D. Nuñez, I. Agudo, and J. Lopez, "NTRUReEncrypt: An Efficient Proxy Re-Encryption Scheme Based on NTRU", *10th ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*, pp. 179-189, 2015.
http://doi.org/10.1145/2714576.2714585
NICS Lab. Publications: https://www.nics.uma.es/publications

NTRUReEncrypt: An Efficient Proxy Re-Encryption Scheme Based on NTRU

David Nuñez
Universidad de Málaga
Network, Information and Computer Security Laboratory (NICS Lab)
Málaga, Spain
dnunez@lcc.uma.es

Isaac Agudo
Universidad de Málaga
Network, Information and Computer Security Laboratory (NICS Lab)
Málaga, Spain
isaac@lcc.uma.es

Javier Lopez
Universidad de Málaga
Network, Information and Computer Security Laboratory (NICS Lab)
Málaga, Spain
jlm@lcc.uma.es

ABSTRACT

The use of alternative foundations for constructing more secure and efficient cryptographic schemes is a topic worth exploring. In the case of proxy re-encryption, the vast majority of schemes are based on number theoretic problems such as the discrete logarithm. In this paper we present NTRUReEncrypt, a new bidirectional and multihop proxy re-encryption scheme based on NTRU, a widely known lattice-based cryptosystem. We provide two versions of our scheme: the first one is based on the conventional NTRU encryption scheme and, although it lacks a security proof, remains as efficient as its predecessor; the second one is based on a variant of NTRU proposed by Stehlé and Steinfeld, which is proven CPA-secure under the hardness of the Ring-LWE problem. To the best of our knowledge, our proposals are the first proxy re-encryption schemes to be based on the NTRU primitive. In addition, we provide experimental results to show the efficiency of our proposal, as well as a comparison with previous proxy re-encryption schemes, which confirms that our first scheme outperforms the rest by an order of magnitude.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems

1. INTRODUCTION

Proxy re-encryption is a type of public-key cryptographic scheme that enables a user to delegate her decryption rights to other users. From a high-level viewpoint, a proxy re-encryption scheme is an asymmetric encryption scheme that permits a proxy to transform ciphertexts under Alice's public key pk_A into ciphertexts decryptable by Bob's secret key sk_B. In order to do this, the proxy is given a re-encryption key $rk_{A\to B}$, which makes this process possible. So, besides defining traditional encryption and decryption functions, a proxy re-encryption scheme also defines a re-encryption function for executing the transformation.

Since the introduction of proxy re-encryption by Blaze et al. in 1998 [11], many proxy re-encryption schemes have been proposed. The vast majority of these schemes are based on bilinear pairings, which have several drawbacks. For instance, pairings are known to be computationally intensive operations. With respect to security, pairing-based proxy re-encryption schemes ultimately rely on the hardness of the discrete logarithm problem. This poses a problem in the case that efficient cryptanalytic attacks against the discrete logarithm problem are developed or quantum computation becomes practical. This problem is also applicable to other schemes not based on pairings but that also depend on the discrete logarithm or on integer factorization assumptions.

Apart from bilinear pairings, the use of other foundations for constructing proxy re-encryption schemes has received little attention. Recently, some lattice-based proxy re-encryption schemes have been proposed [29, 4]. Lattice-based cryptography is a promising field, chiefly because of its role in post-quantum cryptography, but in some particular cases, also for its efficiency. As a prime example of widely-accepted and efficient lattice-based cryptosystem we find NTRU, which was introduced in 1996 by Hoffstein, Pipher and Silverman [20]. Since then, it has remained as the most practical lattice-based public-key cryptosystem, at the point of being standardized by IEEE Std 1363.1-2008 [28] and ANSI X9.98-2010 [2]. One of the main reasons of NTRU's popularity is its overwhelming performance with respect to traditional public-key cryptosystems. For instance, optimized versions of NTRU executed on a GPU proved to be up to 1000 times faster than RSA and 100 times faster than ECC [19]. The results behind the efficiency of NTRU are explained by the simplicity of its basic underlying operation, which is the convolution, i.e., the polynomial multiplication. NTRU uses polynomials with relatively small coefficients, so multiplications can be efficiently performed, even in constrained devices [8]. Another implication of this is that the size of the keys is relatively compact, when compared to other lattice-based schemes. In addition, the convolution operation can even be parallelized, which is very convenient as multicore and GPU processors are becoming more relevant nowadays.

One of the main drawbacks of NTRU is that it lacks a formal security proof, so its security stems from the practicality of best known attacks. This has meant that, over the

ASIA CCS'15, April 14–17, 2015, Singapore, Singapore

http://dx.doi.org/10.1145/2714576.2714585 .

years, NTRU has followed a somewhat "break-and-repair" approach, where advances in attacks have stimulated changes in the scheme and its set of parameters, although it has remained essentially the same. In order to advance towards solving this gap, Stehlé and Steinfeld proposed in 2011 a provable-secure variant of NTRU [25, 26], whose security is based on the assumed hardness of lattice problems.

Our contribution. In this paper, we propose new bidirectional proxy re-encryption schemes based on NTRU. Our first result, NTRUReEncrypt, is a slight modification of the conventional NTRU encryption scheme; this proposal extends the original scheme in order to support re-encryption, thus it remains just as efficient. However, it lacks of a formal security proof, as does NTRU. For this reason, we also present PS-NTRUReEncrypt, a provably-secure version based on the variant of the NTRU primitive proposed by Stehlé and Steinfeld in [25, 26]. To the best of our knowledge these are the first proxy re-encryption schemes based on NTRU, and one of the first to be based on lattices. In addition, we complement our proposal with an experimental comparison of the performance of several proxy re-encryption schemes, which confirms that our first scheme outperforms the rest by an order of magnitude.

Related work. The notion of proxy re-encryption was introduced in 1998 by Blaze et al. [11]; their proposal, which is usually referred to as the BBS scheme, is bidirectional (it is trivial to obtain $rk_{B\to A}$ from $rk_{A\to B}$) and multihop (the re-encryption process can be repeated multiple times), but not resistant to collusions between the proxy and one of the users. Ateniese, Fu, Green and Hohenberger proposed in [6] new proxy re-encryption schemes based on bilinear pairings. Their schemes are unidirectional, unihop and resistant to collusions. Green and Ateniese propose an identity-based proxy re-encryption scheme in [18]; however, this scheme is not resistant to collusions. An improved proposal that is secure against chosen-ciphertext attacks (CCA) is presented in [13], but again, it is not collusion-resistant. In [12], Canetti and Hohenberger present a CCA-secure bidirectional scheme; based on this security model, Libert and Vergnaud propose in [21] a unidirectional scheme with chosen-ciphertext security in the standard model. Another interesting proposal is presented in [5], where the authors define the notion of *key privacy* in the context of proxy re-encryption, which prevents the proxy to derive the identities of both sender and receiver from a re-encryption key.

As mentioned, the security in these schemes is based on number theoretic assumptions, such as the hardness of the discrete logarithm problem, but the use of other foundations for constructing proxy re-encryption schemes has been much less explored. In [29], Xagawa and Tanaka present the first lattice-based proxy re-encryption sheme, which relies on the Learning With Errors (LWE) problem; this scheme is bidirectional, multihop and CPA-secure. After this seminal work, Aono et al [4] proposed a unidirectional and multihop scheme, also based on the LWE problem.

