Cryptanalysis of an NTRU-Based Proxy Encryption Scheme from ASIACCS'15

Zhen Liu[1,2,3], Yanbin Pan[1], and Zhenfei Zhang[4](✉)

[1] Key Laboratory of Mathematics Mechanization, NCMIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
panyanbin@amss.ac.cn

[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

[3] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China
liuzhen16@mails.ucas.ac.cn

[4] Algorand, Boston, USA
zhenfei@algorand.com

Abstract. In ASIACCS 2015, Nuñez, Agudo, and Lopez proposed a proxy re-encryption scheme, `NTRUReEncrypt`, based on NTRU, which allows a proxy to translate ciphertext under the delegator's public key into a re-encrypted ciphertext that can be decrypted correctly by delegatee's private key. In addition to its potential resistance to quantum algorithm, the scheme was also considered to be efficient. However, in this paper we point out that the re-encryption process will increase the decryption error, and the increased decryption error will lead to a reaction attack that enables the proxy to recover the private key of the delegator and the delegatee. Moreover, we also propose a second attack which enables the delegatee to recover the private key of the delegator when he collects enough re-encrypted ciphertexts from a same message. We reevaluate the security of `NTRUReEncrypt`, and also give suggestions and discussions on potential mitigation methods.

Keywords: `NTRUReEncrypt` · NTRU · Decryption failure · Reaction attack · Key recovery

1 Introduction

The concept of proxy re-encryption (PRE) scheme was proposed by Blaze, Bleumer and Strauss in 1998 [5]. A re-encryption scheme allows a proxy to translate ciphertext under the delegator's public key into a ciphertext of the same message that can be decrypted correctly by the delegatee's private key, whereas the proxy is given just a re-encryption key and learns nothing about

Yanbin Pan was supported by the NNSF of China (No. 61572490), and by the National Center for Mathematics and Interdisciplinary Sciences, CAS.

the message. A PRE scheme can be seen as an extension of public-key encryption. It uses same fundamental algorithms as a traditional public key encryption scheme. Additionally, it also requires algorithms to generate re-encryption keys and to re-encrypt ciphertexts.

In the literature, there exit a number of proxy re-encryption schemes, based on number theoretic problems such as the discrete logarithm problem [6]. However, due to Shor's quantum algorithm, the integer factorization problem and the discrete logarithm problem can be solved efficiently [17,18]. It is crucial to have alternatives that are robust against quantum computers. In 2017, NIST [1,7] started a standardization process on post-quantum cryptography. Among all candidate proposals, lattice based solutions are ones of most promising. Although NIST considers only public key encryption and signature schemes at this stage, it is also important to identify lattice based candidate for proxy re-encryption schemes, for examples [3,20].

At AsiaCCS 2015, Nuñez, Agudo and Lopez [16] proposed a new proxy re-encryption scheme, `NTRUReEncrypt`, based on a well-established lattice-based public-key encryption scheme NTRU. Here the encryption and decryption messages are identical to the classical `NTRUEncrypt` scheme. With an additional re-encryption mechanism, they achieved an efficient post-quantum PRE scheme.

NTRU [12], introduced by Hoffstein, Pipher and Silverman in 1996, has been standardized by IEEE 1363.1 [19] and ANSI X9.98 [2]. It features high efficiency and low memory requirement. After 20 years of development, there are three mainstreams of the NTRU algorithms. The IEEE standardized version, `NTRUEncrypt` was later on submitted to NIST-PQC process as [21]. The parameters follow the design principals outlined in [11]. The other two NTRU based submissions are NTRU-prime [4] and NTRU-HRSS [15] schemes.

Similar to other lattice based cryptosystems, the NTRU scheme may admit decryption errors. When a decryption failure occurs, information on private keys may be (partially) leaked. In 2003, Howgrave *et al.* [14] successfully demonstrated an attack that employs large number of queries to a *weak* decryption oracle. Unlike a classical decryption oracle, a weak decryption oracle will only tell whether a valid ciphertext was decrypted correctly or not (see [13]). This attack is later known as the *reaction* attack, and becomes common to lattice based cryptography [8,22]. In practice, to address this attack one may choose optimized parameters so that the decryption error is negligible in security parameter, for example, `NTRUEncrypt` [11]; or less optimized ones that eliminate the decryption errors, for example, NTRU-HRSS [15].

