# An Efficient Key Recovery Attack Against NTRUReEncrypt from AsiaCCS 2015

Zijian Song[1,2], Jun Xu[1,2(✉)], Zhiwei Li[1,2], and Dingfeng Ye[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
xujun@iie.ac.cn

[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China

**Abstract.** At AsiaCCS 2015, Nuñez et al. proposed a NTRU-based proxy re-encryption (PRE) scheme, called NTRUReEncrypt. A complete PRE scheme permits the sender to encrypt messages to the proxy, and allows the receiver to decrypt the ciphertexts re-encrypted by the proxy. At PQCrypto 2019, Liu et al. provided cryptanalysis of the scheme based on decryption failures and statistical analysis, both of which need huge amount of ciphertexts. For instance, for ees1171ep1 parameter set, the number of ciphertexts required are $4.68 \cdot 10^{17}$ and $4.83 \cdot 10^{17}$ respectively. In this paper we point out that the security of NTRUReEncrypt would be impacted by an efficient key recovery attack based on linearization technique, it can reduce the number of required ciphertexts drastically. To be specific, two parties sending and receiving messages can recover the other's private key by communicating $O(N + [\frac{N}{2}])$ times, where $N$ is an odd prime in the ring $\mathcal{R} = \mathbb{Z}[x]/\left(x^N - 1\right)$. For specific scheme on parameter sets ees1087ep1, ees1171ep1, ees1499ep1, where $N$ equals 1087, 1171 and 1499 respectively, the amount of ciphertexts used in our attack is only on the order of $10^3$, and our experiments are all completed within one hour on PC. Moreover, we discuss the NTRUReEncrypt instantiated with the NTRU parameter sets in the third round of NIST-PQC competition and give the theoretical analysis.

**Keywords:** NTRUReEncrypt · NTRU · Linearization technique · Key recovery attack

## 1 Introduction

In 1998, Blaze, Bleumer and Strauss [3] proposed a new type of public-key cryptographic scheme, namely proxy re-encryption (PRE) scheme. A complete PRE scheme consists of three parties: the sender Alice, the receiver Bob, and the proxy. It permits Alice to encrypt messages to the proxy, and allows Bob to decrypt the ciphertexts re-encrypted by the proxy. Further, in the communication process, the proxy only provides the re-encryption operation without knowing any information about messages. In fact, proxy re-encryption scheme is a variant of

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
T. Isobe and S. Sarkar (Eds.): INDOCRYPT 2022, LNCS 13774, pp. 644–657, 2022.
https://doi.org/10.1007/978-3-031-22912-1_28

the traditional public key cryptosystem. Its basic algorithm is the same as that of public key encryption scheme, except that there are two more steps: generating proxy re-encryption key and re-encrypting ciphertexts. After a period of development, many PRE schemes have been constructed, but the vast majority of these are based on traditional number theoretic problems, such as discrete logarithm problem [4]. However, these problems suffered the impact after Shor's algorithm [14,15] was put forward. Therefore, the attention turned to the field of post-quantum cryptography, such as lattice-based schemes [1].

At AsiaCCS 2015 [13], Nuñez et al. proposed a NTRU-based proxy re-encryption (PRE) scheme, called NTRUReEncrypt. It only has one more re-encryption step, and the rest is the same as NTRU scheme including parameter sets. In 1996, Hoffstein, Pipher and Silverman proposed a cryptosystem called NTRU [7], which has the advantages of high efficiency and low memory usage. Due to these properties, it becomes an indispensable part of post-quantum cryptography, and has been standardized by IEEE P1363.1 [2]. Recently, it was also submitted to the third round of NIST-PQC competition, e.g. NTRU-HPS, NTRU-HRSS [5]. One reason for the efficiency of NTRU is that, some of polynomials in NTRU have small coefficients, which we will call "small" polynomials for ease of description. The brief process of NTRUReEncrypt scheme is as follows: (1) Alice encrypts the message $m$ as $C_A = h_A * r + m$ by selecting a small polynomial $r$, where Alice's private key is $(f_A, g_A)$ and public key is $h_A = p * g_A * f_A^{-1}$. (2) The proxy selects a small polynomial $e$, encrypts $C_A$ sent by Alice as $C_B = C_A * rk_{A\to B} + p * e$, where $rk_{A\to B} = f_A * f_B^{-1} \bmod q$ is the re-encrypted key of the proxy and $f_B$ is Bob's private key. (3) Bob decrypts $C_B$ sent by the proxy as $(C_B * f_B \bmod q) \bmod p$ to obtain the message $m$.

There are many attacks against NTRU, such as decryption failure attack [8], broadcast attack [6,10]. The method of latter uses the linearization technique, whose main idea is to generate a linear system by linearizing monomials into new variables. At PQCrypto 2019 [11], Liu et al. proposed cryptanalysis of NTRUReEncrypt, whose strategy is based on two points: one is decryption failure, the other is statistical analysis. Due to the huge amount of data required, these two attacks are hard to implement in practice. For instance, for ees1171ep1 parameter set, the number of required ciphertexts are $4.68 \cdot 10^{17}$ and $4.83 \cdot 10^{17}$ respectively.

**Our Contribution.** We present a key recovery attack based on the linearization technique against NTRUReEncrypt, where the parameter sets are from those in AsiaCCS 2015 and PQCrypto 2019. To implement an attack, $O(N + \lceil \frac{N}{2} \rceil)$ legal communications are needed to collect ciphertexts $C_{A_i}$ and $C_{B_i}$, where $N$ is an odd prime in the ring $\mathcal{R} = \mathbb{Z}[x]/\left(x^N - 1\right)$. The comparison of PQCrypto 2019 and our work is shown in Table 1.

