total degree of r
7. Γ_{Xr}: The term set of the random polynomial $e(x, y)$

Akiyama et al. defined

$$\Gamma_{Xr} := \{(i, j) + (k, l) \mid (i, j) \in \Gamma_X, (k, l) \in \Gamma_r\}$$

in order to avoid the linear algebraic attacks against the previous cryptosystems [AGO+18, Section 2.2]. They also require large q as

$$q > \#\Gamma_{Xr} \cdot p(p - 1) \cdot (n(p - 1))^{w_X + w_r} \tag{1}$$

to make the scheme perfectly correct. They implicitly defined

$$\Gamma_X = \{(i, j) \in \mathbb{Z}^2_{\ge 0} \mid i + j \le w_X\} \text{ and } \Gamma_r = \{(i, j) \in \mathbb{Z}^2_{\ge 0} \mid i + j \le w_r\}.$$

Although Γ_X and Γ_r can be different, they always take $\Gamma_X = \Gamma_r$. Hence, they just parameterize $\deg X$ instead of w_X and w_r. They give two sets of parameter values in Table 1.

Table 1: Proposed sets of parameter values [AGO+18, Table 3]. PS2' is obtained by setting $n = 83$ in PS2

	n	p	q	$\deg X$	$\deg r$	$\#\Gamma_{Xr}$	$\lvert sk\rvert$ (bits)	$\lvert pk\rvert$ (bits)	$\lvert ct\rvert$ (bits)
PS1	80	3	921601	1	1	6	256	4755	9510
PS2	80	3	58982400019	2	2	15	256	17174	42935
PS2'	83	3	68339982247	2	2	15	264	17928	44820

Key Generation: The secret key is a *small* solution of the indeterminate equation $X(x, y) = 0$. We denote the solution by

$$u\colon (x, y) = (u_x(t), u_y(t)) \in R_{n,q,p}^2.$$

The public key is the indeterminate equation $X(x, y) = 0$ that has a small solution u. We denote it by

$$X(x, y) = \sum_{(i,j)\in\Gamma_X} a_{ij} x^i y^j, \text{ where } a_{ij} \in R_{n,q}.$$

Akiyama et al. recommend to choose a_{ij} except a_{00} uniformly at random and set $a_{00} := -\sum_{(i,j)\in\Gamma_X\setminus\{(0,0)\}} a_{ij} u_x^i u_y^j$.

Encryption: A plaintext is treated as $m(t) \in R_{n,q,p}$. The ciphertext is

$$c(x, y) := m(t) + X(x, y) \cdot r(x, y) + p \cdot e(x, y) \in \mathfrak{F}(\Gamma_{Xr}, R_{n,q}),$$

where we choose $r(x, y) \leftarrow \mathfrak{F}(\Gamma_r, R_{n,q})$ and $e(x, y) \leftarrow \mathfrak{F}(\Gamma_{Xr}, R_{n,q,p})$.

Decryption: Given a ciphertext $c(x, y) \in \mathfrak{F}(\Gamma_{Xr}, R_{n,q})$,

1. Compute $c(u_x, u_y) \in R_{n,q}$
2. regard $c(u_x, u_y)$ as a polynomial in R_n $(= \mathbb{Z}[t]/(t^n - 1))$, compute $m'(t) := c(u_x, u_y) \mod p$, and output $m'(t)$

Notice that $c(u_x, u_y) = m(t) + p \cdot e(u_x, u_y) \in R_{n,q}$ because $X(u_x, u_y) = 0 \in R_{n,q}$. By the condition on q and p, if c is a valid ciphertext, then $c(u_x, u_y) \bmod q = m(t) + p \cdot e(u_x, u_y) \in R_n$. Thus, we have $m(t) = (c(u_x, u_y) \bmod q) \bmod p$.

See our implementation in Listing 1.1.

3.1 Security Assumption

Let $\mathfrak{X}(\Gamma_X, R_{n,q}, p)$ be the set of $X(x, y)$ which has a small solution u, that is,

$$\mathfrak{X}(\Gamma_X, R_{n,q}, p) := \{X \in \mathfrak{F}(\Gamma_X, R_{n,q}) \mid \exists u_x, u_y \in R_{n,q,p} : X(u_x, u_y) = 0\}.$$

Akiyama et al. defined the following decision problem:

Definition 3.1 (IE-LWE Problem). *For parameters $n, p, q, \Gamma_X, \Gamma_r$, and Γ_{Xr}, we define two sets*

$$U := \mathfrak{X}(\Gamma_X, R_{n,q}, p) \times \mathfrak{F}(\Gamma_{Xr}, R_{n,q})$$
$$T := \{(X, Xr + e) \mid X \in \mathfrak{X}(\Gamma_X, R_{n,q}, p), r \in \mathfrak{F}(\Gamma_r, R_{n,q}), e \in \mathfrak{F}(\Gamma_{Xr}, R_{n,q,p})\}.$$

The IE-LWE problem is distinguishing the multivariate polynomials chosen from a 'noisy' set T of polynomials from a 'uniform' set U.

The IE-LWE assumption states that it is infeasible to solve the IE-LWE problem, where X is chosen by the key-generation algorithm Gen.

Definition 3.2 (IE-LWE Assumption). *For parameters $n, p, q, \Gamma_X, \Gamma_r$, and Γ_{Xr}, a key-generation algorithm* Gen, *and an adversary $\mathcal{A}$, we define $\mathcal{A}$'s advantage as*

$$\mathsf{Adv}^{\text{ie-lwe}}_{\mathsf{Gen},\mathcal{A}}(\kappa) := \left| \begin{array}{c} \Pr\left[X \leftarrow \mathsf{Gen}(1^\kappa); r \leftarrow \mathfrak{F}(\Gamma_r, R_{n,q}); e \leftarrow \mathfrak{F}(\Gamma_{Xr}, R_{n,q,p}); Y := Xr + e; \mathcal{A}(X, Y) \to 1\right] \\ - \Pr\left[X \leftarrow \mathsf{Gen}(1^\kappa); Y \leftarrow \mathfrak{F}(\Gamma_{Xr}, R_{n,q}); \mathcal{A}(X, Y) \to 1\right] \end{array} \right|.$$

We say that the IE-LWE assumption on Gen *holds if for any PPT adversary $\mathcal{A}$, its advantage $\mathsf{Adv}^{\text{ie-lwe}}_{\mathsf{Gen},\mathcal{A}}(\kappa)$ is negligible in κ.*

Akiyama et al. showed that the IEC scheme (Gen, Enc, Dec) is IND-CPA secure if the IE-LWE assumption on Gen holds [AGO+18, Theorem 1].

4 Review of Linear-Algebraic Attacks

We review the linear-algebraic attacks in [AGO+18]. In the following, we omit the subscript n from Rot_n and vec_n.

4.1 Key-Recovery Attack

We review the example in the case $\deg X = 1$.