In `NTRUReEncrypt`, the delegator first chooses a small polynomial s, and encrypts the message m as $C_A = h_A * s + m$, where delegator's private key is (f_A, g_A) and delegator's public key is $h_A = p * g_A * f_A^{-1}$. After receiving C_A, the proxy chooses small polynomial e and sends $C_B = C_A * rk_{A\to B} + p * e$ to the delegatee, where $rk_{A\to B} = f_A * f_B^{-1}$ is the re-encrypted key of the proxy and f_B is the private key of the delegatee. Finally, the delegatee computes $C_B * f_B$ modulo q and reduces it modulo p to recover the message m.

Our first contribution is to analyze the `NTRUReEncrypt` scheme using the aforementioned reaction attack. Note that the `NTRUReEncrypt` scheme follows the parameter sets in [11,19,21]. The probability of decryption failure was set to be negligible in the security parameter for a public key encryption scheme. However, the re-encryption process of `NTRUReEncrypt` significantly increases decryption error[1]. We give a detailed analyze the probability of decryption failure in Table 1, and show how to use a reaction attack to recover private keys, given sufficient many decryption failures. We also note that one can simply mitigate this attack by increasing the modulus (and also the dimensions accordingly to ensure the lattice problem is still hard in practice) so that decryption failure probability becomes negligible again.

Table 1. The probabilities of decryption failure after encryption and re-encryption

Parameter sets	$log_2(P_{dec}(c))$	$log_2(P_{dec}(c'))$
ees1087ep1	-219	-92
ees1171ep1	-245	-117
ees1499ep1	-323	-200
ntru-443	-217	-35
ntru-743	-122	-16

Our other contribution is a new attack in which a curious delegatee receiving a large re-encrypted ciphertexts from a single message can recover the private key of a delegator. Roughly speaking, note that the intermediate polynomial during the delegatee's decryption has the form of $C_{B_i} * f_B = p * g_A * s_i + m * f_A + p * e_i * f_B$. Once the delegatee collects enough (denoted by l) intermediate polynomials for a same message m, he can average them to obtain $p * g_A * \sum_{i=1}^{l} s_i/l + m * f_A + p * f_B * \sum_{i=1}^{l} e_i/l = f_B * \sum_{i=1}^{l} C_{B_i}/l$. Since s_i, e_i are randomly chosen small polynomials, for sufficiently large l, the coefficients of $p * g_A * \sum_{i=1}^{l} s_i/l$ and $p * f_B * \sum_{i=1}^{l} e_i/l$ will be very small. Hence, with overwhelming probability, the equation $m * f_A = Round(\sum_{i=1}^{l} f_B * C_{B_i}/l)$ holds, from which we can efficiently recover the private key f_A. To resist such an attack, some randomized padding scheme should be added carefully (Table 2).

Our second attack indeed bases on the fact that each re-encrypted messages leaks partial information of the secret key. Our attack is a simple illustration of such a leakage. In lattice based signatures schemes, transcript leakages are usually fixed with rejection sampling methods. It is not trivial to apply this method to an re-encryption scheme. We leave secure instantiation of NTRU based re-encryption schemes to future work.

[1] Indeed, even if the `NTRUReEncrypt` adopts NTRU-HRSS parameter sets that don't have decryption errors by design, the re-encryption process will introduce decryption errors.

Table 2. The approximate number of required re-encrypted ciphertexts

Parameter sets	Number of ciphertexts
ees1087ep1	$2^{58.5}$
ees1171ep1	$2^{58.7}$
ees1499ep1	$2^{59.7}$
ntru-443	$2^{53.5}$
ntru-743	$2^{57.3}$

Roadmap. The remainder of the paper is organized as follows. In Sect. 2, we recall the original NTRU encryption and explain its decryption failures. In Sect. 3, we present the proxy re-encryption scheme `NTRUReEncrypt`. In Sect. 4, we give our first attack against `NTRUReEncrypt`and analyze the decryption failure probability. In Sect. 5, we give our second attack against `NTRUReEncrypt`. Finally, we give a short conclusion in Sect. 6.

2 Notations and Preliminaries

2.1 Notations and Definitions

Let $\mathcal{R}$ denote the ring $Z[X]/(X^N - 1)$, where N is prime. Let $+$ and $*$ denote addition and multiplication in $\mathcal{R}$, respectively. For integer p, q, $\gcd(p, q) = 1$ and $p \ll q$. Let $\mathcal{R}_q$ be the ring $Z_q[X]/(X^N - 1)$ and $\mathcal{R}_p$ be the ring $Z_p[X]/(X^N - 1)$. We use $\|.\|_\infty$ to denote the infinite norm and $\|.\|$ to denote the Euclidean norm.