The technical overview is as follows. First, we focus on the following relation from the proxy's re-encryption stage:

$$C_B = C_A * rk_{A\to B} + p * e \bmod q.$$

Here, $C_A, C_B, p, q = 2^\gamma$ are known, where $\gamma$ is an integer, and $rk_{A\to B}, e$ are unknown. Our goal is to recover the re-encryption key $rk_{A\to B}$, and then obtain

**Table 1.** Number of ciphertexts needed

| | ees1087ep1 | ees1171ep1 | ees1499ep1 |
|---|---|---|---|
| PQCrypto 2019 | $4.06 \cdot 10^{17}$ | $4.83 \cdot 10^{17}$ | $9.67 \cdot 10^{17}$ |
| Our work | $3.17 \cdot 10^{3}$ | $3.58 \cdot 10^{3}$ | $4.45 \cdot 10^{3}$ |

the private key $f_A$ or $f_B$ based on $rk_{A\to B} = f_A * f_B^{-1} \bmod q$. For the sake of efficiency, we first choose to recover $rk_{A\to B} \bmod 2$ instead of $rk_{A\to B} \bmod q$. Due to the special structure of coefficients in the polynomial $e$, i.e., its coefficients have certain numbers of $+1$, $-1$, and 0. Hence, the inner product of the coefficient vector of $e$ is fixed. Thus, we can establish a system of linear congruence equations by using inner product calculation, and then obtain $rk_{A\to B} \bmod 2$ by using the linearization technique. According to $rk_{A\to B} = f_A * f_B^{-1} \bmod q$ and $q$ is a power of 2, we get that $f_B * (rk_{A\to B} \bmod 2) = f_A \bmod 2$. Without loss of generality, suppose that Bob is an attacker, where $f_B$ is the private key of Bob. Based on the above equation, Bob can determine the position of 0 bits of Alice's private key $f_A$. Notice that the private key pair $(f_A, g_A)$ of Alice satisfies $h_A = p*g_A*f_A^{-1} \bmod q$, where $h_A$ is the public key. It implies $f_A * h_A = p * g_A \bmod 2$. Furthermore, the attacker Bob can also deduce the position of 0 bits of $g_A$. Finally, combining with the position of 0 bits about $f_A$ and $g_A$, we get a new system of linear congruence equations derived from $f_A * h_A = p * g_A \bmod q$, and compute the remaining bits of $f_A$ and $g_A$ using Gaussian elimination. Theoretically, the algorithm overhead is divided into two main parts: constructing linear equations from the proxy's re-encryption stage and solving linear equations. Since we choose to work on $\mathbb{F}_2$ rather than $\mathbb{Z}_{2048}$, the cost of the latter is greatly reduced to negligible. This means that the time required to implement an attack is almost dependent on constructing a system of equations, which could be completed within one hour on PC.

Our another contribution is to discuss the NTRUReEncrypt instantiated with the NTRU parameter sets in the third round of NIST-PQC competition. Unlike the parameter sets from AsiaCCS 2015 and PQCrypto 2019, the parameter sets in the third round of NIST-PQC competition, e.g., NTRU-HPS and NTRU-HRSS [5], no longer determine the certain numbers of $+1$, $-1$, 0 in the coefficients of the secret polynomials. It means that the inner product of $e$ is not fixed. However, we can still take advantage of another property of ternary polynomials $e$. Denote the coefficient vector of $e$ as $\mathbf{e}$, hence each component $\mathbf{e}_i \in \{-1, 0, 1\}$ satisfies $\mathbf{e}_i = (\mathbf{e}_i)^3$. The remaining operations are the same as the previous attack, except that the number of communications is increased to $O(N^2)$.

**Organization.** The rest of this paper is organized as follows: In Sect. 2, we provide some basic preliminaries for the linear form and parameter sets of NTRU. In Sect. 3, we briefly describe NTRU and NTRUReEncrypt schemes, provide the specific parameter sets used in this paper. In Sect. 4, we present our attack in detail and give a comparison with PQCrypto 2019 [11]. In Sect. 5, we discuss the NTRUReEncrypt instantiated with the NTRU parameter sets in the third round

of NIST-PQC competition, and also compare with previous parameter sets used in Sect. 4. In Sect. 6, we present the experimental results, whose parameter sets are ees1087ep1, ees1171ep1, ees1499ep1 respectively. In Sect. 7, we conclude the paper.

## 2 Preliminaries

In this section, we provide some basic preliminaries of NTRU and NTRUReEncrypt scheme. The operations of both schemes are defined over the quotient ring $\mathcal{R} = \mathbb{Z}_q[x]/\left(x^N - 1\right)$, where $N$ is an odd prime. Other parameters $p$, $q$ are integers, where $p$ is much smaller than $q$ and $\gcd(p, q) = 1$.

The polynomials are selected from four subset of $\mathcal{R}$, denote as $\mathcal{L}_f = \mathcal{T}_{(d_f, d_f-1)}$, $\mathcal{L}_g = \mathcal{T}_{(d_g, d_g)}$, $\mathcal{L}_r = \mathcal{T}_{(d_r, d_r)}$,

$$\mathcal{L}_m = \left\{ m \in \mathcal{R} : \text{ every coefficient of } m \text{ lies between } -\frac{p-1}{2} \text{ and } \frac{p-1}{2} \right\}.$$

In addition, elements in $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_r$ are ternary polynomials. We introduce the definition of ternary polynomial from PQCrypto 2019 [11].