We are given $X(x, y) = a_{00} + a_{10}x + a_{01}y$ and want to find a *small* solution $(u_x, u_y) \in R_{n,q}^2$ satisfying

$$a_{10} \cdot u_x + a_{01} \cdot u_y + a_{00} = 0 \text{ (in } R_{n,q}).$$

This implies

$$\mathrm{vec}(u_x) \cdot \mathrm{Rot}(a_{10}) + \mathrm{vec}(u_y) \cdot \mathrm{Rot}(a_{01}) \equiv \mathrm{vec}(-a_{00}) \pmod{q},$$

that is,

$$\big(\mathrm{vec}(u_x), \mathrm{vec}(u_y)\big) \cdot \begin{pmatrix} \mathrm{Rot}(a_{10}) \\ \mathrm{Rot}(a_{01}) \end{pmatrix} \equiv \mathrm{vec}(-a_{00}) \pmod{q}.$$

Therefore, we let

$$A_{\mathrm{kr1}} = [\mathrm{Rot}(a_{10})^\top \mid \mathrm{Rot}(a_{01})^\top] \in \mathbb{Z}_q^{n \times 2n}$$

and consider the lattice

$$\begin{aligned} \Lambda^\perp(A_{\mathrm{kr1}}) &= \{v \in \mathbb{Z}^{2n} \mid v \cdot A_{\mathrm{kr1}}^\top \equiv 0 \pmod{q}\} \\ &= \{(v_x, v_y) \in \mathbb{Z}^{2n} \mid v_x \cdot \mathrm{Rot}(a_{10}) + v_y \cdot \mathrm{Rot}(a_{01}) \equiv 0 \pmod{q}\}. \end{aligned}$$

Now, we consider a target vector $t \in \mathbb{Z}^{2n}$, an arbitrary solution of $t \cdot A_{\mathrm{kr1}}^\top \equiv \mathrm{vec}(-a_{00}) \pmod{q}$. Solving the CVP instance $(\Lambda^\perp(A_{\mathrm{kr1}}), t)$, we obtain a vector $w \in \Lambda^\perp(A_{\mathrm{kr1}})$. We let $\bar{u} = (\mathrm{vec}(u_x), \mathrm{vec}(u_y)) := t - w$.

We have $\bar{u} \cdot A_{\mathrm{kr1}}^\top \equiv \mathrm{vec}(-a_{00}) \pmod{q}$ because $\bar{u} = t - w$. In addition, we expect that the norm of $\bar{u}$ is small, since w is the close vector to t and $\bar{u}$ is the difference.

Remark 4.1. In the case of $\deg X = \deg r = 2$, we have $X(x, y) = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2$ and consider a matrix

$$A_{\mathrm{kr2}} = [\mathrm{Rot}(a_{10})^\top \mid \mathrm{Rot}(a_{01})^\top \mid \mathrm{Rot}(a_{20})^\top \mid \mathrm{Rot}(a_{11})^\top \mid \mathrm{Rot}(a_{02})^\top] \in \mathbb{Z}_q^{n \times 5n}.$$

4.2 Message-Recovery Attack

We again review the example in the case $\deg X = 1$ and $\deg r = 1$.

Let us consider $f(x, y) = p \cdot e(x, y) + m \in \mathfrak{F}(\Gamma_{Xr}, R_{n,q})$. The ciphertext c of m has the relation

$$\sum_{(i,j) \in \Gamma_{Xr}} c_{ij} x^i y^j = \left(\sum_{(i,j) \in \Gamma_X} a_{ij} x^i y^j \right) \cdot \left(\sum_{(i,j) \in \Gamma_r} r_{ij} x^i y^j \right) + \left(\sum_{(i,j) \in \Gamma_{Xr}} f_{ij} x^i y^j \right). \tag{2}$$

Let us consider the following matrix

$$A_{\mathrm{mr1}} = \begin{array}{c} \\ 1 \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^2 & xy & y^2 \end{array} \\ \begin{pmatrix} A_{00} & A_{10} & A_{01} & & & \\ & A_{00} & & A_{10} & A_{01} & \\ & & A_{00} & & A_{10} & A_{01} \end{pmatrix} \end{array} \in \mathbb{Z}^{3n \times 6n},$$

where $A_{ij} := \mathrm{Rot}(a_{ij}) \in \mathbb{Z}^{n \times n}$. Let

$$\begin{aligned} \bar{r} &:= \big(\mathrm{vec}(r_{00}), \mathrm{vec}(r_{10}), \mathrm{vec}(r_{01})\big) \in \mathbb{Z}^{3n}, \\ \bar{f} &:= \big(\mathrm{vec}(f_{00}), \mathrm{vec}(f_{10}), \mathrm{vec}(f_{10}), \mathrm{vec}(f_{20}), \mathrm{vec}(f_{11}), \mathrm{vec}(f_{02})\big) \in \mathbb{Z}^{6n}, \\ \bar{c} &:= \big(\mathrm{vec}(c_{00}), \mathrm{vec}(c_{10}), \mathrm{vec}(c_{10}), \mathrm{vec}(c_{20}), \mathrm{vec}(c_{11}), \mathrm{vec}(c_{02})\big) \in \mathbb{Z}^{6n}. \end{aligned}$$

According to Equation 2, we have

$$\bar{c} \equiv \bar{r} \cdot A_{\mathrm{mr1}} + \bar{f} \pmod{q}.$$

Now, we consider a lattice

$$\Lambda_q(A_{\mathrm{mr1}}) = \{z \in \mathbb{Z}^{6n} \mid z \equiv sA_{\mathrm{mr1}} \pmod{q} \text{ for some } s \in \mathbb{Z}^{3n}\}$$

and a target vector $\bar{c} \in \mathbb{Z}^{6n}$. Solving the CVP instance $(\Lambda_q(A_{\mathrm{mr1}}), \bar{c})$, we obtain $w \in \Lambda_q(A_{\mathrm{mr1}})$. We let $\bar{v} := \bar{c} - w$.

Now, we have $\bar{c} \equiv sA + \bar{v} \pmod{q}$ for some $s \in \mathbb{Z}^{3n}$ and expect that $\bar{v}$ is small. If we obtain $\bar{v} = \bar{f}$, we finally obtain m by taking it modulo p.

Remark 4.2. In the case of $\deg X = \deg r = 2$, we will consider a matrix

$$A_{\mathrm{mr2}} = \begin{array}{c} \\ 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & x & y & \dots & x^4 & x^3y & x^2y^2 & xy^3 & y^4 \end{array} \\ \begin{pmatrix} A_{00} & A_{10} & A_{01} & & & & & & \\ & A_{00} & & & & & & & \\ & & A_{00} & & & & & & \\ & & & & A_{20} & A_{11} & A_{02} & & \\ & & & & & A_{20} & A_{11} & A_{02} & \\ & & & & & & A_{20} & A_{11} & A_{02} \end{pmatrix} \end{array} \in \mathbb{Z}^{6n \times 15n},$$

and solve the CVP instance with $15n$-dimensional lattice.

Experimental Results: Akiyama et al. estimate IEC's security by mounting these attacks against the small parameter sets $n = 10, 20, \dots, 60$ for $\deg X = 1$ and $n = 10, 20, 30, 40$ for $\deg X = 2$. Their environment is

- CPU: AMD Opteron(TM) Processor 848
- Memory: 64 GB
- OS: Linux version 2.6.18-406.el5.centos.plus
- Software: Magma Ver2.21-5

They also define q as small as possible.

They mount a key-recovery attack, which succeeds if and only if $(u_x, u_y) \in R_{n,q,p}$ satisfying $X(u_x, u_y) = 0$ is found. In their experiments, the key-recovery attack for $\deg X = 1$ failed for $n \geq 50$ and that for $\deg X = 2$ failed even for $n \geq 10$.

They also mount a message-recovery attack, which, given X and $Xr+e$, succeeds if and only if $e = (e_1, \dots, e_{6n})$ with $e_i \in [0, p-1]$ is found. The message-recovery attack for $\deg X = 1$ failed for $n \geq 50$. Curiously, the attack for $\deg X = 2$ succeed to find short e even for $n = 40$. (They seem stop their experiment due to time constraint. Their experiment took about 230000 seconds $\approx$ 2.7 days to process a 600-dimensional lattice.)