A polynomial $a(x) = a_0 + a_1 x + \cdots + a_{N-1} x^{N-1}$ is identified with its vector of coefficients $a = [a_0, a_1, \cdots, a_{N-1}]$. The maximum and minimum coefficients of polynomial or vector are denoted by

$$Max(a(x)) = \max_{0 \le i \le N-1} \{a_i\} \quad and \quad Min(a(x)) = \min_{0 \le i \le N-1} \{a_i\}.$$

The width of a polynomial $a(X)$ is the difference between its largest and smallest coefficients

$$Width(a(x)) = Max(a(x)) - Min(a(x)).$$

The reversal polynomial $\bar{a}(x)$ of a polynomial $a(x)$ in $\mathcal{R}$ is defined to be $\bar{a}(x) = a(x^{-1})$. If $a = (a_0, a_1, \cdots, a_{N-1})$, then $\bar{a} = (a_0, a_{N-1}, a_{N-2}, \cdots, a_1)$.

Let $\widehat{a}(x) = a(x) * \bar{a}(x)$ in $\mathcal{R}$, a coefficient $\widehat{a}_i$ of $\widehat{a}(x)$ is the dot products of a with its successive rotations $x^i * a$. We have $\widehat{a}_0 = \sum_{i=0}^{N-1} a_i^2 = \|a\|^2$.

For positive integers d_1, d_2, We set the notation:

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

2.2 Overview of NTRU

We now briefly present the basic NTRU encryption scheme, for more details see [12]. The polynomials used in NTRU are selected from four sets $\mathcal{L}_f$,$\mathcal{L}_g$,$\mathcal{L}_s$,$\mathcal{L}_m$, where $\mathcal{L}_f = \{f : f \in \mathcal{T}_{(d_f, d_f - 1)}\}$, $\mathcal{L}_g = \{g : g \in \mathcal{T}_{(d_g, d_g)}\}$, $\mathcal{L}_s = \{s : s \in \mathcal{T}_{(d_s, d_s)}\}$, and $\mathcal{L}_m$=$\{m \in \mathcal{R}$: every coefficient of m lies between $\frac{p-1}{2}$ and $\frac{p-1}{2}\}$.

- **KeyGen(1^k):** On input security parameter k, the key generation algorithm KenGen first chooses $f \in \mathcal{L}_f$, such that f has inverse f_q^{-1} in R_q and f_p^{-1} in R_p, $g \in \mathcal{L}_g$, then computes $h = p * g * f_q^{-1} \mod q$ and outputs public key $pk = h$ and private key $sk = (f, g)$.

- **Enc(pk, m):** On input the public key pk and a message $m \in \mathcal{L}_m$, the encryption algorithm Enc chooses $s \in \mathcal{L}_s$ and outputs the ciphertext $c = h * s + m \mod q$.

- **Dec(sk, c):** On input the private key sk and the ciphertext c, the decryption algorithm Dec computes $a = c * f \mod q$, and place the coefficient of a in the interval $(-q/2, q/2]$. Outputs $m = a * f_p^{-1} \mod p$.

2.3 Decryption Failures

When decrypting a ciphertext c, one caluates

$$a = c * f = p * g * s + m * f \mod q. \tag{1}$$

Since the polynomials f, g, s and m are small, the coefficients of polynomial $p * g * s + m * f$ lie in $(-q/2, q/2]$ with high probability. If the equality mod q in Eq. (1) also holds over $\mathbb{Z}$. Then, we have

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

Hence decryption works if Eq. (1) also holds over $\mathbb{Z}$. A warp failure occurs if $\|p * g * s + m * f\|_\infty \geq q/2$ and a gap failure occurs if the width of $p*g*s+m*f$ is greater than or equal to q.

Howgrave *et al.* [14] presented the attack based on decryption failure. The attacker selected (m, s_i) with fixed m, such that $\|p * g * s_i + m * f\|_\infty \geq q/2$. Once the attacker collected sufficiently large (m, s_i), the attacker can recover the private key (g, f).

3 NTRUReEncrypt

3.1 Presentation of the Scheme

In [16], Nuñez *et al.* proposed a proxy re-encryption scheme NTRUReEncrypt based on NTRU, where a proxy is given re-encryption key $rk_{A \to B}$ that allows him to translate a message m encrypted under Alice's public key pk_A into a re-encrypted ciphertext of the same message m decryptable by Bob's private key sk_B.