**Definition 1.** *A ternary polynomial $\mathcal{T}$ with positive integers $d_1$, $d_2$ is defined as:*

$$\mathcal{T}_{(d_1, d_2)} = \left\{ \begin{array}{c} \textit{trinary polynomials of } \mathcal{R} \textit{ with } d_1 \textit{ entries} \\ \textit{equal to } 1 \textit{ and } d_2 \textit{ entries equal to } -1 \end{array} \right\}.$$

### 2.1 Vector and Matrix Forms of NTRU

A polynomial $f \in \mathcal{R}$ in NTRU can be presented as $f = \sum_{i=0}^{N-1} f_i x^i$. Its vector form can be presented as $\mathbf{f} = (f_0, f_1, \cdots, f_{N-1})^T$. The polynomial $f$ can be written in the form of a circular matrix $\mathbf{F}$ in $\mathbb{Z}_q^{N \times N}$:

$$\mathbf{F} = \begin{pmatrix} f_0 & f_{N-1} & \cdots & f_1 \\ f_1 & f_0 & \cdots & f_2 \\ \vdots & \vdots & \ddots & \vdots \\ f_{N-1} & f_{N-2} & \cdots & f_0 \end{pmatrix}$$

Further, the matrix form of multiplication of two polynomials $f, g \in \mathcal{R}$ can be presented as:

$$\begin{pmatrix} f_0 & f_{N-1} & \cdots & f_1 \\ f_1 & f_0 & \cdots & f_2 \\ \vdots & \vdots & \ddots & \vdots \\ f_{N-1} & f_{N-2} & \cdots & f_0 \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_{N-1} \end{pmatrix}.$$

As needed, there are the following fundamental lemmas [9]:

**Lemma 1.** *If* $\mathbf{H} \in \mathbb{Z}_q^{N\times N}$ *is a circular matrix over* $\mathbb{Z}_q^{N\times N}$*, then* $\mathbf{H}^T$ *is also a circular matrix over* $\mathbb{Z}_q^{N\times N}$*.*

**Lemma 2.** *If* $\mathbf{G}, \mathbf{H} \in \mathbb{Z}_q^{N\times N}$ *are circular matrices, then* $\mathbf{GH}$ *is also a circular matrix. In particular,* $\mathbf{H}^T\mathbf{H}$ *is a symmetric circular matrix.*

## 3 NTRU and Its Proxy Re-encryption Scheme

In this section, we overview the NTRU and NTRU-based proxy re-encryption scheme, called NTRUReEncrypt. Parameters sets are shown in the following table, related to version 3.3 of the EESS#1 specification [2], from IEEE P1363.1 standard. For ees1087ep1, ees1171ep1, ees1499ep1, the private keys are $f, g$ selected from $\mathcal{L}_f = \mathcal{T}_{(d_f, d_f-1)}$ and $\mathcal{L}_g = \mathcal{T}_{(d_g, d_g)}$ respectively, the set of small polynomials is $\mathcal{L}_r = \mathcal{T}_{(d_r, d_r)}$, where small means that the coefficients of the polynomials are small.

**Table 2.** Instance of polynomial sets

| Instance | $N$ | $p$ | $q$ | $d_g$ | $d_f = d_r$ |
|---|---|---|---|---|---|
| ees1087ep1 | 1087 | 3 | 2048 | 362 | 120 |
| ees1171ep1 | 1171 | 3 | 2048 | 390 | 106 |
| ees1499ep1 | 1499 | 3 | 2048 | 499 | 79 |

In practice, some variants of NTRU take the following approach to generating key $f$ for efficiency: $f$ have the form of $1 + p * F$ with $F \in \mathcal{T}_{(d_f, d_f)}$, first generate $F \in \mathcal{T}_{(d_f, d_f)}$, and then calculate $1 + p * F$ to obtain $f$. We would use this form throughout the rest of the paper.

### 3.1 NTRU Cryptosystem

The brief description of NTRU cryptosystem is as follows, see [7] for more details.

- **Key Generation:** Randomly chooses $F \in \mathcal{T}_{(d_f, d_f)}$, then calculate $1 + p * F$ to obtain $f$, where $f$ has inverse $f_p^{-1}$, $f_q^{-1}$ in $R_p$, $R_q$, then randomly chooses $g \in \mathcal{T}_{(d_g, d_g)}$. Outputs public key $pk = h = p * g * f_q^{-1} \bmod q$, private key $sk = (f, g)$.
- **Encryption:** To encrypt a plaintext $m \in \mathcal{L}_m$, randomly chooses $r \in \mathcal{L}_r$. Outputs ciphertext $c = h * r + m \bmod q$.
- **Decryption:** To decrypt a ciphertext $c$, receiver uses private key $f$ and computes $a = f * c \bmod q$ such that coefficients of $a$ are all lie between $(-q/2, q/2]$. Outputs plaintext $m = a * f_p^{-1} \bmod p$.

Note that $f$, $g$, $s$, $m$ are small, i.e. each of its coefficients is small, then all coefficients of $a = c * f = p * g * s + m * f \bmod q$ lie in $(-q/2, q/2]$ with high probability. Thus, one computes $a = c * f \bmod q$ turns to $a = c * f$ over $\mathbb{Z}$. Then can decrypt the message:

$$a * f_p^{-1} = p * g * s * f_p^{-1} + m * f * f_p^{-1} = m \mod p.$$

### 3.2 NTRUReEncrypt

NTRUReEncrypt is a NTRU-based proxy re-encryption scheme, all parameter sets are related to formal NTRU scheme. Its initial key generation and first encryption stage are consistent with NTRU encryption, at second re-encryption stage, algorithm selects the same set of polynomials as NTRU. NTRUReEncrypt has a unique re-encrypt key generation, which ensures that Bob can decrypt the re-encrypted ciphertext sent from proxy.