5 Review of Gentry's Attack

We review Gentry's attack against NTRU-Composite [Sil01]. Let us consider NTRU's key generation and encryption: Roughly speaking, we choose a secret key $(f, g) \in R_{n,q,p}^2$ and compute a public key as $h = g/f \in R_{n,q}$. The ciphertext of plaintext $m \in R_{n,q,p}$ with randomness $r \in R_{n,q,p}$ is $c = phr + m \in R_{n,q}$.

Lattice Attack: Coppersmith and Shamir [CS97] pointed out that a short vector $(\mathrm{vec}_n(f), \mathrm{vec}_n(g)) \in \mathbb{Z}^{2n}$ is in a lattice spanned by a matrix

$$L_{CS} := \begin{pmatrix} \mathrm{Rot}_n(1) & \mathrm{Rot}_n(h) \\ \mathrm{Rot}_n(0) & \mathrm{Rot}_n(q) \end{pmatrix} \in \mathbb{Z}^{2n \times 2n}.$$

We have $h = g/f \bmod q$ and this implies $fh + kq = g$ for some $k \in R_n$. Therefore, $(\mathrm{vec}_n(f), \mathrm{vec}_n(k)) \cdot L_{CS} = (\mathrm{vec}_n(f), \mathrm{vec}_n(g))$ as we wanted. Hence, we solve the SVP problem on the lattice and expect to find $(\mathrm{vec}_n(f), \mathrm{vec}_n(g)) \in \mathbb{Z}^{2n}$ as the solution.

Gentry's Attack: Gentry pointed out that there is a ring homomorphism $\theta\colon R_n \to R_d$, where $d \mid n$ is a non-trivial divisor.

Theorem 5.1 ([Gen01, Theorem 1]). *Let n be a composite, and d be a non-trivial divisor of n. The mapping*

$$\theta\colon R_n \to R_d : f = \sum_{i=0}^{n-1} f_i t^i \mapsto \sum_{i=0}^{d-1} \left(\sum_{j=0}^{n/d-1} f_{jd+i} \right) t^i$$

is a ring-homomorphism.

Gentry considered the $2d$-dimensional lattice analogue of $\Lambda(L_{CS})$, the lattice spanned by a matrix

$$L_d = \begin{pmatrix} \mathrm{Rot}_d(1) & \mathrm{Rot}_d(\theta(h)) \\ \mathrm{Rot}_d(0) & \mathrm{Rot}_d(q) \end{pmatrix} \in \mathbb{Z}^{2d \times 2d}.$$

The lattice $\Lambda(L_d)$ contains a short vector $\left(\mathrm{vec}_d(\theta(f)), \mathrm{vec}_d(\theta(g))\right)$, whose norm is approximately equals to that of $\left(\mathrm{vec}_n(f), \mathrm{vec}_n(g)\right)$ (see [Gen01, Appendix A.2]). Therefore, we expect the basis-reduction algorithm, say, LLL or BKZ, finds $\theta(f)$ and $\theta(g)$. We can exploit this partial information $\theta(f)$ as follows:

1. Message-Recovery Attack: We have $\theta(f) \cdot \theta(c) = \theta(f) \cdot \theta(m) + p\theta(r) \cdot \theta(g) \bmod q$. Thus, the expected magnitudes of coefficients of $\theta(f) \cdot \theta(m) + p\theta(r) \cdot \theta(g)$ are small, then we can recover $\theta(m)$.
2. Secret-Key Recover Attack: Using $\theta(f)$ and $\theta(g)$ as hint, we again solve the SVP problem and find (f, g). Indeed, Gentry succeeds to find f in the case of $(n, q, p) = (256, 127, 2)$ in his experiment.

6 Attacks against Composite n

We employ Gentry's idea. Let us expand the range of the homomorphism $\theta\colon R_n \to R_d$ to

$$\theta\colon R_{n,q}[x, y] \to R_{d,q}[x, y].$$

6.1 Key-Recovery Attack for $\deg X = 1$

We are given $X(x, y) = a_{01}x + a_{01}y + a_{00}$ and want to find a *small* solution $(u_x, u_y) \in R_{n,q}^2$ satisfying

$$a_{10} \cdot u_x + a_{01} \cdot u_y + a_{00} = 0 \text{ (in } R_{n,q}).$$

Applying the homomorphism θ, we have

$$\theta(a_{10}) \cdot \theta(u_x) + \theta(a_{01}) \cdot \theta(u_y) + \theta(a_{00}) = 0 \text{ (in } R_{d,q}).$$

Thus, we can try to find $(\theta(u_x), \theta(u_y))$ by using the lattice-basis reduction algorithms on the lattice of dimension $2d$ ($< 2n$).

The concrete attack consists of two sub-attacks, finding $\theta(u_x)$ and $\theta(u_y)$ and finding u_x and u_y by using those hints. The details follow.

Finding $\theta(u_x)$ and $\theta(u_y)$: We set

$$A_{\mathrm{kr1,d}} = [\mathrm{Rot}_d(\theta(a_{10}))^\top | \mathrm{Rot}_d(\theta(a_{01}))^\top] \in \mathbb{Z}_q^{d \times 2d}$$

and want to find a short vector v_d satisfying

$$v_d \cdot A_{\mathrm{kr1,d}}^\top \equiv \mathrm{vec}_d(-\theta(a_{00})) \pmod q. \tag{3}$$

We consider a lattice $\Lambda_q^\perp(A_{\mathrm{kr1,d}})$. Let $t \in \mathbb{Z}^{2d}$ be an arbitrary solution of Equation 3. We solve the CVP instance $(\Lambda_q^\perp(A_{\mathrm{kr1,d}}), t)$ and obtain $w \in \Lambda_q^\perp(A_{\mathrm{kr1,d}})$.

Now, we have "short" $\bar{v}_d := t - w$ satisfying Equation 3. Let us interpret the vector $\bar{v}_d$ as the pair of polynomials $(v_x^{(d)}, v_y^{(d)}) \in R_{d,q}^2$. and assume that $v_x^{(d)} = \theta(u_x)$ and $v_y^{(d)} = \theta(u_y)$.

We have $\mathrm{vol}(\Lambda_q^\perp(A_{\mathrm{kr1,d}})) = q^d$, $\gamma \approx \sqrt{2d/(2\pi e)} \cdot \mathrm{vol}(\Lambda_q^\perp(A_{\mathrm{kr1,d}}))^{1/2d} = \sqrt{d/\pi e} \cdot q^{1/2}$, and $\|v_d\| \le 2p\sqrt{2d}$. Since $\gamma \gg \|v_d\|$, that is, the target vector is very shorter than the expected length of the shortest vector, we expect that the LLL/BKZ algorithm can find v_d.

Finding u_x and u_y: We already have a hint $(\theta(u_x), \theta(u_y))$. In this paper, we consider a simpler method than Gentry's one: We set

$$A_{\mathrm{kr1,hint}} = \begin{bmatrix} \mathrm{Rot}_n(a_{10})^\top & \mathrm{Rot}_n(a_{01})^\top \\ \underbrace{I_d \ \cdots \ I_d}_{n/d} & \\ & \underbrace{I_d \ \cdots \ I_d}_{n/d} \end{bmatrix} \in \mathbb{Z}_q^{(n+2d)\times 2n}$$

and try to find a short vector v satisfying

$$v \cdot A_{\mathrm{kr1,hint}}^\top \equiv \left(\mathrm{vec}_n(-a_{00}), \mathrm{vec}_d(\theta(u_x)), \mathrm{vec}_d(\theta(u_y))\right) \pmod q. \tag{4}$$

We again consider a lattice $\Lambda_q^\perp(A_{\mathrm{kr1,hint}})$. Let $t \in \mathbb{Z}^{2n}$ be an arbitrary solution of Equation 4. We solve the CVP instance $(\Lambda_q^\perp(A_{\mathrm{kr1,hint}}), t)$ and obtain w. Now, we have a short vector $\bar{v} := t - w$ satisfying Equation 4.