The `NTRUReEncrypt` scheme consists of five algorithms:

- **KeyGen(1^k):** On input the security parameter k, the output of key generation algorithm for Alice is (sk_A, pk_A), where $sk_A = (f_A, g_A)$ and $pk_A = h_A$. Let f_A^{-1} denote the inverse of f_A in the ring R_q.

- **ReKeyGen(sk_A, sk_B):** On input the secret key sk_A and the secret key sk_B, the re-encryption algorithm ReKeyGen computes the re-encryption key between Alice and Bob as $rk_{A\to B} = f_A * f_B^{-1} \mod q$. The re-encryption key can be computed by a simple three-party protocol originally proposed in [6], is as follows: Alice selects $r \in \mathcal{R}_q$ and sends $r * f_A \mod q$ to Bob and r to the proxy, then Bob sends $r * f_A * f_B^{-1} \mod q$ to the proxy, so the proxy can compute $rk_{A\to B} = f_A * f_B^{-1} \mod q$.

- **Enc(pk_A, m):** On input the public key pk_A and the message m, the encryption algorithm Enc generates $s \in \mathcal{T}_{d_s,d_s}$, and outputs $C_A = h_A * s + m \mod q$.

- **ReEnc($rk_{A\to B}, C_A$):** On input a re-encryption key $rk_{A\to B}$ and a ciphertext C_A, the re-encryption algorithm ReEnc generates $e \in \mathcal{T}_{d_s,d_s}$ and outputs $C_B = C_A * rk_{A\to B} + p * e \mod q$.

- **Dec(sk_A, C_A):** On input the secret key sk_A and the ciphertext C_A, the decryption algorithm computes ${C_A}' = C_A * f_A \mod q$ and outputs $m = {C_A}' \mod p$.

Next, We would like to point out that

- In order to decrypt the re-encrypted ciphertext correctly, the private polynomial f_B has to be congruent to 1 modulo p. So the difference between NTRU and `NTRUReEncrypt` of the key generation is that the private key f has the form of $1 + p * F$, where $F \in \mathcal{T}_{(d_f,d_f)}$.
- In practical, the message m is padded with random bits and masked according to a hamming weight restriction, which means message representatives are trinary polynomials with the number of $+1s$, $-1s$, and $0s$ each be greater than d_m. So for simplicity, m satisfies the hamming weight restriction in this paper.
- The error term e is chosen randomly from the ring $\mathcal{R}$ during the re-encryption in [16], which is unreasonable. In fact, e should be small, we therefore assume that e is sampled from the same set as s.

For the correctness of Bob's decryption, when Bob gets the re-encryption ciphertext C_B, he first computes

$$\begin{aligned} C_B * f_B &= (C_A * f_A * f_B^{-1} + p * e) * f_B \\ &= (h_A * s + m) * f_A + p * e * f_B \\ &= p * g_A * s + m * f_A + p * e * f_B \mod q. \end{aligned} \tag{2}$$

If the last part of Eq. (2) also holds over $\mathbb{Z}$, then we have $C'_B = p * g_A * s + m * f_A + p * e * f_B$. After taking modulo p, Bob can obtain the original message m.

Remark 1. *The scheme is also bidirectional and multihop, namely it's trivial to obtain $rk_{B\rightarrow A}$ from $rk_{A\rightarrow B}$ and the re-encryption process can be repeated multiple times.*

3.2 Parameter Sets

The author of [16] implemented `NTRUReEncrypt` scheme on ees439ep1, ees1087ep1, ees1171ep1, ees1499ep1 parameter sets following the IEEE P1363.1 standards [19]. They also used the product form polynomials for optimization of each set. However, some specific parameters are not clear in [16], so we only list ees1087ep1, ees1171ep1, ees1499ep1 in Table 3.

Note that the NTRU project has proposed new parameter sets ntru-443 and ntru-743, which are submitted to NIST PQC competition [21]. For completeness, we also list them in Table 3 to analyze the security of the scheme.

For ees1087ep1, ees1171ep1, ees1499ep1, ntru-443, ntru-743, the private key is $(f, g) = (1+p*F)$ with $F \in \mathcal{T}_{(d_f,d_f)}$ and $g \in \mathcal{T}_{(dg,dg)}$, the polynomial $s \in \mathcal{T}_{(d_s,d_s)}$.