The flow of the algorithm is as follows:

- **Key Generation:** Key generation algorithm is the same as that in NTRU. Outputs a pair of public and secret keys $(pk_A, sk_A)$ for Alice, where $sk_A = (f_A, g_A)$ and $pk_A = h_A$, and Bob also obtains a public-private key pair in the same way.
- **Re-encrypt Key Generation:** The algorithm requires two private keys $sk_A$ and $sk_B$, from sender Alice and receiver Bob respectively. Outputs re-encrypt key $rk_{A\to B} = f_A * f_B^{-1} \bmod q$. The re-encryption key can be computed by a simple three-party protocol below:
  1. Alice selects $t \in \mathcal{R}_q$, sends $t * f_A \bmod q$ to Bob and $t$ to proxy;
  2. Bob sends $t * f_A * f_B^{-1} \bmod q$ to proxy;
  3. Proxy computes $rk_{A\to B} = f_A * f_B^{-1} \bmod q$.
- **Encryption:** Alice encrypts a plaintext $m \in \mathcal{L}_m$, randomly chooses $r \in \mathcal{L}_r$. Outputs ciphertext $C_A = h_A * r + m \bmod q$.
- **Re-encryption:** Proxy encrypts a ciphertext $C_A$ sent by Alice, randomly chooses $e \in \mathcal{L}_r$. Outputs ciphertext $C_B = C_A * rk_{A\to B} + p * e \bmod q$.
- **Decryption:** Bob decrypts a ciphertext $C_B$, uses private key $f_B$ and compute

$$C_B * f_B = p * g_A * r + m * f_A + p * e * f_B \mod q$$

  such that coefficients of $C_B * f_B$ are all lie between $(-q/2, q/2]$. Outputs plaintext $m = C_B * f_B \bmod p$.

Decryption stage is similar to previous NTRU decryption, see [13] for more details.

## 4 Key Recovery Attack Against NTRUReEncrypt

In this section, we propose an efficient key recovery attack by only collecting ciphertexts $C_A$ and $C_B$ based on the algorithm of Li et al. [10]. They proposed a broadcast attack against NTRU only to recover messages at AsiaCCS 2015, however in NTRUReEncrypt, we find out that the re-encryption key $rk_{A\to B}$ can be recovered from the proxy's re-encryption stage, then a malicious receiver (sender) can directly recover the private key of the other one.

### 4.1 Construction of Equations

We now recall the re-encryption stage, proxy encrypts a ciphertext $C_A$ sent by Alice, randomly choose$e \in \mathcal{L}_r$. Outputs ciphertext

$$C_B = C_A * rk_{A\to B} + p * e \mod q. \tag{4.1}$$

For convenience, we denote $\mathbf{e}$, $\mathbf{c_B}$, $\lambda$ as their vector form in lowercase, and denote $\mathbf{C_A}$ as its matrix form in uppercase, then write Eq. (4.1) in linear form:

$$p\mathbf{e} = \mathbf{c_B} - \mathbf{C_A}\lambda \mod q,$$

where $\lambda$ is the vector form of re-encryption key $rk_{A\to B}$.

Then, do the inner product of both sides of the equation:

$$(p\mathbf{e})^T(p\mathbf{e}) = (\mathbf{c_B} - \mathbf{C_A}\lambda)^T(\mathbf{c_B} - \mathbf{C_A}\lambda) \mod q.$$

Note that $p = 3$ and secret polynomial $e$ selected in set $\mathcal{L}_r$, the numbers of $+1$ and $-1$ in their coefficients are $d_r$, thus $(p\mathbf{e})^T(p\mathbf{e}) = 2d_rp^2$ is a constant, denote as $d$.

We can get

$$d - \mathbf{c_B}^T\mathbf{c_B} = \lambda^T\mathbf{C_A}^T\mathbf{C_A}\lambda - 2\mathbf{c_B}^T\mathbf{C_A}\lambda \mod q. \tag{4.2}$$

### 4.2 Linearization

For convenience, let $d - \mathbf{c_B}^T\mathbf{c_B} = u$, $\mathbf{c_B}^T\mathbf{C_A} = (k_0, k_1, \cdots, k_{N-1})$, and

$$\mathbf{C_A}^T\mathbf{C_A} = \begin{pmatrix} c_0 & c_{N-1} & \cdots & c_1 \\ c_1 & c_0 & \cdots & c_2 \\ \vdots & \vdots & \ddots & \vdots \\ c_{N-1} & c_{N-2} & \cdots & c_0 \end{pmatrix}.$$

From **Lemma 2.2**, $\mathbf{C_A}^T\mathbf{C_A}$ is a symmetric circular matrix, where $c_i = c_{N-i}$, for $i \in \{0, 1, \cdots, N-1\}$. Then expanding Eq. (4.2), we can get

$$\begin{aligned} u = {} & c_0\left(\lambda_0^2 + \lambda_1^2 + \cdots + \lambda_{N-1}^2\right) \\ & + c_1\left(\lambda_1\lambda_0 + \lambda_2\lambda_1 + \cdots + \lambda_0\lambda_{N-1}\right) \\ & + \cdots\cdots \\ & + c_{N-1}\left(\lambda_{N-1}\lambda_0 + \lambda_0\lambda_1 + \cdots + \lambda_{N-2}\lambda_{N-1}\right) \\ & - 2k_0\lambda_0 - 2k_1\lambda_1 - \cdots - 2k_{N-1}\lambda_{N-1} \mod q \end{aligned} \tag{4.3}$$

Note that when choosing a specific parameter $N$, vector $\lambda = (\lambda_0, \lambda_1, \cdots, \lambda_{N-1})$ has $N$ unknown components. After the inner product operation, it generates $O(N^2)$ new monomials $\lambda_i\lambda_j$, for $0 \le i \le j \le N-1$.