With regard to the NTRU primitive, it has already been used as the basis for constructing other cryptographic schemes, such as multikey fully homomorphic encryption [22], private information retrieval [3], and public key traitor tracing scheme [23].

Organization. The rest of this paper is organized as follows: In Section 2, we provide some basic definitions for bidirectional proxy re-encryption schemes. In Section 3, we describe the NTRU encryption scheme and, based on it, we present our bidirectional proxy re-encryption scheme NTRUReEncrypt. In Section 4, we describe the provably-secure version of NTRU from Stehlé and Steinfeld and present our second scheme, PS-NTRUReEncrypt. In Section 5, we describe the experimental results from a prototype implementation of our proposals, as well as a comparison with previous proxy re-encryption schemes. Finally, Section 6 concludes the paper and future work is outlined.

2. DEFINITIONS

In this section, we provide some basic definitions relevant to bidirectional proxy re-encryption schemes. We first define the specification of a bidirectional CPA-secure proxy re-encryption scheme, based on the one given by Canetti and Hohenberger [12].

Definition 1. Bidirectional PRE. A bidirectional proxy re-encryption scheme is a tuple of algorithms (Setup, KeyGen, ReKeyGen, Enc, ReEnc, Dec):

- $\mathsf{Setup}(1^k) \to params$. On input the security parameter 1^k, the setup algorithm Setup outputs the parameters $params$[1].
- $\mathsf{KeyGen}() \to (pk_A, sk_A)$. The key generation algorithm KeyGen outputs a pair of public and secret keys (pk_A, sk_A) for user A.
- $\mathsf{ReKeyGen}(sk_A, sk_B) \to rk_{A\to B}$. On input the secret keys sk_A and sk_B, the re-encryption key generation algorithm ReKeyGen outputs a re-encryption key $rk_{A\to B}$.
- $\mathsf{Enc}(pk_A, M) \to C_A$. On input the public key pk_A and a message $M \in \mathcal{M}$, the encryption algorithm Enc outputs a ciphertext $C_A \in \mathcal{C}$.
- $\mathsf{ReEnc}(rk_{A\to B}, C_A) \to C_B$. On input a re-encryption key $rk_{A\to B}$ and a ciphertext $C_A \in \mathcal{C}$, the re-encryption algorithm ReEnc outputs a second ciphertext $C_B \in \mathcal{C}$.
- $\mathsf{Dec}(sk_A, C_A) \to M$. On input the secret key sk_A and a ciphertext $C_A \in \mathcal{C}$, the decryption algorithm Dec outputs a message $M \in \mathcal{M}$.

Once we have defined which are the inputs and outputs of a bidirectional proxy re-encryption scheme, we formulate the concept of correctness, also based on the one given by [12]. In this case, as these kinds of schemes are usually multihop, we refer to multihop correctness (i.e., ciphertexts that can be re-encrypted multiple times).

Definition 2. Multihop Correctness. A bidirectional PRE scheme (Setup, KeyGen, ReKeyGen, Enc, ReEnc, Dec) is multihop correct with respect to plaintext space $\mathcal{M}$ if:

- For all (pk_A, sk_A) output by KeyGen and all messages $M \in \mathcal{M}$, it holds that $\mathsf{Dec}(sk_A, \mathsf{Enc}(pk_A, M)) = M$.
- For any sequence of pairs (pk_i, sk_i) output by KeyGen, with $0 \le i \le N$, all re-encryption keys $rk_{j\to j+1}$ output by $\mathsf{ReKeyGen}(sk_j, sk_{j+1})$, with $j < N$, all messages

[1]In the following, we will assume $params$ is globally known, so we will omit it in the functions.

$M \in \mathcal{M}$, and all ciphertexts C_1 output by $\mathsf{Enc}(pk_1, M)$, it holds that:

$$\mathsf{Dec}(sk_N, \mathsf{ReEnc}(rk_{N-1\to N}, ...\mathsf{ReEnc}(rk_{1\to 2}, C_1))) = M$$

If for any $M \in \mathcal{M}$ correctness holds only with probability 1 minus a negligible quantity, we say that the scheme is correct with respect to $\mathcal{M}$.

Finally, we define the security game we use for bidirectional CPA-secure proxy re-encryption schemes. This definition is adapted from [12], [29] and [5].

Definition 3. Bidirectional PRE CPA-security game Let k be the security parameter. Let $\mathcal{A}$ be the adversary, and $\mathcal{H}, \mathcal{C}$ the sets of indices of honest and corrupt users, respectively. The game consists of an execution of $\mathcal{A}$ with the following oracles, which can be invoked multiple times in any order, subject to the constraints below:

Phase 0: The challenger takes a security parameter 1^k, obtains global parameters $params \leftarrow \mathsf{Setup}(1^k)$ and initializes sets $\mathcal{H}, \mathcal{C}$ to $\emptyset$. The challenger generates the public key pk^* of target user i^*, adds i^* to $\mathcal{H}$, and sends pk^* to the adversary.

Phase 1:

- Uncorrupted key generation $\mathcal{O}_{honest}$: On input an index i, where $i \notin \mathcal{H} \cup \mathcal{C}$, the oracle obtains a new keypair $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}()$ and adds index i to $\mathcal{H}$. The adversary receives pk_i.
- Corrupted key generation $\mathcal{O}_{corrupt}$: On input an index i, where $i \notin \mathcal{H} \cup \mathcal{C}$, the oracle obtains a new keypair $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}()$ and adds index i to $\mathcal{C}$. The adversary receives (pk_i, sk_i).

Phase 2:

- Re-encryption key generation $\mathcal{O}_{rkgen}$: On input (i, j), where $i \neq j$, and either $i, j \in \mathcal{H}$ or $i, j \in \mathcal{C}$, the oracle returns $rk_{i\to j} \leftarrow \mathsf{ReKeyGen}(sk_i, sk_j)$.
- Challenge oracle $\mathcal{O}_{challenge}$: This oracle can be queried only once. On input (M_0, M_1), the oracle chooses a bit $b \leftarrow \{0, 1\}$ and returns the challenge ciphertext $C^* \leftarrow \mathsf{Enc}(pk^*, M_b)$, where pk^* corresponds to the public key of target user i^*.

Phase 3:

- Decision: Eventually, $\mathcal{A}$ outputs guess $b' \in \{0, 1\}$. $\mathcal{A}$ wins the game if and only if $b' = b$.

As already noted in [5], we only allow queries to $\mathcal{O}_{rkgen}$ where users are either both corrupt or both honest. Otherwise, in a bidirectional setting these queries would corrupt honest users and could be used to simulate a decryption oracle, which is not considered in CPA security. That is, we are adopting a static corruption model. Note that we have not included a re-encryption oracle. On the one hand, in the case of CPA security and multihop schemes, the addition of a re-encryption oracle that allows to re-encrypt ciphertexts from honest to corrupt users could be used to simulate a decryption oracle, which is by definition out of the scope of CPA. On the other hand, restricting the re-encryption oracle only to the honest-to-honest and corrupt-to-corrupt cases would result in a superfluous oracle, since the same capabilities are acheived by means of $\mathcal{O}_{rkgen}$. For these reasons, a re-encryption oracle is not provided.