Table 3. Some instances of trinary polynomials

Instance	N	p	q	dg	df=ds=dm
ees1087ep1	1087	3	2048	362	120
ees1171ep1	1171	3	2048	390	106
ees1499ep1	1499	3	2048	499	79
ntru-443	443	3	2048	143	143
ntru-743	743	3	2048	247	247

4 Reaction Attack Against NTRUReEncrypt

Recall that in Bob's decryption, the intermediate polynomial is $p * g_A * s + m * f_A + p * e * f_B$ and the additional term $p * e * f_B$ produces an increased error. Hence, the decryption failure probability is expected to significantly increase. On the other hand, the attacks based on the decryption failures has been studied well in [14]. Therefore, we employ their attack to analyze the security of the `NTRUReEncrypt` scheme.

More precisely, it is assume that the attacker has access to an oracle to determine whether a validly created ciphertext can be decrypted correctly or not. The attack takes as follows. The first stage is that the attacker uses the oracle to collect (m, s, e), which generates the re-encrypted ciphertext C_B that can not be decrypted correctly. The second stage is that the attacker fixes (m, s) and randomly searches e_i, where (m, s, e_i) causes decryption failure. The final stage is that the attacker uses those $\widehat{e_i}$ correlated with $\widehat{f_B}$ to determine the private key f_B. Note that the proxy can create C_{A_i} by encrypting random m_i with random s_i and C_{B_i} by re-encrypting C_{A_i} with random e_i, we therefore assume that a corrupt proxy can act as an attacker.

Before explaining our attack, we first show that the decryption failure probability of `NTRUReEncrypt` significantly increases for Bob.

4.1 Estimating Decryption Failure Probability of C_A

We use the method introduced in [10] to estimate the decryption failure probability. Recall that in Alice's decryption, she computes

$$\begin{aligned} {C_A}' &= p * g_A * s + f_A * m \\ &= p * g_A * s + p * F_A * m + m \mod q, \end{aligned}$$

Decryption works, if

$$\left\|{C_A}'\right\|_\infty = \|p * (g_A * s + F_A * m) + m\|_\infty < q/2.$$

Therefore, the decryption failure probability P_{dec} can be bounded by the probability that one or more coefficients of $g_A * s + F_A * m$ has an absolute value greater than $c = (q-2)/(2p)$. So we have

$$P_{dec}(c) = \Pr\left[\|g_A * s + F_A * m\|_\infty \geq c\right].$$

For trinary $F_A \in \mathcal{T}_{(d_f,d_f)}$, $g_A \in \mathcal{T}_{(d_g,d_g)}$, $s \in \mathcal{T}_{(d_s,d_s)}$. Let X_j denote a cofficient of $g_A * s + F_A * m$, then X_j has the form

$$(g_A * s + F_A * m)_j = (s * g_A)_j + (F_A * m)_j,$$

and each term in the sum is a sum of either $2d_s$ or $2d_f$ coefficients of g_A or m. Note that each term in the sum has mean 0.

For instance, let $\varepsilon(i) \in \{1, -1\}$ and $a(i)$ represents index, we have

$$(s * g_A)_j = \sum_{i=1}^{2d_s} \varepsilon(i)(g_A)_{a(i)}.$$

We assume that the coefficients of g_A are independent random variables taking the value 1 with probability $\frac{d_g}{N}$, -1 with probability $\frac{d_g}{N}$ and 0 with probability $\frac{N-2d_g}{N}$. Hence, the variance σ_1^2 of $(g_A * s)_j$ is computed as:

$$\sigma_1^2 = E((s * g_A)_j^2) = \sum_{i=1}^{2d_s} E((g_A)_{a(i)}^2) = \frac{4d_sd_g}{N}.$$

Recall that the message m is sampled uniformly from the set of trinary polynomials, which restrains that the number of non-zero coefficients can not exceed $N - d_m$. We also assume that the coefficient of m is chosen as ± 1 with the probability $\frac{N-d_m}{N}$ and 0 with the probability $\frac{d_m}{N}$. Similarly, the variance σ_2^2 of $(F_A * m)_j$ is

$$\sigma_2^2 = E((F_A * m)_j^2) = 2d_f \cdot \frac{N-d_m}{N}.$$

Suppose $2d_s$, $2d_g$ are large, the central limit theorem suggests that the distribution of X_j has the normal distribution with mean 0 and variance σ^2:

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 = \frac{4d_s d_g + 2d_f \cdot (N - d_m)}{N}$$

With complementary error function $\mathbf{erfc}(\cdot)$, the probability that a coefficient X_j has absolute value exceeds c is given by

$$\Pr[|X_j| \geq c] = \mathbf{erfc}(c/\sqrt{2}\sigma).$$

After applying the union bound, the probability $P_{dec}(c)$ is bounded by

$$P_{dec}(c) = N \cdot \mathbf{erfc}(c/\sqrt{2}\sigma),$$

where

$$\mathbf{erfc}(c/\sqrt{2}\sigma) = \frac{2}{\sqrt{\pi}} \cdot \int_{c/\sqrt{2}\sigma}^{\infty} e^{-x^2}\,dx.$$

4.2 Estimating Decryption Failure Probability of C_B

When Bob receives the re-encrypt ciphertext C_B, the intermediate process is to compute

$$C_B{}' = p * g_A * s + m * f_A + p * e * f_B,$$

and the failure occurs if

$$\|p \cdot (g_A * s + F_A * m + p * F_B * e) + pe + m\|_\infty \geq q/2.$$

Similarly, for trinary $F_B \in \mathcal{T}_{(d_f,d_f)}$, we get the probability

$$P_{dec}(c') = N \cdot \mathbf{erfc}(c'/\sqrt{2}\sigma'),$$

where

$$\sigma'^2 = \sigma^2 + \frac{p^2 \cdot 4d_s d_f}{N},$$

and

$$c' = c - 1.$$

We estimate decryption failure probabilities with the parameters specified in Sect. 3.2 and list them below (Table 4).

As we can see, the probability that the re-encrypted ciphertext C_B fails to decrypt is much greater than that of C_A. What's more, the decryption failures lead to reaction attack.

Table 4. The probabilities of decryption failure during encryption and re-encryption

Instance	σ^2	σ'^2	$log_2(P_{dec}(c))$	$log_2(P_{dec}(c'))$
ees1087ep1	373	850	-219	-92
ees1171ep1	334	679	-245	-117
ees1499ep1	255	405	-323	-200
ntru-443	378	2040	-217	-35
ntru-743	658	3614	-122	-16

4.3 Description of the Attack

For completeness, we simply describe the attack as below. See [14] for more details about the reaction attack based on the decryption failure.

- **Stage 1:** The attacker first collects (m, s, e), which will generate the re-encrypted ciphertext C_B that can not be decrypted correctly. Moreover, the triplet (m, s, e) should satisfy two conditions: there must be a coefficient of $p * g_A * s + m * f_A$ that is both abnormally far from its expected value and further from the expected value than any other coefficient, and the distances between the two coefficients of $p * g_A * s + m * f_A$ furthest from their expected value, which is known as the gap of $p * g_A * s + m * f_A$, should be large enough.

- **Stage 2:** For fixed (m, s) found in Stage 1, the attacker randomly chooses e_i and collects (m, s, e_i) that causes decryption failure for Bob. Suppose the $i-$th coefficient of $p * g_A * s + m * f_A$ is abnormally far from its expected value, then it is most likely that the absolute value of the $i-$th coefficient of $p * g_A * s + m * f_A + p * e_i * f_B$ exceeds $q/2$. The strength of this bias towards the $i-$th coefficient of the $p * g_A * s + m * f_A + p * e_i * f_B$ will depend on the gap of $p * g_A * s + m * f_A$. What's more, it suggests that e_i is correlated with $x^i * \overline{f_B}$. Since the reversal of $x^i * \overline{f_B}$ equals to $x^{-i} * f_B$, $\widehat{e_i}$ is corrected with $\widehat{f_B}$.

- **Stage 3:** For sufficiently large k, the value of $\widehat{f_B}$ can be derived from the average of the polynomials $\widehat{e_1}$,$\widehat{e_2}$, $\cdots$,$\widehat{e_k}$. Furthermore, f_B can be recovered from $\widehat{f_B}$ according to the algorithm introduced in [9].
  Since the proxy has the re-encryption key $rk_{A \to B} = f_A * f_B^{-1} \mod q$ and the public key of Alice is $h_A = p * g_A * f_A^{-1} \mod q$. Once the attaker recovers the private key f_B, f_A can be found by computing $f_A = rk_{A \to B} * f_B \mod q$ and g_A can be found by computing $g_A = p * g_A * h_A \mod q$.