A trivial idea is to linearize these variables to $O(N^2)$ one-order monomials, denoted as $\mathbf{x} = (x_0, x_1, \cdots, x_{O(N^2)-1})$. Then Eq. (4.3) turns to a congruence

equation with $O(N^2+N)$ variables, thus $\lambda_i$ can be recovered by collecting $O(N^2)$ equations in time $O(N^6)$ by Gaussian elimination. In certain parameter sets defined by NTRUReEncrypt, the size of $N$ generally amounts to $10^3$, which means the system of linear equations with around $10^6$ variables and it is hard to implement in practice.

To reduce the number of variables, let

$$x_i = \lambda_i\lambda_0 + \lambda_{i+1}\lambda_1 + \cdots + \lambda_{N-1}\lambda_{N-i-1} + \lambda_0\lambda_{N-i} + \cdots + \lambda_{i-1}\lambda_{N-1},$$

for $i = 0, 1, \cdots, N-1$. In the parameter sets we attacked, $N$ is an odd prime. Note that $c_i = c_{N-i}$ and $x_i = x_{N-i}$ for $i = 0, 1, \cdots, N-1$, the Eq. (4.3) is equivalent to

$$\begin{aligned} u = {} & c_0x_0 + 2c_1x_1 + \cdots + 2c_{[\frac{N}{2}]}x_{[\frac{N}{2}]} \\ & - 2k_0\lambda_0 - 2k_1\lambda_1 - \cdots - 2k_{N-1}\lambda_{N-1} \mod q, \end{aligned} \tag{4.4}$$

where $q$ is a power of 2 denoted as $q = 2^\gamma$, $\gamma$ is a positive integer. Further, assuming that $c_0, u$ are even, the equation could be converted to

$$\begin{aligned} \frac{1}{2}u = {} & \frac{1}{2}c_0x_0 + c_1x_1 + \cdots + c_{[\frac{N}{2}]}x_{[\frac{N}{2}]} \\ & - k_0\lambda_0 - k_1\lambda_1 - \cdots - k_{N-1}\lambda_{N-1} \mod 2^{\gamma-1}. \end{aligned} \tag{4.5}$$

Notice that we can get one congruence Eq. (4.4) with $(N + [\frac{N}{2}] + 1)$ variables by collecting $C_A$ and $C_B$ through one legal communication, so we could collect a series of samples by communicating relevant times. In fact through experiment, we could always select enough equations in the form of (4.5) by choosing these samples, and the number of samples is $O(N + [\frac{N}{2}])$, which is related to the number of variables.

### 4.3 Solving the System of Linear Congruence Equations

Denote $n$ as the number of variables and $n = N + \left[\frac{N}{2}\right] + 1$, then we build a linear system $\mathbf{L} \times \mathbf{X} = \mathbf{S} \bmod 2^{\gamma-1}$ by collecting $n + l$ equations from Eq. (4.5), where $l$ is a positive integer, the vector $\mathbf{X} = (x_0, x_1, \cdots, x_{[\frac{N}{2}]}, \lambda_0, \lambda_1, \cdots, \lambda_{N-1})^T$, the row of the matrix $\mathbf{L}$ corresponds to (4.5) equals

$$(\frac{1}{2}c_0, c_1, \cdots, c_{[\frac{N}{2}]}, -k_0, -k_1, \cdots, -k_{N-1})^T,$$

and $\mathbf{S}$ is the column vector related to $\frac{1}{2}u$. For the sake of efficiency, we choose to apply our algorithm to work over the finite field $\mathbb{F}_2$ but not the ring $\mathbb{Z}_{2^{\gamma-1}}$, which means that we turn to solve the system of equations $\mathbf{L} \times \mathbf{X} = \mathbf{S} \bmod 2$. That is, our goal is to find $rk_{A\to B} \bmod 2$ not $rk_{A\to B} \bmod 2^{\gamma-1}$, and we would show that it is enough for recovering the private key in the next subsection.

Note that the vector $\mathbf{S} \in \mathbb{F}_2^n$, the matrix $\mathbf{L} \in \mathbb{F}_2^{(n+l)\times n}$, we aim to find $rk_{A\to B} \bmod 2 = (\lambda_0, \lambda_1, \cdots, \lambda_{N-1})^T$ by selecting last $N$ bits of $\mathbf{X} \in \mathbb{F}_2^n$. It is

obvious that there is a unique solution is equivalent to the matrix $\mathbf{L}$ is invertible, which means that the rank of $\mathbf{L}$ equals to $n$. The problem turns to figure out the proportion of the matrices of rank $n$ in $\mathbf{L} \in \mathbb{F}_2^{(n+l)\times n}$. Li et al. [10] gave the following result estimating the proportion of invertible matrices in finite field among all matrices:

**Theorem 1.** *Let $\mathbb{F}_q$ be the finite field with $q$ elements, where $q$ is a prime power. The proportion of matrices of rank $n$ in the set of $(n+l)\times n$ matrices with entries in $\mathbb{F}_q$ is equal to:*

$$\prod_{k=l+1}^{n+l} \left(1 - q^{-k}\right), \quad l = 0, 1, 2, \cdots.$$

According to the theorem above, we give the proportion of the matrices of rank $n$ in $\mathbb{F}_q$ in Table 3 blow. It implies that if $l$ grows, the probability that the random matrix $\mathbf{L}$ is invertible is also increasing. In the case of our attack, $q = 2$, $l = 4$, and the random matrix $\mathbf{L}$ is invertible with high probability.