3. A PROXY RE-ENCRYPTION SCHEME BASED ON NTRU

In this section we firstly introduce the conventional NTRU encryption scheme and describe its operation. Then, we extend it in order to construct a bidirectional multihop proxy re-encryption scheme.

3.1 The NTRU Encryption Scheme

The NTRU encryption scheme, originally proposed by Hoffstein, Pipher and Silverman in [20], is one of the first public key encryption schemes based on lattices. The main reason behind the interest in NTRU is its efficiency, which is much better than other public-key cryptosystems and is even comparable to symmetric ciphers. In addition, its security is conjectured to be based on hard problems over lattices, although it lacks a formal proof.

We now briefly describe the operation of NTRU. The original NTRU cryptosystem is defined over the quotient ring $\mathcal{R}_{NTRU} = \mathbb{Z}[x]/(x^n - 1)$, where n is a prime parameter. Thus, elements of the ring $\mathcal{R}_{NTRU}$ are integer polynomials of degree less than n; the operators $+$ and $\cdot$ denote addition and multiplication in $\mathcal{R}_{NTRU}$, respectively. Other parameters of NTRU are the integer q, which is a small power of 2 of the same order of magnitude than n, and the small polynomial $p \in \mathcal{R}_{NTRU}$, which usually takes values $p = 3$ or $p = x + 2$. In general, operations over polynomials will be performed in $\mathcal{R}_{NTRU}$ modulo q or p, which will be denoted respectively as $\mathcal{R}_{NTRU}/q$ and $\mathcal{R}_{NTRU}/p$.

In NTRU, the private key sk consists of a polynomial $f \in \mathcal{R}_{NTRU}$ chosen at random, with a determined number of coefficients equal to 0, -1, and 1. The polynomial f must have an inverse in $\mathcal{R}_{NTRU}/q$ and $\mathcal{R}_{NTRU}/p$, respectively, f_q^{-1} and f_p^{-1}. For efficiency, f can be chosen to be congruent to 1 modulo p. The public key pk consists of the polynomial $h = p \cdot g \cdot f_q^{-1} \mod q$, where $g \in \mathcal{R}_{NTRU}$ is chosen at random.

The encryption process is very simple: a plaintext M from message space $\mathcal{R}_{NTRU}/p$ is encrypted to ciphertext $C = h \cdot s + M \mod q$, where s is a small random polynomial in $\mathcal{R}_{NTRU}$. For decrypting a ciphertext C, one must first compute $C' = f \cdot C$ and reduce it modulo q. If the original polynomial C' is sufficiently small then the result of the reduction is $pgs + fM \in \mathcal{R}_{NTRU}/q$. Next, C' is reducted modulo p, obtaining fM. Finally, this result is multiplied by f_p^{-1} to extract the original message M; note that the last step is redundant if f was selected congruent to 1 modulo p.

3.2 NTRUReEncrypt

Based on the NTRU encryption scheme, we define the proxy re-encryption scheme NTRUReEncrypt. Our proposal is essentialy an extension of the scheme that includes the definition of the re-encryption and re-encryption key generation algorithms. One of the fundamental differences of our scheme with respect to the original NTRU is that the secret polynomial f has to be congruent to 1 mod p, not for efficiency reasons, but because it is necessary to correctly decrypt re-encrypted ciphertexts.

3.2.1 The scheme

Now we present our scheme NTRUReEncrypt. In the following, the delegator is represented by user A, whereas the delegatee is user B. Our scheme is specified by the following algorithms:

- KeyGen(): The output of the key generation algorithm for user A is a pair of public and secret keys (pk_A, sk_A). If first chooses a pair of polynomials $(f_A, g_A) \in \mathcal{R}^2_{NTRU}$ at random, with a determined number of coefficients equal to 0, -1, and 1, as in the conventional NTRU scheme. The only added requirement is that f_A has to be congruent to 1 modulo p. As in the original case, f_A must have an inverse in $\mathcal{R}_{NTRU}/q$, denoted by f_A^{-1}. Note that now it is not necessary the inverse of f_A modulo p. The private key sk_A is the polynomial f_A, whereas the public key pk_A consists of the polynomial $h_A = p \cdot g_A \cdot f_A^{-1} \mod q$.

- ReKeyGen(sk_A, sk_B): On input the secret keys $sk_A = f_A$ and $sk_B = f_B$, the re-encryption key generation algorithm ReKeyGen computes the re-encryption key between users A and B as $rk_{A\to B} = sk_A \cdot sk_B^{-1} = f_A \cdot f_B^{-1}$. The re-encryption key can be computed by means of a simple three-party protocol, so neither A, B nor the proxy learns any secret key. The protocol, originally proposed in [12], is as follows: A selects a random $r \in \mathcal{R}_{NTRU}/q$ and sends $r \cdot f_A \mod q$ to B and r to the proxy; next, B sends $r \cdot f_A \cdot f_B^{-1} \mod q$ to the proxy, who computes $rk_{A\to B} = f_A \cdot f_B^{-1} \mod q$.

- Enc(pk_A, M): On input the public key pk_A and a message $M \in \mathcal{R}_{NTRU}/p$, the encryption algorithm Enc generates a small random polynomial $s \in \mathcal{R}_{NTRU}$, and outputs the ciphertext $C_A = h_A s + M$.

- 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 samples a random polynomial $e \in \mathcal{R}_{NTRU}$ and outputs ciphertext $C_B = C_A \cdot rk_{A\to B} + pe$.

- Dec(sk_A, C_A): On input the secret key $sk_A = f_A$ and a ciphertext C_A, the decryption algorithm Dec computes $C'_A = (C_A \cdot f_A) \mod q$ and outputs the original message $M = (C'_A \mod p)$.

Note that if ciphertexts are not re-encrypted, NTRUReEncrypt behaves in exactly the same way as the original NTRU encryption scheme.

3.2.2 Correctness

Now, we informally explain the reason why the re-encryption process works. Re-encrypted ciphertexts are of the form:

$$\begin{aligned} C_B &= C_A \cdot rk_{A\to B} + pe \\ &= (pg_A f_A^{-1} s + M) \cdot f_A f_B^{-1} + pe \\ &= pg_A f_B^{-1} s + pe + f_A f_B^{-1} M \end{aligned}$$

When decrypting a re-encrypted ciphertext, the delegatee multiplies the ciphertext with the secret key f_B:

$$\begin{aligned} C_B \cdot f_B &= (pg_A f_B^{-1} s + pe + f_A f_B^{-1} M) \cdot f_B \\ &= pg_A s + pe f_B + f_A M \end{aligned}$$

Now, taking modulo p, we get rid of the additional terms. Recall that we require secret key polynomial f_A to fulfill $f_A = 1 \mod p$, so $(C_B \cdot f_B) \mod p = (pg_A s + pe f_B + f_A M) \mod p = M$, which is the original message.