Interpreting the vector $\bar{v}$ as the pair of polynomials $(u_x, u_y) \in R_{n,q}^2$, we have $a_{10} \cdot u_x + a_{01} \cdot u_y + a_{00} = 0$ in $R_{n,q}$ as we wanted.

We have $\mathrm{vol}(\Lambda_q^\perp(A_{\mathrm{kr1,hint}})) = q^{n+d}$. $\gamma \approx \sqrt{2n/(2\pi e)} \cdot \mathrm{vol}(\Lambda_q^\perp(A_d))^{1/2n} = \sqrt{d/\pi e} \cdot q^{1+d/n}$, and $\|\bar{v}\| \le p\sqrt{2n}$. Since $\gamma \gg \|\bar{v}\|$, we expect that the LLL/BKZ algorithm can find the target vector $\bar{v}$.

6.2 Partial-Message-Recovery Attack for $\deg X = 1$

We try to find $\theta(m) \bmod p$ from a ciphertext c of m. If so, it easily breaks the IND-CPA security of the IEC scheme.

For simplicity, we define $f(x, y) = pe(x, y) + m$, which results in $\theta(f) = p\theta(e) + \theta(m)$. Since θ is a ring homomorphism from $R_n[x, y] \to R_d[x, y]$, we have

$$\theta(c) = \theta(r) \cdot \theta(X) + \theta(f).$$

Let us consider the following matrix:

$$A_{\mathrm{pmr1,d}} := \begin{array}{c} \\ 1 \\ x \\ y \end{array} \!\!\begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^2 & xy & y^2 \end{array} \\ \begin{pmatrix} A'_{00} & A'_{10} & A'_{01} & & & \\ & A'_{00} & & A'_{10} & A'_{01} & \\ & & A'_{00} & & A'_{10} & A'_{01} \end{pmatrix} \end{array} \in \mathbb{Z}^{3d\times 6d},$$

where $A'_{ij} := \mathrm{Rot}_d(\theta(a_{ij})) \in \mathbb{Z}^{d\times d}$. Let

$$\begin{aligned} \bar{r}_d &:= \left(\mathrm{vec}_d(\theta(r_{00})), \mathrm{vec}_d(\theta(r_{10})), \mathrm{vec}_d(\theta(r_{01}))\right) \in \mathbb{Z}^{3d}, \\ \bar{c}_d &:= \left(\mathrm{vec}_d(\theta(c_{00})), \mathrm{vec}_d(\theta(c_{10})), \mathrm{vec}_d(\theta(c_{01})), \mathrm{vec}_d(\theta(c_{20})), \mathrm{vec}_d(\theta(c_{11})), \mathrm{vec}_d(\theta(c_{02}))\right) \in \mathbb{Z}^{6d}, \\ \bar{f}_d &:= \left(\mathrm{vec}_d(\theta(f_{00})), \mathrm{vec}_d(\theta(f_{10})), \mathrm{vec}_d(\theta(f_{01})), \mathrm{vec}_d(\theta(f_{20})), \mathrm{vec}_d(\theta(f_{11})), \mathrm{vec}_d(\theta(f_{02}))\right) \in \mathbb{Z}^{6d}. \end{aligned}$$

We have

$$\bar{c}_d \equiv \bar{r}_d \cdot A_{\mathrm{pmr1,d}} + \bar{f}_d \pmod q.$$

Now, we consider a lattice $\Lambda_q(A_{\mathrm{pmr1,d}})$ and solve the CVP instance $(\Lambda_q(A_{\mathrm{pmr1,d}}), \bar{c}_d)$ and obtain $\bar{v}_d$. Let us interpret the vector $\bar{v}_d$ as a tuple of polynomials $(v_{00}, v_{10}, v_{01}, v_{20}, v_{11}, v_{02}) \in R_{d,q}^6$. Suppose that we have $\bar{v}_d = \bar{f}_d$, if so, we have $v_{00} = \theta(f_{00})$ and, thus,

$$v_{00} \equiv \theta(f_{00}) \equiv p\theta(e_{00}) + \theta(m) \equiv \theta(m) \pmod p.$$

We have $\mathrm{vol}(\Lambda_q^\perp(A_{\mathrm{pmr1,d}})) = q^{3d}$, $\gamma \approx \sqrt{6d/(2\pi e)} \cdot \mathrm{vol}(\Lambda_q^\perp(A_d))^{1/6d} = \sqrt{3d/\pi e} \cdot q^{1/2}$, and $\|\bar{v}_d\| \le (n/d)p^2\sqrt{6d}$. We expect that the LLL/BKZ algorithm can find $\bar{v}_d$, because $\gamma \gg \|\bar{v}_d\|$.

6.3 Partial-Message-Recovery Attack for $\deg X = 2$

In the case of $\deg X = \deg r = 2$, we consider a matrix

$$A_{\mathrm{pmr2,d}} = \begin{array}{c} \\ 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{array} \!\!\begin{array}{c} \begin{array}{cccccccc} 1 & x & y & \ldots & x^4 & x^3y & x^2y^2 & xy^3 & y^4 \end{array} \\ \begin{pmatrix} A'_{00} & A'_{10} & A'_{01} & & & & & & \\ & A'_{00} & & & & & & & \\ & & A'_{00} & & & & & & \\ & & & & A'_{20} & A'_{11} & A'_{02} & & \\ & & & & & A'_{20} & A'_{11} & A'_{02} & \\ & & & & & & A'_{20} & A'_{11} & A'_{02} \end{pmatrix} \end{array} \in \mathbb{Z}^{6d\times 15d},$$

where $A'_{ij} := \mathrm{Rot}_d(\theta(a_{ij})) \in \mathbb{Z}^{d \times d}$. By the similar way, we solve the CVP instance $(\Lambda_q(A_{\mathrm{pmr2,d}}), \bar{c}_d)$ and obtain $\bar{v}_d$, which corresponding to a tuple of polynomials $(v_{00}, v_{10}, \ldots, v_{04}) \in R^{15}_{d,q}$. We output v_{00} mod p as $\theta(m)$ mod p.

We have $\mathrm{vol}(\Lambda_q^{\perp}(A_{\mathrm{pmr2,d}})) = q^{9d}$, $\gamma \approx \sqrt{15d/(2\pi e)} \cdot \mathrm{vol}(\Lambda_q^{\perp}(A_d))^{1/15d} = \sqrt{15d/2\pi e} \cdot q^{3/5}$, and $\|\bar{v}_d\| \leq (n/d)p^2\sqrt{15d}$. We expect that the LLL/BKZ algorithm can find $\bar{v}_d$ because $\gamma \gg \|\bar{v}_d\|$.

7 Attacks against Prime n

After reporting the previous attacks to the authors of [AGO+18], they set n as a prime, say, $n = 83$ (and $q = 68339982247$) [Aki17]. In this section, we propose a sub-ring attack, which is applicable to the case that n is a prime.