**Table 3.** The proportion of the matrices of rank $n$ in $\mathbf{L} \in \mathbb{F}_q^{(n+l)\times n}$

| $q$ | $l=0$ | $l=1$ | $l=2$ | $l=3$ | $l=4$ |
|---|---|---|---|---|---|
| 2 | 0.2889 | 0.5776 | 0.7701 | 0.8801 | 0.9388 |
| 3 | 0.5601 | 0.8402 | 0.9452 | 0.9816 | 0.9938 |
| 7 | 0.8368 | 0.9763 | 0.9966 | 0.9995 | 0.9999 |

For any ciphertext pair $(C_A, C_B)$ in Eq. (4.1), we could always get $C_B(1) = C_A(1)rk_{A\to B}(1) \bmod q$, which also holds on $\mathbb{F}_2$. Specifically, we could obtain a new equation:

$$C_B(1) = C_A(1)(\lambda_0 + \lambda_1 + \cdots + \lambda_{N-1}) \mod 2,$$

where $C_A(1), C_B(1)$ are fixed number. Adding this equation to the system of linear equations that we seek to solve, and now we can take $l = 3$ to implement our attack. Since the number of variables is $n = N + \left[\frac{N}{2}\right] + 1$, $l = 3$, thus we can construct a system of linear equations with the number of equations $n + l + 1 = N + \left[\frac{N}{2}\right] + 5$, which could be solved to obtain a unique solution in time $O(N^3)$ using Gaussian elimination. Compared to running on $\mathbb{Z}_{1024}$, our algorithm requires significantly less time to run on $\mathbb{F}_2$, just a few seconds.

### 4.4 Recovering Private Keys

In Sect. 4.3, we have obtained the re-encryption key $rk_{A\to B} \bmod 2$. Now, we discuss how to recover the private key in this subsection. First, we recover the position of 0 bits of the private key pair $(f, g)$ by means of the obtained

$rk_{A\to B} \bmod 2$, and then reveal the remaining bits of the private key $f$ by solving a system of linear equations.

Since $rk_{A\to B} = f_A * f_B^{-1} \bmod q$, we have that $rk_{A\to B} = f_A * f_B^{-1} \bmod 2$. If one party to the communication obtains $rk_{A\to B} \bmod 2$, then can immediately calculate the other party's private key in the sense of modulo 2. Now we design a roadmap to show how to recover the private keys. For the sake of description, we assume that Bob is the malicious party, who knows $rk_{A\to B} \bmod 2$ and the private key $f_B$:

**Step 1.** Considering $f_A = f_B * rk_{A\to B} \bmod 2$. Since $f_A = 1 + p * F$ with $F \in \mathcal{L}_{(d_f,d_f)}$ and $p = 3$, we get $p * F = f_B * rk_{A\to B} - 1 \bmod 2$. Note that there are $d_f$ $+1$'s, $d_f$ $-1$'s and $(N - 2d_f)$ 0's in the coefficients of $F$, so the position of 0 bits of $F$ can be determined by counting the position of the 0 bits of $f_B * rk_{A\to B} - 1 \bmod 2$, where the number of 0 bits of $F$ is $N - 2d_f$. It means that we can also get the position of the 0 bits of $f_A$.

**Step 2.** Since the public key $h_A = p * g_A * f_A^{-1} \bmod q$ with $g_A \in \mathcal{L}_{(d_g,d_g)}$ holds, $h_A = p * g_A * f_A^{-1} \bmod 2$ is also satisfied, where the coefficients of $g_A$ have $d_g$ $+1$'s and $d_g$ $-1$'s, $(N-2d_g)$ 0's. Based on $p * g_A = h_A * f_A \bmod 2$, the position of the 0 bits of $g_A$ can be determined by counting the position of 0 bits of $h_A * f_A \bmod 2$, where the number of 0 bits is $N - 2d_g$.

**Step 3.** Plugging $f_A = 1 + p * F$ into $h_A * f_A = p * g_A \bmod q$, we get $h_A * (1 + p * F) = p * g_A \bmod q$, which is equivalent to the equation

$$p * h_A * F = p * g_A - h_A \bmod q. \tag{4.6}$$

For convenience, we denote $\mathbf{f}$, $\mathbf{g}$, $\mathbf{h}$ as the vector form of $F$, $g_A$, $h_A$, and $\mathbf{H_A}$ as the matrix form of $h_A$. The Eq. (4.6) can be rewritten as the following linear form:

$$p\mathbf{H_A}\mathbf{f} = p\mathbf{g} - \mathbf{h} \mod q.$$

That is,

$$p \cdot \mathbf{H_A} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{pmatrix} = p \cdot \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_{N-1} \end{pmatrix} - \begin{pmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{pmatrix} \mod q, \tag{4.7}$$

where $\mathbf{H_A}$ is a $N \times N$ matrix. Considering the $2N$ variables ($\mathbf{f}$ and $\mathbf{g}$) of Eq. (4.7), there are $N - 2d_f$ and $N - 2d_g$ known in $\mathbf{f}$ and $\mathbf{g}$ respectively. Hence, the number of unknown variables is $2d_f + 2d_g$, whereas the number of equations is $N$. According to Table 2, $N$ is larger than $2d_f + 2d_g$ (e.g. in ees1171ep1, $N = 1171$, $d_g = 390$, $d_f = 106$, the number of equations $N = 1171$ is larger than the number of variables $2d_f + 2d_g = 992$). Hence we can recover the remaining bits of $\mathbf{f}$ by solving the system of linear equations using Gaussian elimination, then recover all bits of $f_A$ which is Alice's private key.