The inclusion of the random term e during the re-encryption phase is necessary in order to prevent a simple ciphertext-only attack from the delegatee. Imagine that a re-encrypted ciphertext is of the form $C_B = C_A \cdot rk_{A\to B} = C_A \cdot f_A \cdot f_B^{-1}$; that is, without the random term e. Then, the delegatee could extract the secret key of the delegator based on the observation that $C_B \cdot f_B = C_A \cdot f_A$ holds, since $C_B = C_A \cdot rk_{A\to B}$. Next, assuming that C_A is invertible modulo q, the attacker can extract the secret key by computing $f_A = C_A^{-1} \cdot C_B \cdot f_B$.

The scheme is also multihop, that is, it supports multiple re-encryptions. However, this property is limited, since the addition of the error term during the re-encryption produces an increasing error that grows on each hop, until eventually, decryption fails. Our experiments show that this depends heavily on the choice of parameters. This issue is discussed in more detail in Section 5.1.

3.2.3 Analysis

The resulting proxy re-encryption scheme preserves the performance level of the original NTRU; a detailed experimentation on this matter is presented in Sections 5.1 and 5.2. These experimental results show that NTRUReEncrypt outperforms other proxy re-encryption schemes by an order of magnitude.

A theoretical analysis of the computational costs associated with NTRUReEncrypt shows that the main algorithms, namely encryption, decryption and re-encryption, only need a single multiplication (actually, re-encryption takes an additional multiplication for computing the term pe, but this can be computed beforehand and, in addition, the polynomial p is small and known in advance). As already stated in the introduction, the core operation in NTRU is the multiplication of polynomials, which can be done in $O(n \log n)$ time using the Fast Fourier Transform (FFT) [19].

Table 1: Space costs of NTRUReEncrypt

Element	Size
Keys	$O(n \cdot \log_2 q)$
Message	$O(n)$
Ciphertext	$O(n \cdot \log_2 q)$
Ciphertext expansion	$O(\log_2 q)$

As for the space costs associated to NTRUReEncrypt, Table 1 presents a theoretical analysis of the size of the messages, keys, and ciphertexts, as well as a measure of the ciphertext expansion it produces. It can be seen that the expansion is logarithmic in the parameter q, and that the size of keys is within $O(n \log_2 q)$. This can be put in contrast with other lattice-based cryptosystems, such as the proxy re-encryption scheme from Aono et al [4], where keys and messages are tipically matrices whose dimensions grow linearly with the parameter n. This implies that the size of these matrices is at least quadratic with respect to n.

Table 2 illustrates this fact by showing a comparison between Aono's scheme and ours. The space costs from Aono's scheme are determined by the parameters shown in [4] for the case of 143 bits of security and 128-bit plaintexts, whereas our figures are calculated using a parameter set that achieves 256 bits of security and supports up to 186-bits plaintexts. More details about the parameters used are given in Section 5.1. It can be seen that our space costs are lower in

Table 2: Comparison of space costs (in KB)

Size	Aono et al. [4]	NTRUReEncrypt
Public keys	60.00	1.57
Secret key	60.00	1.57
Re-Encryption key	2520.00	1.57
Ciphertext	0.66	1.57

comparison, even considering the difference in bits of security. In the case of re-encryption keys, the difference in size is remarkable as it grows up to 3 orders of magnitude. On the other hand, our ciphertexts are slightly bigger. Another interesting fact is that we use the same kind of polynomials for conveying all the types of keys and ciphertexts, so the space costs are the same in all cases (1.57 KB).

It is easy to check that our scheme is bidirectional. Given $rk_{A\to B} = f_A f_B^{-1}$, the proxy can easily compute $rk_{B\to A} = (rk_{A\to B})^{-1} = f_B f_A^{-1}$. This property is a consequence of how re-encryption keys are constructed ($rk_{A\to B} = sk_A \cdot sk_B^{-1}$), in the same way as in other bidirectional schemes [11, 12, 27].

With respect to the security of the scheme, we cannot overcome the problem of NTRU lacking a formal security proof. However, we can study it from a practical way, as NTRU does. For instance, the time required for a lattice-based attack to the public key is conjectured to be exponential in n [20]. This is also relevant to the re-encryption scheme since extracting private keys from the re-encryption key is at least as hard as attacking NTRU public key, as they are constructed in the same way: for instance, the public key of user B is of the form $pg_B f_B^{-1}$, whereas re-encryption key $rk_{A\to B}$ is of the form $f_A f_B^{-1}$. In NTRU, the polynomials f are larger than the polynomials g, i.e., they have more non-zero terms, so the term f_A of a re-encryption key would be larger than the term pg_B of a public key. However, as it happens to the majority of bidirectional proxy re-encryption schemes [11, 12, 27], our scheme is not collusion-safe. That is, it is vulnerable to a collusion of the proxy and the users, since extracting the delegator's secret key f_A is as simple as computing $rk_{A\to B} \cdot f_B$. The same problem applies to the delegatee's secret key.

As a way to provide a formal security proof of our scheme, we explored how this problem has been addressed in the literature. An interesting approach is made by Stehlé and Steinfeld in [25, 26], where they present a variant of NTRU which is proven CPA-secure under lattice-based assumptions. Based on this solution, we build a second version of our NTRU-based proxy re-encryption scheme, called PS-NTRUReEncrypt, which is provably-secure.

4. PROVABLY SECURE NTRUREENCRYPT

In this section we provide a second proxy re-encryption scheme, called PS-NTRUReEncrypt, that is provable secure under a hard problem for lattices, namely the Ring-LWE assumption. Before explaining our provably-secure scheme, we must first establish some background concepts; after that, we describe the NTRU variant proposed in [25, 26] by Stehlé and Steinfeld, which serves as a basis for our second scheme.

4.1 Background

Before continuing, we provide some basic definitions and present the notation that is used throughout this section. Most of this definitions and notation is taken from [25, 26].

Let $\Phi(x) = x^n + 1$, with n a power of 2; that is, $\Phi(x)$ is the $2n$-th cyclotomic polynomial. Let q be a prime integer such that $q = 1 \mod 2n$. Let $\mathcal{R}$ be the ring $\mathbb{Z}[x]/\Phi(x)$, and $\mathcal{R}_q = \mathcal{R}/q = \mathbb{Z}_q[x]/\Phi(x)$. We will denote the set of invertible elements of $\mathcal{R}_q$ as $\mathcal{R}_q^\times$. Although elements of $\mathcal{R}$ are polynomials, we often treat them as vectors. We denote by $\|\cdot\|$ and $\|\cdot\|_\infty$ the Euclidean norm and infinite norm, respectively. During the proofs in Section 4.3.1 and 4.3.2, we use the asymptotic notations $O(\cdot), \omega(\cdot), \Omega(\cdot)$. A function $f(n)$ is negligible if $f(n) = n^{-\omega(1)}$.

The Ring Learning With Errors (Ring-LWE) problem is a hard decisional problem based on lattices introduced by Lyubaskevsky et al in [24]. In this paper, we use an adapted version of this problem found in [25, 26].