(Non-trivial) subring: Notice that $R_{n,q}[x]$ is a subring of $R_{n,q}[x, y]$. We consider a ring homomorphism

$$\pi \colon R_{n,q}[x, y] \mapsto R_{n,q}[x] \colon f(x, y) \mapsto f(x, 0).$$

We have the relation $c(x, y) = r(x, y) \cdot X(x, y) + f(x, y)$, where $f(x, y) = pe(x, y) + m$. Applying the ring homomorphism π, we obtain

$$\pi(c) \equiv \pi(r) \cdot \pi(X) + \pi(f) \equiv \pi(r) \cdot \pi(X) + p \cdot \pi(e) + m \pmod q \tag{5}$$

and notice that the max norm of $\pi(f)$ is at most that of $f = p \cdot e + m$.

7.1 Message-Recovery Attack against deg $X = 1$

Let us recall the message-recovery attack against deg $X = 2$ in subsection 4.2. We consider

$$A_{\mathrm{mr1}} := \begin{array}{c} \\ 1 \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^2 & xy & y^2 \end{array} \\ \begin{pmatrix} A_{00} & A_{10} & A_{01} & & & \\ & A_{00} & & A_{10} & A_{01} & \\ & & A_{00} & & A_{10} & A_{01} \end{pmatrix} \end{array} \in \mathbb{Z}^{3n \times 6n},$$
$$\bar{c} := (\mathrm{vec}_n(c_{00}), \mathrm{vec}_n(c_{10}), \mathrm{vec}_n(c_{01}), \mathrm{vec}_n(c_{20}), \mathrm{vec}_n(c_{11}), \mathrm{vec}_n(c_{02})) \in \mathbb{Z}^{6n},$$

where $A_{ij} := \mathrm{Rot}_n(a_{ij}) \in \mathbb{Z}^{n \times n}$, and try to solve the CVP instance $(\Lambda_q(A_{\mathrm{mr1}}), \bar{c})$ to find $\bar{f}$.

In the lattice-based attacks, we often shorten the basis of the lattice and the target vector to reduce the dimension. Here, we give another approach to shorten them.

Concrete Attack: Deleting the rows and columns whose indices contain y from A and $\bar{c}$, we obtain

$$A'_{\mathrm{mr1}} := \begin{array}{c} \\ 1 \\ x \end{array} \begin{array}{c} \begin{array}{ccc} 1 & x & x^2 \end{array} \\ \begin{pmatrix} A_{00} & A_{10} & \\ & A_{00} & A_{10} \end{pmatrix} \end{array} \in \mathbb{Z}^{2n \times 3n},$$
$$\bar{c}' := \left(\mathrm{vec}_n(c_{00}), \mathrm{vec}_n(c_{10}), \mathrm{vec}_n(c_{20})\right) \in \mathbb{Z}^{5n}.$$

Letting

$$\bar{r}' = (\mathrm{vec}_n(r_{00}), \mathrm{vec}_n(r_{10})) \in \mathbb{Z}^{2n},$$
$$\bar{f}' = (\mathrm{vec}_n(f_{00}), \mathrm{vec}_n(f_{10}), \mathrm{vec}_n(f_{20})) \in \mathbb{Z}^{3n},$$

we have

$$\bar{c}' \equiv \bar{r}' \cdot A'_{\mathrm{mr1}} + \bar{f}' \pmod q,$$

which corresponds to Equation 5. Thus, solving the CVP instance $(\Lambda_q(A'_{\mathrm{mr1}}), \bar{c}')$, we expect to find $\bar{f}'$ and obtain $m := \mathrm{vec}_n(f_{00})$ mod p.

Gaussian Heuristic: This shortening reduces the dimension of the lattice from $5n = 415$ to $3n = 249$. We have $\mathrm{vol}(\Lambda_q(A'_{\mathrm{mr2}})) = q^n$ and $\gamma \approx \sqrt{3n/(2\pi e)} \cdot \mathrm{vol}(\Lambda_q(A'))^{1/3n} = \sqrt{3n/2\pi e} \cdot q^{1/3}$ and $\|\bar{f}'\| \le p^2\sqrt{3n}$. In our parameter setting, we have $\gamma \approx 380.81$ and $\|\bar{f}'\| \le 142.02$ and the gap between γ and $\|\bar{f}'\|$ is not so large. Thus it seems hard to find $\bar{f}'$ in this setting.

7.2 Message-Recovery Attack against $\deg X = 2$

Let us recall the message-recovery attack against $\deg X = 2$ in subsection 4.2. We consider $A_{\mathrm{mr2}} \in \mathbb{Z}^{6n \times 15n}$ and $\bar{c} := (\mathrm{vec}_n(c_{00}), \mathrm{vec}_n(c_{10}), \mathrm{vec}_n(c_{01}), \ldots, \mathrm{vec}_n(c_{04})) \in \mathbb{Z}^{15n}$, and try to solve the CVP instance $(\Lambda_q(A_{\mathrm{mr2}}), \bar{c})$ to find $\bar{f}$.

Concrete Attack: Deleting the rows and columns whose indices contain y from A and $\bar{c}$, we obtain

$$
A'_{\mathrm{mr2}} := \begin{array}{c} \\ 1 \\ x \\ x^2 \end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & x & x^2 & x^3 & x^4 \end{array} \\
\begin{pmatrix}
A_{00} & A_{10} & A_{20} & & \\
 & A_{00} & A_{10} & A_{20} & \\
 & & A_{00} & A_{10} & A_{20}
\end{pmatrix}
\end{array} \in \mathbb{Z}^{3n \times 5n},
$$

$$
\bar{c}' := (\mathrm{vec}_n(c_{00}), \mathrm{vec}_n(c_{10}), \mathrm{vec}_n(c_{20}), \mathrm{vec}_n(c_{30}), \mathrm{vec}_n(c_{40})) \in \mathbb{Z}^{5n}.
$$

Letting

$$
\begin{aligned}
\bar{r}' &= (\mathrm{vec}_n(r_{00}), \mathrm{vec}_n(r_{10}), \mathrm{vec}_n(r_{20})) \in \mathbb{Z}^{3n}, \\
\bar{f}' &= (\mathrm{vec}_n(f_{00}), \mathrm{vec}_n(f_{10}), \mathrm{vec}_n(f_{20}), \mathrm{vec}_n(f_{30}), \mathrm{vec}_n(f_{40})) \in \mathbb{Z}^{5n},
\end{aligned}
$$

we have

$$
\bar{c}' \equiv \bar{r}' \cdot A'_{\mathrm{mr2}} + \bar{f}' \pmod{q},
$$

which corresponds to Equation 5. Thus, solving the CVP instance $(\Lambda_q(A'_{\mathrm{mr2}}), \bar{c}')$, we expect to find $\bar{f}'$ and obtain $m := \mathrm{vec}_n(f_{00}) \bmod p$.

Gaussian Heuristic: We note that this shortening reduces the dimension of the lattice from $15n = 1243$ to $5n = 415$. We have $\mathrm{vol}(\Lambda_q(A'_{\mathrm{mr2}})) = q^{2n}$ and $\gamma \approx \sqrt{5n/(2\pi e)} \cdot \mathrm{vol}(\Lambda_q(A'))^{1/5n} = \sqrt{5n/2\pi e} \cdot q^{2/5}$ and $\|\bar{f}'\| \le p^2\sqrt{5n}$. In our parameter setting, $\gamma \approx 106330.25$ and $\|\bar{f}'\| \le 183.35$. We expect that the LLL/BKZ algorithm can find a short vector $\bar{f}'$ because of this large gap.