At PQCrypto 2019, Liu, Pan, and Zhang [11] proposed a key recovery attack based on statistical methods, malicious receiver Bob needed huge amount of ciphertexts $C_{B_i}$ encrypted by the same plaintext $m$, which is illegal and hard to implement. Here are the approximate number of ciphertexts in Table 4.

**Table 4.** Comparison of our work with PQCrypto 2019

| | ees1087ep1 | ees1171ep1 | ees1499ep1 |
|---|---|---|---|
| PQCrypto 2019 | $4.06 \cdot 10^{17}$ | $4.83 \cdot 10^{17}$ | $9.67 \cdot 10^{17}$ |
| Our work | $3.17 \cdot 10^{3}$ | $3.58 \cdot 10^{3}$ | $4.45 \cdot 10^{3}$ |

**Remark.** The cryptanalysis proposed by PQCrypto 2019 is based on decryption failure and statistical analysis, both require huge amount of ciphertexts and the chosen plaintexts. Moreover, the ciphertexts in latter case should be encrypted by the same plaintext. Unlike the previous ones, our attack has two advantages: (1) The amount of ciphertext required is greatly reduced. (2) There are no restrictions on plaintext, our attack only needs to be done in legal communication.

## 5 Case of NTRU Scheme with Different Parameter Sets

In this section, we discuss other schemes of NTRU with different parameter sets instantiated to the NTRUReEncrypt. We divide them into two cases, one with a constant number of $+1$, $-1$ (if any) coefficients of the secret polynomial selected in $\mathcal{L}_r$, in which case we can still attack with the same method as in the previous section, and the other with the NTRU schemes in the third round of NIST-PQC competition, which have a variable number of $+1$, $-1$ (if any) coefficients of the secret polynomial selected in $\mathcal{L}_r$, and we analyze this case by a new trick.

### 5.1 Case of Certain Secret Polynomial Coefficients

For the case of certain secret polynomial coefficients, [12] summarised some instantiations of NTRU, and their specific parameter sets are listed in the table below, where $\mathcal{B}$ denotes the set of all polynomials with binary coefficients, $\mathcal{B}(d)$ denotes a subset of $\mathcal{B}$ with exactly $d$ coefficients equal 1, $\mathcal{L}_m$ denotes the polynomial set whose coefficients lying between $-\frac{1}{2}(p-1)$ and $\frac{1}{2}(p-1)$ (Table 5).

One can check that, as for the secret polynomial $e$ selected from $\mathcal{L}_r$ in these schemes, the inner product of its coefficient vectors is a constant. Then we can use the method proposed in Sect. 4 to recover the private keys.

**Table 5.** Some instantiations of NTRU

| Parameter Sets | $q$ | $p$ | $\mathcal{L}_f$ | $\mathcal{L}_g$ | $\mathcal{L}_m$ | $\mathcal{L}_r$ |
|---|---|---|---|---|---|---|
| NTRU-1998 | $2^k \in \left[\frac{N}{2}, N\right]$ | 3 | $\mathcal{L}_{(d_f, d_f-1)}$ | $\mathcal{L}_{(d_g, d_g)}$ | $\mathcal{L}_m$ | $\mathcal{L}_{(d_r, d_r)}$ |
| NTRU-2001 | $2^k \in \left[\frac{N}{2}, N\right]$ | $x+2$ | $1+p*F$ | $\mathcal{B}(d_g)$ | $\mathcal{B}$ | $\mathcal{B}(d_r)$ |
| NTRU-2005 | prime | 2 | $1+p*F$ | $\mathcal{B}(d_g)$ | $\mathcal{B}$ | $\mathcal{B}(d_r)$ |

### 5.2 Case of Uncertain Secret Polynomial Coefficients

We now discuss the case in the third round of NIST-PQC competition, such as NTRU-HPS, NTRU-HRSS [5], whose parameter sets are instantiated to the NTRUReEncrypt. For specific parameter sets in ees1087ep1, ees1171ep1, ees1499-ep1, our attack's point is that the secret polynomial $e$ selected in set $\mathcal{L}_r$, whose coefficients have a certain number of $+1$, $-1$, and 0.

However, in NTRU-HPS and NTRU-HRSS, polynomial set $\mathcal{L}_r = \mathcal{T}$ and $\mathcal{T}$ is the set of non-zero ternary polynomials of degree at most $N-2$. It indicates that we no longer have information on the number of coefficients in the secret polynomial $e$, thus the inner product calculation would fail. Ding et al. [6] used the property $\mathbf{e}_i = {\mathbf{e}_i}^3$, for $i \in \{0, 1, \cdots, N-1\}$ in the broadcast attack against NTRU to recover plantexts, it could also be used in this case to recover the secret keys.