5 Key Recovery Attack Against NTRUReEncrypt

In this section, we show that curious Bob can recover Alice's secret keys f_A when collecting enough ciphertexts from a single message.

5.1 Key Idea of Recovering f_A for Bob

For simplicity, suppose a message m could be encrypted l times using the same public key f_A of Alice, and the ciphertexts are computed as

$$C_{A_i} = h_A * s_i + m \quad i = 1, \cdots, l.$$

When Bob receives C_{B_i} corresponding to C_{A_i}, he can first computes the following relation

$$f_B * C_{B_i} = p * g_A * s_i + p * f_B * e_i + m * f_A.$$

Next, Bob obtains

$$f_B * \sum_{i=1}^{l} C_{B_i} = p * g_A * (\sum_{i=1}^{l} s_i) + p * f_B * (\sum_{i=1}^{l} e_i) + l * m * f_A.$$

Note that p, g_A, s_i, f_B, and e_i are small. We can expect that for sufficiently large l, $p * g_A * \sum_{i=1}^{l} s_i/k$ and $p * f_B * \sum_{i=1}^{l} e_i/k$ are small enough. Since the coefficients of $m * f_A$ are integer, the following equation holds with high probability,

$$m * f_A = Round(\sum_{i=1}^{l} f_B * C_{B_i}/l).$$

where $Round(\cdot)$ is a rounding function.

Since Bob can decrypt correctly to obtain the message m, so the unknown private key f_A will be recovered by solving the above linear equations.

5.2 Analyze the Size of l

For the attack, we need l that satisfies

$$\left\| p * g_A * \sum_{i=1}^{l} s_i/l \right\|_{\infty} \le \frac{1}{4}, \quad \left\| p * f_B * \sum_{i=1}^{l} e_i/l \right\|_{\infty} \le \frac{1}{4},$$

to ensure $m * f_A = Round(\sum_{i=1}^{l} f_B * C_{B_i}/l)$.

For any $s_i \in \mathcal{T}_{(d_s,d_s)}$, let $X = \sum_{i=1}^{l} s_i/l = (X_0, \cdots, X_{N-1})$. For sufficiently large l, the central limit theorem states that X has the N dimension normal distribution $\mathcal{N}(0, \Sigma)$, where the diagonal elements of Σ are $\frac{2d_s}{lN}$ and the rest are $\frac{-2d_s}{lN(N-1)}$.

We define $\|\Sigma\|_{\infty} = \max_i \sum_{j=1}^{N} |\sigma_{ij}|$, where σ_{ij} is the component of Σ. Now we have $\|\Sigma\|_{\infty} = \frac{4d_s}{lN}$. Let λ denote the maximal eigenvalue of Σ, then we have $\lambda \le \|\Sigma\|_{\infty} = \frac{4d_s}{lN}$.

On the other hand, there exists $Y = (Y_0, Y_1, \cdots, Y_{N-1})$ and an orthogonal matrix D, such that

$$X = YD,$$

where $Y_0, \cdots, Y_{N-1}$ are independent variables and the covariance matrix of Y is a diagonal matrix in which the elements on the diagonal are the eigenvalues of the covariance matrix Σ of X. Hence, let $Var(Y_j)$ denote the variance of Y_j, we know

$$\lambda = \max_j Var(Y_j) \leq \frac{4d_s}{lN}.$$

To estimate the probability $\Pr\left[\left\|p * g_A * \sum_{i=1}^{l} s_i/l\right\|_\infty \leq \frac{1}{4}\right]$, we can consider the probability $\Pr\left[\bigcap_{j=1}^{N} |X_j| \leq \epsilon\right]$, where ϵ satifies

$$\left\|p * g_A * (\sum_{i=1}^{l} s_i/l)\right\|_\infty \leq 2d_g p\epsilon \leq \frac{1}{4}.$$

Since $X_0, \cdots, X_{N-1}$ are not independent, we can consider the probability $\Pr\left[\bigcap_{j=1}^{N} |Y_j| \leq \epsilon/N\right]$ instead, where

$$\Pr\left[\bigcap_{j=1}^{N} |Y_j| \leq \epsilon/N\right] = \prod_{i=0}^{N-1} \Pr\left[|Y_j| \leq \epsilon/N\right].$$

By the Chebyshev inequality, we know that

$$\Pr\left[|Y_j| \leq \epsilon/N\right] \geq 1 - \frac{Var(Y_j)}{(\epsilon/N)^2}.$$