7.3 Distinguishing Attack for $\deg X = 1$ and $\deg X = 2$

Further, we try to falsify the IE-LWE assumption, that is to distinguish $(X, c) = (X, Xr + e)$ from (X, u). In order to do so, we try to find a short vector $\bar{v}'$ from $\Lambda_q(A'_{\mathrm{mr1}})$. If c is $Xr + e$, then we have $\langle \bar{c}', \bar{v}' \rangle \bmod q$ is "short," while if c is chosen uniformly at random, then $\langle \bar{c}', \bar{v}' \rangle \bmod q$ is distributed according to the uniform distribution over $\mathbb{Z}_q$.

This can also be applied to the case of $\deg X = 2$.

8 Experiments

We run our experiment on a virtual machine on our company's internal private cloud. Our environment is

- CPU: QEMU Virtual CPU version 2.5+
- Memory: 32GB
- OS: CentOS7 (Linux version 3.10.0-693.5.2.el7.x86_64)
- Software: SageMath version 8.0

8.1 Key-Recovery Attack for $\deg X = 1$

We mount our attack in subsection 6.1 with $n = 80$ and $d = 40$. We employ the default BKZ algorithm in SageMath 8.0 as the lattice-basis reduction algorithm and the rounding algorithm to solve the CVP instance. We generate 100 key pairs and try to find a pair $(u_x, u_y) \in R_{n,q,p}^2$ satisfying $X(u_x, u_y) = 0$. In our experiment, 84 secret keys are found from 100 public keys. The attack used an average CPU time of 32.68 seconds per key on a single core of our server. (min: 29.16, avg: 32.68, med: 32.54, max: 39.11)

The script is in Listing 1.2. We did not check the other settings, say, $d = 20$ or $d = 16$.

8.2 Partial-Message-Recovery Attack for $\deg X = 1$

We mount our attack in subsection 6.2 with $n = 80$ and $d = 10$. We employ the default BKZ algorithm with block size 10 as the lattice-basis reduction algorithm and the embedding algorithm to solve the CVP instance. We generate 100 pairs of a public key and a random ciphertext on a random plaintext. In our experiment, all partial message $\theta(m) \bmod p$ are recovered. The attack used an average CPU time of 0.47 seconds per key on a single core of our server. (min: 0.29, avg: 0.47, med: 0.46, max: 0.73)

8.3 Partial-Message-Recovery Attack for $\deg X = 2$

We mount our attack in subsection 6.3 with $n = 80$ and $d = 10$. We employ the default BKZ algorithm as the lattice-basis reduction algorithm and the embedding algorithm to solve the CVP instance. We generate 100 pairs of a public key and a random ciphertext on a random plaintext. In our experiment, all partial message $\theta(m) \bmod p$ are recovered. The attack used an average CPU time of 33.40 seconds per key on a single core of our server. (min: 20.95, avg: 33.40, med: 32.41, max: 84.77)

8.4 Message-Recovery Subring Attack for $\deg X = 2$

We mount our attack in subsection 7.2 with $n = 83$ (and $q = 68339982247$). We employ the BKZ algorithm with options `block_size=10,fp="rr",precision=150` as the lattice-basis reduction algorithm and the embedding algorithm to solve the CVP instance. We generate 10 pairs of a public key and a random ciphertext on a random plaintext. In our experiment, all message m are recovered. The attack used an average CPU time of 54842.55 seconds per key on a single core of our server. (min: 51481.51, avg: 54842.55, med: 54127.69, max: 61770.88)

8.5 Distinguishing Subring Attack for $\deg X = 2$

We mount our attack in subsection 7.3 with various prime n with $p = 3$ and a smallest prime q satisfying Equation 1. We generate 10 public keys on each $n \in \{83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149\}$ and try to find a short vector $\bar{v}'$ in the lattice $\Lambda_q(A'_{\mathrm{mr2}})$. We employ the BKZ algorithm with options `block_size=10`, `fp="rr"`, `precision=150` up to $n = 113$ and `block_size=10`, `fp="rr"`, `precision=200` for $n \geq 127$ as the lattice-basis reduction algorithm.

The timing results are summarized in Figure 1 and the qualities of $\bar{v}'$ are summarized in Figure 2. The attack on $n = 83, 113, 149$ used an average CPU time of 57471.10, 309815.82, 762618.22 seconds per key. The attack on $n = 83, 113$ found short vectors $\bar{v}'$ such that the average of ratio $\|\bar{v}'\|/q$ is 0.021, and 0.11. In the case of $n = 149$, we fail to find short vectors $\bar{v}'$.

We check the quality of $\bar{v}'$ as follows. We generate 50000 random errors $e_i(x, y) \in \mathfrak{F}(\Gamma_{Xr}, R_{n,q}, p)$ and 50000 random polynomials $u_i(x, y) \in \mathfrak{F}(\Gamma_{Xr}, R_{n,q})$. We then compute compute $\delta_i := \bar{v}' \cdot \bar{e}_i \bmod_c q$ and $\xi_i := \bar{v}' \cdot \bar{u}_i \bmod_c q$, where we denote by $\bmod_c$ the centered modulo operator. We check how they vary.

For example, in the case of $n = 113$, we take the worst vector $\bar{v}'$ with $\|\bar{v}'\|/q = 0.12$. Although this is the worst vector, it is enough to distinguish the errors from uniform as the histogram in Figure 3 shows.

9 Conclusion

In this paper, we propose two strategies to reduce the dimension of lattices in lattice-based attacks. The first one is for composite n and is inspired by Gentry's attack [Gen01] against NTRU Composite [Sil01]. This strategy exploits the ring homomorphism $\theta \colon R_{n,q}[x, y] \to R_{d,q}[x, y]$ to reduce the dimension of lattices in lattice-based

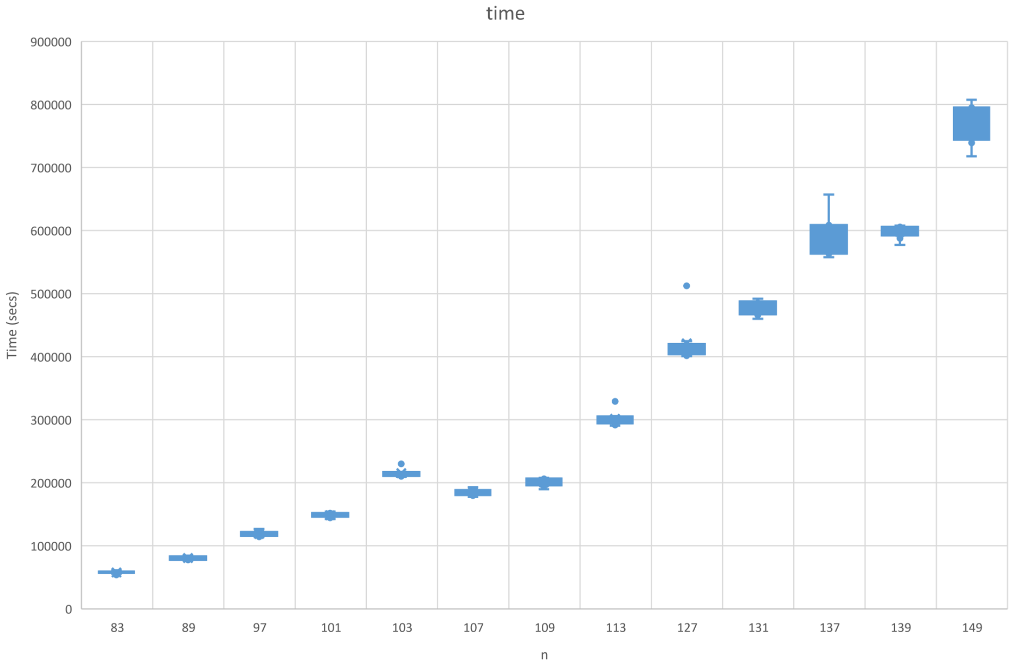

Fig. 1: Summary of Running Time

attacks. The second one is for prime n and exploits another class of subring $R_{n,q}[x]$ of $R_{n,q}[x, y]$ to reduce the dimension. The message-recovery attack succeeds in the case $\deg X = 2$ but fails in the case $\deg X = 1$. The distinguishing attack also succeeds in larger n, say, $n = 113$.