Definition 4. The Ring-LWE problem. Let $s \in \mathcal{R}_q$ and ψ a distribution over $\mathcal{R}_q^\times$, then we define $A_{s,\psi}^\times$ as the distribution that samples pairs of the form (a, b), where a is chosen uniformly from $\mathcal{R}_q^\times$ and $b = a \cdot s + e$, for some e sampled from ψ. The distribution $A_{s,\psi}^\times$ is also called the Ring-LWE distribution. The Ring-LWE problem is to distinguish distribution $A_{s,\psi}^\times$ from a uniform distribution over $\mathcal{R}_q^\times \times \mathcal{R}_q$. The Ring-LWE assumption is that this problem is computationally infeasible.

This variant of the Ring Learning With Errors problem is denoted in [25, 26] as the R-LWE$_{\text{HNF}}^\times$ problem; however, we will not use that notation for simplicity. As in [25, 26], the error distributions ψ are sampled from a family of distributions Ψ_α, with parameter α; for more details of the definition of these distributions, see [25, 26].

4.2 Provably Secure NTRU

Stehlé and Steinfeld proposed in [25, 26] a variant of the NTRU primitive which is proven CPA-secure under the hardness assumption of the Ring-LWE problem. This scheme is defined over the rings $\mathcal{R}$ and $\mathcal{R}_q$, determined by parameters n and q, as defined in previous section. The plaintext space $\mathcal{M}$ corresponds to the ring $\mathcal{R}/p$, where $p \in \mathcal{R}_q^\times$ is also a parameter of the scheme; as conventional NTRU, typical values for this parameter are $p = 3$ and $p = x+2$, but in this scheme one may choose $p = 2$, since q is prime. The parameter α characterizes the family of distributions Ψ_α, and the parameter σ is the standard deviation of the Gaussian distribution used during the key generation algorithm. Thus, global parameters are a tuple $(n, q, p, \alpha, \sigma)$. The algorithms of this encryption scheme are the following:

- KeyGen(): The output of the key generation algorithm for user A is the pair of public and secret keys $(pk_A, sk_A) \in \mathcal{R}_q^\times \times \mathcal{R}_q^\times$. Let $D_{\mathbb{Z}^n,\sigma}$ be the Gaussian distribution over $\mathbb{Z}^n$ with standard deviation σ. The keys are computed as follows:
  1. Sample f' from $D_{\mathbb{Z}^n,\sigma}$; let $f_A = 1 + p \cdot f'$; if $(f_A \mod q) \notin \mathcal{R}_q^\times$, resample.
  2. Sample g_A from $D_{\mathbb{Z}^n,\sigma}$; if $(g_A \mod q) \notin \mathcal{R}_q^\times$, resample.
  3. Compute $h_A = p \cdot g_A \cdot f_A^{-1}$.
  4. Return secret key $sk_A = f_A$ and $pk_A = h_A$.
- Enc(pk_A, M): On input the public key pk_A and a message $M \in \mathcal{M}$, sample noise polynomials s, e from a

distribution from Ψ_α, and output ciphertext $C_A = h_A s + pe + M \in \mathcal{R}_q$.

- $\mathsf{Dec}(sk_A, C_A)$. On input the secret key $sk_A = f_A$ and a ciphertext C_A, the decryption algorithm Dec computes $C'_A = C_A \cdot f_A$ and outputs the message $M = (C'_A \mod p) \in \mathcal{M}$.

It can be seen that the operation of this scheme is very similar to the original NTRU, except for the generation of the keys and the inclusion of a noise term during encryptions. The authors prove that this NTRU variant is CPA-secure, over the hardness of the Ring-LWE problem. They also prove the correctness of their proposal, by establishing the conditions for avoiding decryption failures. Both proofs will serve as a basis for ours.

4.3 Provably-Secure NTRUReEncrypt

In this section we present the scheme PS-NTRUReEncrypt, which is based on the provably-secure variant of NTRU described in the previous section. This scheme uses the same global parameters as before, a tuple $(n, q, p, \alpha, \sigma)$. This scheme is defined by the following algorithms:

- $\mathsf{KeyGen}()$: The output of the key generation algorithm for user A is the pair of public and secret keys $(pk_A, sk_A) \in \mathcal{R}_q^\times \times \mathcal{R}_q^\times$. The keys are computed as follows:
  1. Sample f' from $D_{\mathbb{Z}^n,\sigma}$; let $f_A = 1 + p \cdot f'$; if $(f_A \mod q) \notin \mathcal{R}_q^\times$, resample.
  2. Sample g_A from $D_{\mathbb{Z}^n,\sigma}$; if $(g_A \mod q) \notin \mathcal{R}_q^\times$, resample.
  3. Compute $h_A = p \cdot g_A \cdot f_A^{-1}$.
  4. Return secret key $sk_A = f_A$ and $pk_A = h_A$.
- $\mathsf{ReKeyGen}(sk_A, sk_B)$: On input the secret keys $sk_A = f_A$ and $sk_B = f_B$, the re-encryption key generation algorithm $\mathsf{ReKeyGen}$ computes the re-encryption key between users A and B as $rk_{A\to B} = sk_A \cdot sk_B^{-1} = f_A \cdot f_B^{-1}$.
- $\mathsf{Enc}(pk_A, M)$: On input the public key pk_A and a message $M \in \mathcal{M}$, the encryption algorithm Enc samples noise polynomials s, e from a distribution from Ψ_α, and outputs ciphertext $C_A = h_A s + pe + M \in \mathcal{R}_q$.
- $\mathsf{ReEnc}(rk_{A\to B}, C_A)$: On input a re-encryption key $rk_{A\to B}$ and a ciphertext C_A, the re-encryption algorithm ReEnc samples noise polynomial e' from a distribution from Ψ_α and outputs ciphertext $C_B = C_A \cdot rk_{A\to B} + pe' \in \mathcal{R}_q$.
- $\mathsf{Dec}(sk_A, C_A)$. On input the secret key $sk_A = f_A$ and a ciphertext C_A, the decryption algorithm Dec computes $C'_A = C_A \cdot f_A$ and outputs the message $M = (C'_A \mod p) \in \mathcal{M}$.

It can be seen that this scheme is an extension of the one from Stehlé and Steinfeld, by including the definition of the re-encryption and re-encryption key generation algorithms. Note also that the same three-party protocol for computing the re-encryption key described in Section 3.2 for $\mathsf{NTRUReEncrypt}$ can be applied here.

4.3.1 Correctness

Following the description of the encryption, decryption and re-encryption algorithms, it is easy to informally check that the correctness conditions defined in Def. 2 are fulfilled:

- For all (pk_A, sk_A) output by KeyGen and all messages $M \in \mathcal{M}$, it holds that $\mathsf{Dec}(sk_A, \mathsf{Enc}(pk_A, M)) = M$.

  The evaluation of the encryption of an arbitraty message $M \in \mathcal{M}$ gives $\mathsf{Dec}(psg_A f_A^{-1} + pe + M, sk_A)$ as a result. Next, as described in the decryption algorithm, we take $C' = f_A \cdot (psg_A f_A^{-1} + pe + M) = psg_A + pef_A + Mf_A$. Finally, since $f_A = 1 \mod p$ and $psg_A = pef_A = 0 \mod p$, we have that $C' \mod p = M$.