Finally we obtain

$$\Pr\left[\bigcap_{j=1}^{N} |Y_j| \leq \epsilon/N\right] \geq (1 - \frac{\lambda}{(\epsilon/N)^2})^N \geq (1 - \frac{4d_s N}{l\epsilon^2})^N.$$

Recall that e_i has the same distribution, a similar analysis applies. So, for simplicity, we compute the value of l that makes $\frac{4d_s N}{l\epsilon^2}$ as small as possible by setting $\epsilon = \frac{1}{8pd_g}$. We roughly give the l needed to recover the private key with overwhelming probability (0.8 for the following table) in ees1087ep1, ees1171ep1, ees1499ep1, ntru-443 and ntru-743 (Table 5).

Table 5. The approximate number of received re-encrypted ciphertexts

Instance	l
ees1087ep1	$4.06 \cdot 10^{17}$
ees1171ep1	$4.83 \cdot 10^{17}$
ees1499ep1	$9.67 \cdot 10^{17}$
ntru-443	$1.26 \cdot 10^{16}$
ntru-743	$1.82 \cdot 10^{17}$

6 Conclusion

In this paper, we presented two key recovery attacks against NTRUReEncrypt to show the weakness of the scheme.

- The first one is based on the attack introduced in [14]. The attacker has access to an oracle that can detect whether the valid ciphertext can be decrypted correctly or not. The countermeasures to mitigate this attack is by tuning the parameters to ensure that the decryption failure probability is negligible, i.e., $< 2^{-128}$.
- The second one is based on the fact that Bob knows the original message m, so he can compute an equation in the form of $p * g_A * \sum_{i=1}^{l} s_i/l + m * f_A + p * f_B * \sum_{i=1}^{l} e_i/l = f_B * \sum_{i=1}^{l} C_{B_i}/l$. For sufficiently large l, $p * g_A * \sum_{i=1}^{l} s_i/l$ and $p * f_B * \sum_{i=1}^{l} e_i/l$ converge to 0. Hence f_A can be recovered by solving $m * f_A = Round(f_B * \sum_{i=1}^{l} C_{B_i}/l)$.

Acknowledgement. The authors would like to thank David Nuñez for his helpful discussions.

References

1. NIST post-quantum cryptography project. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions
2. ANSI X 98: Lattice-based polynomial public key establishment algorithm for the financial services industry. Technical report, ANSI (2010)
3. 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
4. Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU prime: reducing attack surface at low cost. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 235–260. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72565-9_12
5. 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
6. 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. ACM (2007)
7. Chen, L., et al.: Report on post-quantum cryptography. Technical report (2016). https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf
8. Fluhrer, S.R.: Cryptanalysis of ring-LWE based key exchange with key share reuse. IACR Cryptology ePrint Archive 2016, p. 85 (2016). http://eprint.iacr.org/2016/085
9. Gentry, C., Szydlo, M.: Cryptanalysis of the revised NTRU signature scheme. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 299–320. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_20

10. Hirschhorn, P.S., Hoffstein, J., Howgrave-Graham, N., Whyte, W.: Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 437–455. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01957-9_27
11. Hoffstein, J., Pipher, J., Schanck, J.M., Silverman, J.H., Whyte, W., Zhang, Z.: Choosing parameters for NTRUEncrypt. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 3–18. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_1
12. 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
13. Hoffstein, J., Silverman, J.H.: Reaction attacks against the NTRU public key cryptosystem. Technical report, NTRU Cryptosystems Technical Report (1999)
14. 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
15. Hülsing, A., Rijneveld, J., Schanck, J., Schwabe, P.: High-speed key encapsulation from NTRU. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 232–252. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_12
16. 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. ACM (2015)
17. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: 1994 Proceedings of 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. IEEE (1994)
18. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
19. Whyte, W., Howgrave-Graham, N., Hoffstein, J., Pipher, J., Silverman, J., Hirschhorn, P.: IEEE P1363. 1: draft standard for public-key cryptographic techniques based on hard problems over lattices. Technical report, IEEE (2008)
20. Xagawa, K., Tanaka, K.: Proxy re-encryption based on learning with errors (mathematical foundation of algorithms and computer science) (2010)
21. Zhang, Z., Chen, C., Hoffstein, J., Whyte, W.: NTRUencrypt. Technical report (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
22. Zhang, Z., Plantard, T., Susilo, W.: Reaction attack on outsourced computing with fully homomorphic encryption schemes. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 419–436. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31912-9_28