We finally note that we have already reported our attacks to Akiyama et al. and the parameter settings in their paper on Cryptology ePrint Archive [AGO+17b] and NIST PQC submission [AGO+17a] reflected our attacks. They further investigated lattice-based attacks and estimated the security by following the security-estimation methods of the LWE problems [AGVW17,ADPS16,BDGL15,Che13]. See their paper for details.

Acknowledgment

The author would like to thank Akiyama, Goto, Okumura, Takagi, Nuida, and Hanaoka for their kindness and fruitful discussions. The author would like to thank anonymous reviewers of PQCrypto 2018 for their valuable comments. The author finally would like to thank the XFARM Team for providing their could.

References

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security Symposium 2016*, pages 327–343. USENIX Association, 2016. See also https://eprint.iacr.org/2015/1092. 12

AG06. Koichiro Akiyama and Yasuhiro Goto. A public-key cryptosystem using algebraic surfaces. In *PQCrypto 2006*, pages 119–138, 2006. Available at http://postquantum.cr.yp.to/. 1

AGM09. Koichiro Akiyama, Yasuhiro Goto, and Hideyuki Miyake. An algebraic surface cryptosystem. In Stanisław Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 425–442. Springer, Heidelberg, 2009. 1

AGO+17a. Koichiro Akiyama, Yasuhiro Goto, Shinya Okumura, Tsuyoshi Takagi, Koji Nuida, Goichiro Hanaoka, Hideo Shimizu, and Yasuhiko Ikematsu. Giophantus. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. 2, 12

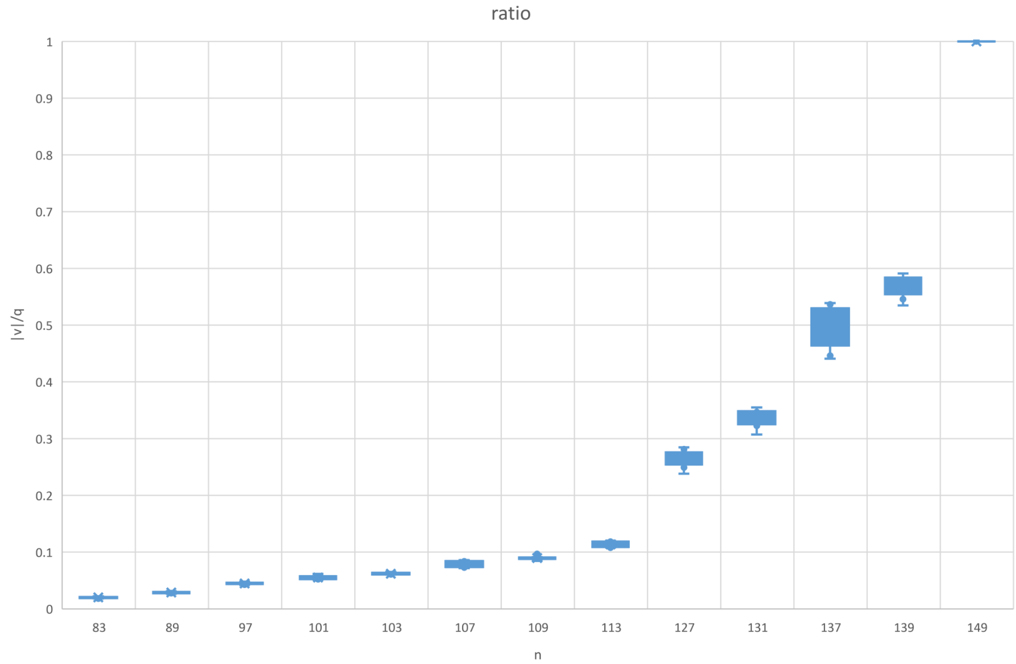

Fig. 2: Summary of Ratio $\|\bar{v}'\|/q$

AGO⁺17b. Koichiro Akiyama, Yasuhiro Goto, Shinya Okumura, Tsuyoshi Takagi, Koji Nuida, Goichiro Hanaoka, Hideo Shimizu, and Yasuhiko Ikematsu. A public-key encryption scheme based on non-linear indeterminate equations (Giophantus). Cryptology ePrint Archive, Report 2017/1241, 2017. Available at https://eprint.iacr.org/2017/1241. 2, 12

AGO⁺18. Koichiro Akiyama, Yasuhiro Goto, Shinya Okumura, Tsuyoshi Takagi, Koji Nuida, and Goichiro Hanaoka. A public-key encryption scheme based on non-linear indeterminate equations. In *SAC 2017 – Proceedings of the 24th Annual Conference on Selected Areas in Cryptography*, volume 10719 of *Lecture Notes in Computer Science*, pages 199–218. Springer, Heidelberg, 2018. To appear. 1, 2, 3, 4, 5, 9

AGVW17. Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 297–322. Springer, 2017. 12

Aki17. Koichiro Akiyama. Private communication, October 2017. 2017-10-04. 9

BDGL15. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. *IACR Cryptology ePrint Archive*, 2015:1128, 2015. 12

Che13. Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, 2013. Informatique Paris 7 2013. 12

CS97. Don Coppersmith and Adi Shamir. Lattice attacks on NTRU. In Walter Fumy, editor, *EUROCRYPT '97*, volume 1233 of *LNCS*, pages 52–61. Springer, Heidelberg, 1997. 6

Gen01. Craig Gentry. Key recovery and message attacks on NTRU-composite. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 182–194. Springer, Heidelberg, 2001. 1, 7, 11

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC 2008*, pages 197–206. ACM, 2008. see also https://eprint.iacr.org/2007/432. 3

Sil01. Joseph H. Silverman. Wraps, gaps, and lattice constants. Technical Report 11, version 2, NTRU Cryptosystems, 2001. 1, 6, 11