- For any sequence of pairs (pk_i, sk_i) output by KeyGen, with $0 \le i \le N$, all re-encryption keys $rk_{j\to j+1}$ output by $\mathsf{ReKeyGen}(sk_j, sk_{j+1})$, with $j < N$, all messages $M \in \mathcal{M}$, and all ciphertexts C_1 output by $\mathsf{Enc}(pk_1, M)$, it holds that:

  $$\mathsf{Dec}(sk_N, \mathsf{ReEnc}(rk_{N-1\to N}, ...\mathsf{ReEnc}(rk_{1\to 2}, C_1))) = M$$

  If for any $M \in \mathcal{M}$ correctness holds only with probability 1 minus a negligible quantity, we say that the scheme is correct with respect to $\mathcal{M}$.

  Let us assume the sequence of secret keys $f_0, f_1, ..., f_N$. Since our scheme is multi-hop, a ciphertext re-encrypted N times will be of the form:

$$\begin{aligned} C_N &= pg_0 f_N^{-1} s + pe_0 f_0 f_N^{-1} + pe_1 f_1 f_N^{-1} + ... \\ &\quad + pe_{N-1} f_{N-1} f_N^{-1} + pe_N + Mf_0 f_N^{-1} \\ &= pg_0 f_N^{-1} s + \left[\sum_{i=0}^{N-1} pe_i f_i f_N^{-1}\right] + pe_N + Mf_0 f_N^{-1} \end{aligned} \quad (1)$$

  where e_i denotes the additional error terms added during the re-encryptions, and e_0 is the error term from the first (and unique) encryption. When decrypting C_N, and assuming there are no decryption failures, one obtains:

$$C'_N = C_N \cdot f_N = pg_0 s + \left[\sum_{i=0}^{N} pe_i f_i\right] + Mf_0 \quad (2)$$

  Since, $f_0 = 1 \mod p$ and $pg_0 s = pe_i f_i = 0 \mod p$, for $0 \ge i \ge N$, then we have that $C'_N \mod p = M$.

In the informal proof above, we have not considered the possibility of decryption failures. This is a recurrent issue from the first definition of the NTRU cryptosystem. It can be seen that the re-encryption process produces an increase in the error terms of the ciphertexts, which can potentially lead to decryption failures. We will now describe the correctness conditions of $\mathsf{NTRUReEncrypt}$ and prove that the decryption algorithm is capable of handling with this increase. This proof is basically a slight extension of the one given in [26, Lemma 3.7] and generalizes single encryption as well as multiple re-encryptions; in this case, single encryption is a handled as a particular case of multihop re-encryption (taking the number of re-encryptions $N = 0$).

THEOREM 1. *If* $deg(p) \leq 1$, $\omega(n^{0.25}\log n)\alpha N\|p\|^2\sigma \leq 1$, *and* $n^{0.75} \leq \alpha q$, *then the decryption algorithm of* PS-NTRUReEncrypt *succeeds in recovering* M *with probability* $1 - n^{-\omega(1)}$ *over the choice of* s, e_i, f_i, g_i, *for* $0 \leq i \leq N$.

PROOF. As stated, a ciphertext C_N re-encrypted N times has the form described in Equation 1. When decrypting C_N, the decryption algorithm computes C'_N, as described in Equation 2. This computation is implicitly performed modulo q, so $C'_N \in \mathcal{R}_q$. However, let us define $C''_N \in \mathcal{R}$, that is, not modulo q:

$$C''_N = pg_0 s + \left[\sum_{i=0}^{N} pe_i f_i\right] + Mf_0 \in \mathcal{R}$$

In order for the decryption algorithm to succeed $\|C''_N\|_\infty \leq q/2$, so $C'_N = C''_N$ in $\mathcal{R}$. Since $f_0 = 1 \mod p$ and $pg_0 s = pe_i f_i = 0 \mod p$, for $0 \leq i \leq N$, then we have that $C''_N \mod p = C'_N \mod p = M$ and the decryption algorithm succeeds. Then, it is sufficient to give an upper bound on the probability that $\|C''_N\|_\infty \geq q/2$.

From Lemma 1 (see Appendix A), polynomials g_0 and f_i, for $0 \geq i \geq N$, have Euclidean norms $\leq 4\sqrt{n}\|p\|\sigma$, with probability $\geq 1 - 2^{-n+3}$. From the properties of the ring $\mathcal{R}$ [17], we know that for all $u, v \in \mathcal{R}$, $\|u \cdot v\| \leq \sqrt{n} \cdot \|u\| \cdot \|v\|$. As a particular case, since $deg(p) \leq 1$, then $\|p \cdot u\| \leq 2\|p\| \cdot \|u\|$. Hence, it follows that $\|pf_i\|, \|pg_0\| \leq 8\sqrt{n}\|p\|^2\sigma$, with probability $\geq 1-2^{-n+3}$. Now, from Lemma 2 (see Appendix A), if $n^{0.25} \leq \alpha q$ we have that with probability $\geq 1-n^{-\omega(1)}$:

$$\|pf_i e_i\|_\infty, \|pg_0 s\|_\infty \leq 8\alpha q n^{0.25}\omega(\log n)\|p\|^2\sigma$$

Apart from this, since $\|M\| \leq \|p\|$ [26], we know that:

$$\|f_0 M\|_\infty \leq \|f_0 M\| \leq 4n\|p\|^2\sigma$$

Hence $\|C''_N\|_\infty \leq \left[8(N+2)\alpha q n^{0.25}\omega(\log n) + 4n\right]\|p\|^2\sigma$, and taking $n^{0.75} \leq \alpha q$, then:

$$\|C''_N\|_\infty \leq (8N+20)\alpha q n^{0.25}\omega(\log n)\|p\|^2\sigma$$

As we assumed initially that $\omega(n^{0.25}\log n)\alpha N\|p\|^2\sigma \leq 1$, then we have that $\|C''_N\|_\infty \leq q/2$, with probability $\geq 1 - n^{-\omega(1)}$. □

It can be seen that for the case of single encryption ($N = 0$), the determined bounds are the same than in [26, Lemma 3.7].

4.3.2 Security proof

In this section, we proceed to prove that the scheme PS-NTRUReEncrypt is CPA-secure. This proof is also based on the one given in [26, Lemma 3.8].

THEOREM 2. *Suppose that* n *is a power of 2, and* q *a prime number such that* $q = 1 \mod 2n$. *Let* $\epsilon \in (0, \frac{1}{3}), \delta > 0, p \in \mathcal{R}_q^\times$ *and* $\sigma \geq n\sqrt{ln(8nq)} \cdot q^{\frac{1}{2}+\epsilon}$. *If there exists and IND-CPA attack against* PS-NTRUReEncrypt *with probability* $\frac{1}{2}+\delta$, *then there exists an algorithm that solves the Ring-LWE problem with probability* $\frac{\delta}{2} - q^{-\Omega(n)}$.