Separating $p$ from Eq. (4.1) and write it in linear form, we can get

$$\mathbf{e} = (\mathbf{C_B} - \mathbf{C_A}\mathbf{r}) * p^{-1} \mod q.$$

Since $\mathbf{e}_i = {\mathbf{e}_i}^3$, so we can get equations that eliminates $\mathbf{e}$:

$$[(\mathbf{C_B} - \mathbf{C_A}\mathbf{r}) * p^{-1}]_i = [(\mathbf{C_B} - \mathbf{C_A}\mathbf{r}) * p^{-1}]_i^3 \mod q, \tag{5.1}$$

for $i \in \{0, 1, \cdots, N-1\}$. Note that in Eq. (5.1) only $\mathbf{r}$ is the unknown variable, cubic computation generates $O(N^3)$ new monomials, and we can also linearize these monomials into new variables. Since one legal communication produces $N$ equations, the system of linear congruence equations can be built by communicating $N^2$ times, thus recover $\mathbf{r}$ in time $O(N^9)$. The following table is the comparison of parameter sets between EESS#1 and NTRU-Round3, where NTRU-HPS is the same as NTRU-HRSS (Table 6).

**Table 6.** Comparison of EESS#1 with NTRU-Round3

| Instance | Number of communications | Variables | Gaussian elimination |
|---|---|---|---|
| EESS#1 | $O(N)$ | $O(N)$ | $O(N^3)$ |
| NTRU-HPS | $O(N^2)$ | $O(N^3)$ | $O(N^9)$ |

## 6 Experiments

In this section, we present experimental results on the assumption that Bob is a malicious receiver. Due to ciphertexts $C_A$ could be collected on the public channel and $C_B$ could be received normally by Bob, we assumed in our experiment that the attacker could collect enough ciphertext pairs $(C_A, C_B)$. All experiments were performed in SageMath 9.6 on a macOS Monterey 12.5.1 system with Apple M1 CPU @ 3.2GHz, 8GB RAM, and our implement was available at https://github.com/s4lTea/NTRUReEncrypt_Attack. We implemented

our attack against NTRUReEncrypt scheme, whose parameter sets defined by EESS#1 are the same as those from AsiaCCS 2015 [13] and PQCrypto 2019 [11]. We performed our attack 50 times for each instance, and gave the average number of communications and running time required by the algorithm. In our experimental results, let $n = N + \left[\frac{N}{2}\right] + 1$, we could always find a matrix $\mathbf{L}$ of rank $n$. We splited the algorithm into 3 steps:

1) Focusing on the proxy's re-encryption stage, then generate a system of linear congruence equations with $n+4$ equations and $n$ variables by communicating enough times.
2) Solving it on $\mathbb{F}_2$ using Gaussian elimination to obtain re-encryption key $rk_{A\to B} \bmod 2$.
3) Building another system of linear congruence equations with $N$ equations and $2d_f + 2d_g$ variables to solve, finally obtain Alice's private key.

**Table 7.** Experimental Results with different parameter sets

| Instance | $N$ | $p$ | $q$ | Rank($\mathbf{L}$) | Number of communications | Total time(min) |
|---|---|---|---|---|---|---|
| ees1087ep1 | 1087 | 3 | 2048 | 1634 | 3174 | 17.4 |
| ees1171ep1 | 1171 | 3 | 2048 | 1757 | 3579 | 22.8 |
| ees1499ep1 | 1499 | 3 | 2048 | 2249 | 4454 | 41.9 |

Step 1 takes some time (minutes) due to matrix multiplication operations. As it works on $\mathbb{F}_2$, so step 2 takes only a few seconds and the running time could be negligible. There are small number of variables related to the equations in step 3, so the time required to either construct or solve the equations is negligible. The experimental results are shown in Table 7. For ease of description, we take the cost of step 1 as the total time of our algorithm.

## 7 Conclusion

In this paper, we presented an efficient key recovery attack against NTRUReEncrypt scheme, whose parameter sets are defined by EESS#1 specification [2] from IEEE P1363.1 standard. The attack is based on a special structure of secret polynomials from the set $\mathcal{L}_r$. In addition, the key recovery attack could be extended to the NTRUReEncrypt instantiated with the NTRU parameter sets in the third round of NIST-PQC competition.

**Acknowledgments.** The authors would like to thank anonymous reviewers for their helpful comments and suggestions. The work of this paper was supported in part by the National Natural Science Foundation of China (Grants 61732021, 62272454).

## References


1. Aono, Y., Boyen, X., Phong, L.T., Wang, L.: Key-private proxy re-encryption under LWE. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 1–18. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03515-4_1
2. Key Cryptographic Techniques Based. IEEE p1363. 1$^{TM}$/d1211 (2008)
3. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054122
4. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 185–194 (2007)
5. Cong, C., Oussama, D., Jerey, H.: NTRU: the round 3 NIST submission (2020). https://ntru.org/release/NIST-PQ-Submission-NTRU-20201016
6. Ding, J., Pan, Y., Deng, Y.: An algebraic broadcast attack against NTRU. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 124–137. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31448-3_10
7. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054868
8. Howgrave-Graham, N., et al.: The impact of decryption failures on the security of NTRU encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 226–246. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_14
9. Kra, I., Simanca, S.R.: On circulant matrices. Notices AMS **59**(3), 368–377 (2012)
10. Li, J., Pan, Y., Liu, M., Zhu, G.: An efficient broadcast attack against NTRU. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, pp. 22–23 (2012)
11. Liu, Z., Pan, Y., Zhang, Z.: Cryptanalysis of an NTRU-based proxy encryption scheme from ASIACCS'15. In: Ding, J., Steinwandt, R. (eds.) PQCrypto 2019. LNCS, vol. 11505, pp. 153–166. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25510-7_9
12. Mol, P., Yung, M.: Recovering NTRU secret key from inversion oracles. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 18–36. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_2
13. Nuñez, D., Agudo, I., Lopez, J.: NTRUReEncrypt: an efficient proxy re-encryption scheme based on NTRU. In Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, pp. 179–189 (2015)
14. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. IEEE (1994)
15. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)