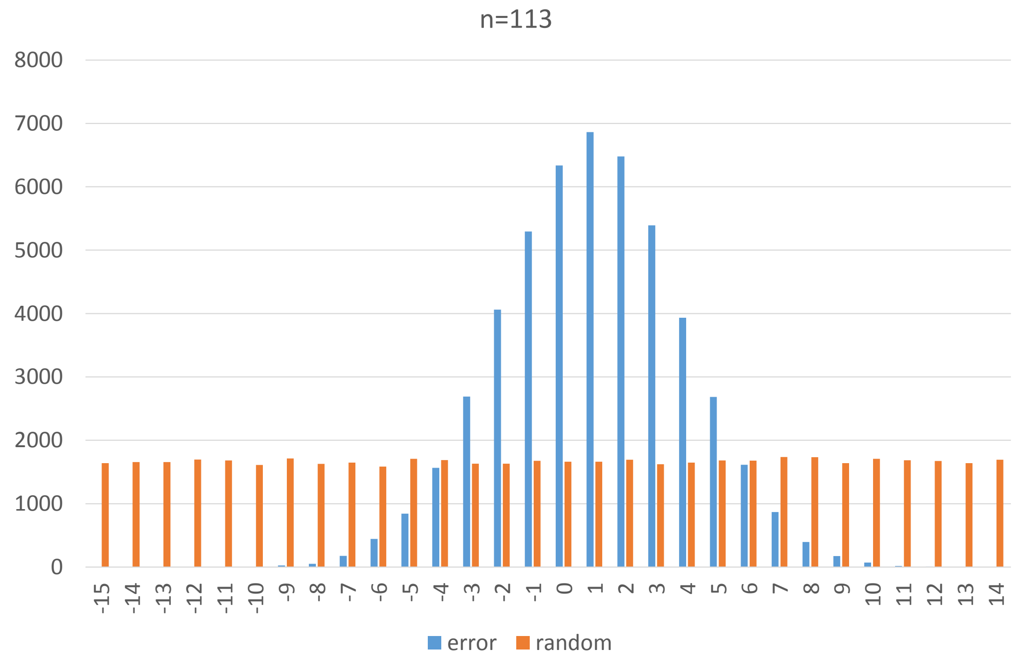

Fig. 3: Histogram of δ_i (blue lines) and ξ_i (orange lines). We count $q/30$

A Implementation

Listing 1.1: ref.sage

```
# Reference Implementation of IEC
# Load this file
# n = 80; p=3; wx=1; wr=1;
# load('ref.sage')

# Parameters ====================
def gen_G(upper_bound, lower_bound):
    # compare with total deg. if equal, (1,0) < (0,1)
    def my_key(a):
        return (a[0] + a[1], a[1], a[0])
    # i for index of x, j for index of y
    l = [(i,j) for j in range(upper_bound+1) \
         for i in range(upper_bound+1) \
         if (lower_bound <= i+j) and (i+j <= upper_bound)]
    return sorted(l, key=my_key)

GX  = gen_G(wx,0); Gr  = gen_G(wr,0)
GXr = gen_G(wx+wr,0); GXp = gen_G(wx,1)

def bd(n,p):
    return len(GXr) * p * (p-1) * (n * (p-1))^(wx+wr)

q = next_prime(bd(n,p))

# Rings ====================
```

```
Zq = Integers(q)
R.<t> = Zq[]
Rq = R.quotient(t^n-1)
F.<x,y> = Rq[]

# Random polys ====================
def random_tpoly(p): return R([randint(0,p-1) for _ in range(n)])

def random_template(p,indices):
    a = 0
    for (i,j) in indices:
        a += Rq(random_tpoly(p)) * x^i * y^j
    return a

def random_r(): return random_template(q,Gr)
def random_e(): return random_template(p,GXr)

# Cryptosystem ====================
def skgen():
    return random_tpoly(p), random_tpoly(p)

def pkgen(ux,uy):
    X = random_template(q,GXp)
    X -= X(ux,uy)
    return X

def encrypt(X,m):
    return Rq(m)+ X * random_r() +p * random_e()

def decrypt(ux,uy,c):
    cu = c(ux,uy)
    mt = cu.lift().change_ring(ZZ).change_ring(Integers(p))
    # output mt in Rq
    return mt.change_ring(Integers(q))
```

Listing 1.2: KRA-PS1.sage

```
# key-recovery attack for PS1

# Make the experiment reproducible
# (at least on given platform/Sage version)
set_random_seed(0)

n = 80; p=3; wx=1; wr=1; d=int(n/2)

load('ref.sage')

# Rings ====================
Rqd = R.quotient(t^d-1)
Fd = F.change_ring(Rqd)

# ====================
def circulant_matrix(b,n):
    # Input: b in Zq[t] (not Zq[t]/(t^n-1) !!!)
    # Output: circlant b
    return matrix(ZZ,n,n,lambda i,j: b.monomial_coefficient(t^((j-i) % n)))

def vectorize(b,n):
    return circulant_matrix(b,n)[0]

def roundingCVP(v,L):
```

```
    l = v * L.inverse()
    lround = vector(map(lambda x: round(x), l))
    w = lround * L
    return w

def recover_folded_sk_roundingCVP(folded_X):
    Alist = []
    for (i,j) in GXp:
        a = folded_X.monomial_coefficient(Fd(x^i*y^j))
        Alist.append(circulant_matrix(a.lift(),d).transpose())
    A_kr = block_matrix(ZZ,1,len(GXp),Alist)
    Z = block_matrix(ZZ,[[A_kr], [q * identity_matrix(d*len(GXp))]])
    Z = Z.echelon_form(include_zero_rows=false)
    L = (q * Z.transpose().inverse()).change_ring(ZZ)
    L = L.BKZ()

    # solve CVP by rounding
    a00 = folded_X.monomial_coefficient(Fd(1)).lift()
    target_vector = vector(Zq,list(vectorize(-a00,d)))
    Aq = A_kr.change_ring(Zq)
    target_t = Aq.solve_right(target_vector).change_ring(ZZ)
    w = roundingCVP(target_t,L)
    v = target_t - w
    return v, R(list(v[0:d])), R(list(v[d:2*d]))

def recover_sk_from_folded_sk(X,vv):
    def f(i,j):
        return 1 if int(n/d) * i <= j and j < int(n/d) * (i+1) else 0

    Alist = []
    for (i,j) in GXp:
        a = X.monomial_coefficient(F(x^i*y^j))
        Alist.append(circulant_matrix(a.lift(),n).transpose())
    A_kr = block_matrix(ZZ,1,len(GXp),Alist)

    Tp = matrix(ZZ,len(GXp),len(GXp)*int(n/d),lambda i,j: f(i,j))
    T = Tp.tensor_product(identity_matrix(d))

    A_kr_hint = block_matrix(ZZ,[[A_kr],[T]])
    Z = block_matrix(ZZ,[[A_kr_hint], [q * identity_matrix(len(GXp)*n)]])
    Z = Z.echelon_form(include_zero_rows=false)
    L = (q * Z.transpose().inverse()).change_ring(ZZ)
    L = L.BKZ()

    # solve CVP by rounding
    a00 = X.monomial_coefficient(F(1)).lift()
    target_vector = vector(Zq,list(vectorize(-a00,n)) + list(vv))
    Aq = A_kr_hint.change_ring(Zq)
    target_t = Aq.solve_right(target_vector).change_ring(ZZ)
    w = roundingCVP(target_t,L)
    v = target_t - w
    return v, R(list(v[0:n])), R(list(v[n:2*n]))


def test(pairs=10,debug=true):
    succ = 0
    tottime = 0.0
    for npair in range(pairs):
        ux,uy = skgen()
        X = pkgen(ux,uy)
        if debug:
```

```
            print "----- Key pair %d -----" % (npair)

        tm = cputime(subprocesses=True)
        folded_X = X.change_ring(Rqd)
        folded_v, folded_ux, folded_uy = recover_folded_sk_roundingCVP(folded_X)
        if debug:
            print "X", X
            print "folded X", folded_X
            print "folded v", folded_v
            print "folded ux", folded_ux
            print "folded uy", folded_uy
        v, ux_cand, uy_cand = recover_sk_from_folded_sk(X,folded_v)
        tottime += float(cputime(tm))
        if debug:
            print "v", v
            print "ux_cand", ux_cand
            print "uy_cand", uy_cand
        if ux == ux_cand and uy == uy_cand:
            succ += 1

    print "===== Results ====="
    print "Total time for extraction: %f seconds." % (tottime)
    print "Average time for extraction: %f seconds." % (tottime/pairs)
    print "Successful recoveries: %d/%d (%f)." % \
            (succ,pairs,RR(100*succ/pairs))


test(100,false)
```