PROOF. Let us assume, by contradiction, that we have an adversary $\mathcal{A}$ that breaks PS-NTRUReEncrypt with probability $\frac{1}{2} + \delta$. We construct an algorithm $\mathcal{B}$ against the Ring-LWE problem as follows:

Let $\mathcal{O}_{sample}$ be an oracle that samples tuples (h', C') from either the uniform distribution over $\mathcal{R}_q^\times \times \mathcal{R}_q$ or $A^\times_{s,\psi}$. Algorithm $\mathcal{B}$ first gets a sample (h', C') from $\mathcal{O}_{sample}$. Next, $\mathcal{B}$ generates target public key $pk^* = h^* = p \cdot h'$ and gives pk^* to $\mathcal{A}$. Note that, regardless of the input distribution, h' will be sampled uniformly from $\mathcal{R}_q^\times$, and since $p \in \mathcal{R}_q^\times$, then $h^* = p \cdot h'$ is uniformly random in $\mathcal{R}_q^\times$; however, by [26, Theorem 3], h^* is within negligible statistical distance $q^{-\Omega(n)}$ to public keys generated by KeyGen. $\mathcal{B}$ will also generate an invalid, but correctly distributed, secret key polynomial f^* as described in the first step of the key generation process, so target secret key is $sk^* = f^*$.

Both honest and corrupted key generation are done as in the PS-NTRUReEncrypt scheme. Re-encryption key generation queries are done using the secret keys derived on the key generation queries. Once $\mathcal{A}$ queries the challenge oracle with inputs M_0 and M_1, $\mathcal{B}$ randomly picks $b \leftarrow \{0, 1\}$, and constructs the challenge ciphertext as $C^* = p \cdot C' + M_b$. Adversary $\mathcal{A}$ receives C^* and, eventually, outputs its guess b'; if $b' = b$, then algorithm $\mathcal{B}$ outputs 1.

On the one hand, when the input to $\mathcal{B}$ is a sample from $A^\times_{s,\psi}$, then it will be of the form $(h', C' = h's + e)$, for some $s, e \leftarrow \psi$. In this case, the challenge ciphertext C^* is a correct encryption of M_b under h^*, since $C^* = p \cdot C' + M_b = p(h's+e)+M_b = h^*s+pe+M_b$, and $\mathcal{A}$ will succeed in distinguishing M_b with probability $\frac{1}{2} + \delta - q^{-\Omega(n)}$. Hence, $\mathcal{B}$ will determine that the sample was from the $A^\times_{s,\psi}$ distribution with the same probability.

On the other hand, when the sample is from the uniform distribution over $\mathcal{R}_q^\times \times \mathcal{R}_q$ then C' is uniformly random in $\mathcal{R}_q$, and since $p \in \mathcal{R}_q^\times$, so is $p \cdot C'$. The challenge ciphertext $C^* = p \cdot C' + M_b$ will be then uniformly random in $\mathcal{R}_q$, as in the original distribution of ciphertexts, and independent of b. In this case, algorithm $\mathcal{B}$ does not have any advantage in distinguishing the distribution and outputs 1 with probability $\frac{1}{2}$. Overall, algorithm $\mathcal{B}$ succeeds with probability $(\frac{1}{2})(\frac{1}{2}+\delta-q^{-\Omega(n)}) + (\frac{1}{2})(\frac{1}{2}) = \frac{1}{2} + \frac{\delta}{2} - q^{-\Omega(n)}$. This contradicts the Ring-LWE assumption when δ is non-negligible.

□

Remark: In this simulation, each re-encryption key that involves the target public key is invalid, as generating a re-encryption key implies knowledge of the secret keys of the users involved. In this case, the target public key is constructed from $\mathcal{O}_{sample}$, so the simulator does not know the corresponding secret key. However, since the secret key of the target used generated at the beginning of the simulation is correctly distributed, any re-encryption key computed from this secret key is indistinguishable from a valid one. A similar argument is made in [18].

4.3.3 Analysis

The properties of PS-NTRUReEncrypt are the same as our first scheme. These properties are:

- Bidirectional: Given $rk_{A\to B} = f_A f_B^{-1}$, one can easily compute $rk_{B\to A} = (rk_{A\to B})^{-1} = f_B f_A^{-1}$.
- Multihop: It supports multiple re-encryptions, as shown in Section 4.3.1. Note that an error term is included during the re-encryption process, so the noise grows on each hop. Nevertheless, the correctness conditions presented before guarantee that the decryption

succeeds with overwhelming probability for proper parameters (which include the devised maximum number of hops, N). See also the results presented on Section 5.3.

- Not collusion-safe: This scheme suffers from the same vulnerability as NTRUReEncrypt, so secret keys can be extracted from the re-encryption key if the proxy colludes with a user involved. This problem is common in bidirectional proxy re-encryption schemes.

Table 3 shows the computational costs associated to PS-NTRUReEncrypt. As for our first scheme NTRUReEncrypt, the main operation here is the multiplication of polynomials, which is denoted by t_m. In addition, another potentially costly operation is sampling from the noise distribution, denoted by t_s. We distinguish here between on-line and off-line operations, as the latter can be performed beforehand (e.g., sampling noise terms).

Table 3: Time costs of PS-NTRUReEncrypt

Operation	On-line	Off-line
Encryption	t_m	$t_m + 2t_s$
Re-Encryption	t_m	$t_m + t_s$
Decryption	t_m	–

With regard to the theoretical space costs, this scheme shares the same results as NTRUReEncrypt, presented in Table 1. However, when considering an experimental instantiation, the parameters used would not be the same, which implies that the experimental costs vary. In addition, given that we lack of a formal analysis for quantifying the level of security that is provided by the scheme, it is not possible to make a direct comparison. An experimental analysis of the costs of PS-NTRUReEncrypt is presented in Section 5.3.

5. SOME EXPERIMENTAL RESULTS

In order to validate the viability of the proposed proxy re-encryption schemes, we have developed and tested an implementation of our proposals. Since we propose two schemes, we have two different implementations. NTRUReEncrypt is implemented on top of an available open-source Java implementation of NTRU [1]. For PS-NTRUReEncrypt, due to its non-standard nature, we had to code it from scratch; however, we made use of the functionalities provided by the Java Lattice-Based Cryptography (jLBC) library [14]. Our execution enviroment consists on an Intel Core 2 Duo @ 2.66 GHz. Although this processor has two cores, our implementation only uses one of them, as it is not parallel.

5.1 Performance of NTRUReEncrypt

The first experiment consisted of measuring the performance of the first proposed scheme using different set of parameters. We have used the parameter sets proposed in [28], although other parameters could be studied. Besides the basic implementation, we have made also some optimizations, such as the possibility of using product-form polynomials [28], which has a great effect on performance.

In Table 4, we give the execution times for the encryption, decryption and re-encryption for each parameter set. Parameters are represented by a tuple (*n, prod, k*), where *prod* is a boolean parameter that indicates the use of product-form polynomials, and k is the claimed security level (in bits). Additional parameters are fixed, such as $q = 2048$ and $p = 3$.

For each set of parameters, the experiment consisted of a looped execution, where a random message from the plaintext space is sequentially encrypted, re-encrypted and decrypted, and new keys are used on each iteration. The time for each operation is computed as the average of 100 iterations. Since sampling random elements is an operation that can be performed off-line, we decided to exclude it from the time measurements; thus, times reflected in this experiment only reflect the operations that must be performed on-line. This approach was also followed by [19].

Table 4: Computation time (in ms) and number of hops of NTRUReEncrypt for different parameters

Parameters	Enc.	Dec.	Re-Enc.	# Hops
(439, no, 128)	0.64	0.30	0.24	5
(439, yes, 128)	0.16	0.30	0.23	5
(1087, no, 256)	1.39	1.25	1.05	21
(1087, yes, 256)	0.48	1.26	1.07	15
(1171, no, 256)	0.80	1.12	1.14	21
(1171, yes, 256)	0.43	1.22	1.15	14
(1499, no, 256)	0.74	1.78	1.73	50
(1499, yes, 256)	0.32	1.67	1.66	42

For example, the sixth set of parameters was already used for a previous NTRU benchmark [19], and we can see that our results are consistent with their study, where they obtained a measure of approximately 0.31 ms for encryption (3220 encryptions per second).

In addition, Table 4 also shows the average number of re-encryptions supported by each parameter set. As stated in Section 3.2.2, the inclusion of the error term *pe* during the re-encryption process produces an increasing error that grows on each hop, causing a decryption failure at some point. It can be seen that this depends on the choice of parameters, so increasing the parameters actually decreases the probability of decryption failure.

5.2 Comparison of NTRUReEncrypt with other proxy re-encryption schemes

In order to benchmark our proposal with respect to other proxy re-encryption schemes, we have implemented three schemes. Two of them share similar properties than ours (i.e., bidirectional and multihop). One of them is the BBS scheme [11], which is CPA-secure as ours, while the other is the one proposed by Weng et al [27], which is proven CCA-secure; note, however, that the latter scheme achieves a better notion of security than ours, and hence, the comparison is unbalanced in this case. Both of them are implemented in Java using elliptic curve cryptography over a prime field. Specifically, we used the NIST P-256 curve, which provides 128 bits of security [10]. In addition, and for the sake of comparison, we have also implemented a third scheme from Aono et al [4]; note that in this case, this scheme is not bidirectional. For our scheme, we have used the *ees1171ep1* parameter set from [28], which achieves 256 bits of security. This parameter set corresponds to the sixth row of Table 4. We chose this parameter set because it was already used for a previous NTRU benchmark [19]. With regard to the experimental setting, we have used the same environment as before.

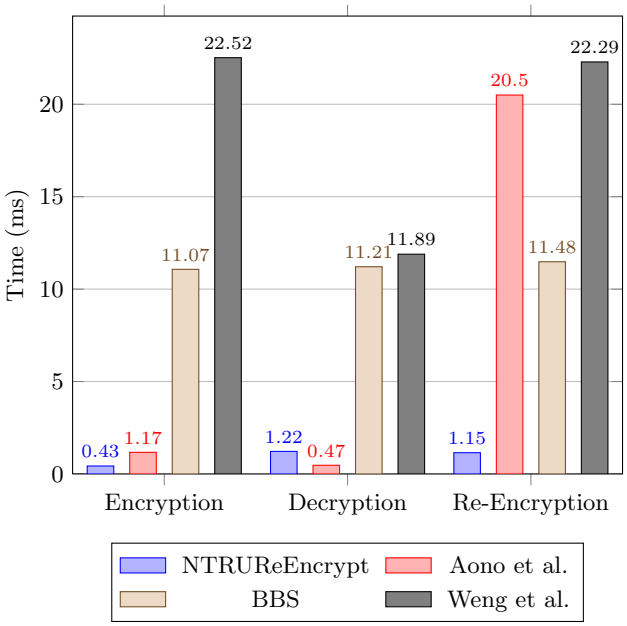

Figure 1: Comparison with other proxy re-encryption schemes

Figure 1 shows the results of our experiment. It can be seen that our scheme outperforms the others, as it is between 10 and 20 times faster. The scheme from Aono et al shows a similar performance in encryption and decryption, but is 20 times slower for re-encryption. The results obtained for our scheme are consistent with previous benchmarks for NTRU [19], as we used the same parameters.

For our experiment, we also implemented other unidirectional schemes, specifically, the ones from Ateniese et al [7] and Libert and Vergnaud [21]. However, we exclude them from this comparison since they are much slower due to the use of bilinear pairings; Appendix B presents these omitted results.

5.3 Performance of PS-NTRUReEncrypt

Table 5 shows the computation time of the main operations of PS-NTRUReEncrypt. It can be seen that the computation time for the main operations is approximately the same for each parameter set. This is also shown in Figure 2. Note that this figure is in logarithm scale, so the growth in computation time is exponential as n increases. This is a consequence of requiring that n is a power of 2, which restricts the choice of parameters.

Table 5: Computation time (in ms) of PS-NTRUReEncrypt for different parameters

Parameters	Enc.	Dec.	Re-Enc.
$n = 32, \log_2 q = 23$	0.93	0.99	1.05
$n = 64, \log_2 q = 28$	4.53	4.23	4.32
$n = 128, \log_2 q = 32$	17.28	17.32	17.45
$n = 256, \log_2 q = 37$	80.64	81.045	86.56
$n = 512, \log_2 q = 41$	333.75	334.07	359.54
$n = 1024, \log_2 q = 46$	1333.03	1344.10	1461.46

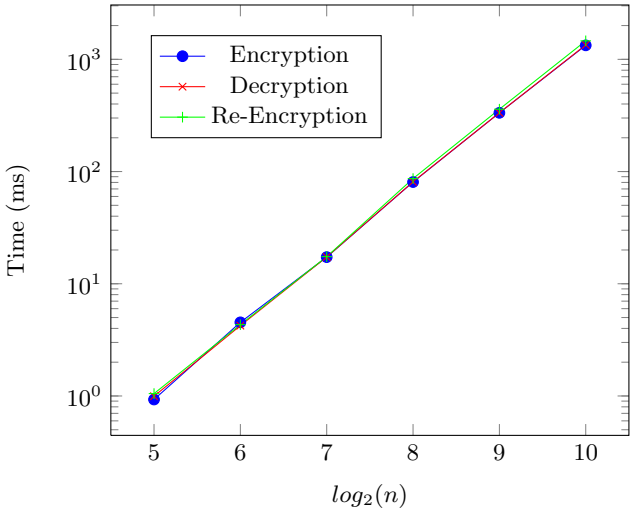

Figure 2: Performance of PS-NTRUReEncrypt

Unlike the first scheme, we do not provide a comparison of PS-NTRUReEncrypt to other proxy re-encryption schemes. We cannot directly compare it to other schemes without properly analizing the security level it achieves, which is done indirectly through evaluating the security against the best known lattice attacks [26].

With regard to the number of hops of this scheme, the results now are much better than in NTRUReEncrypt, where we had just up to 50 re-encryptions, with the parameter sets used. In this case, we were unable to find a maximum number of hops, having executed more than 10000 re-encryptions for each parameter set, without finding any decryption failure. This lead us to think that the parameters used in PS-NTRUReEncrypt are actually very conservative, which explains also that its performance is lower than NTRUReEncrypt. These parameters are, however, derived from the theoretic assumptions; it is an open problem to find ways to decrease them.

Other factor that could impact the results is that the implementation is not as optimized as NTRUReEncrypt, since we had to code it from scratch. In order to improve our implementation, we can make use of optimizations such as faster algorithms for polynomial multiplication, based on the Fast Fourier Transform (FFT).

Table 6: Space costs (in KB) of PS-NTRUReEncrypt for different parameters

Parameters	Size of polynomials
$n = 32, \log_2 q = 23$	0.09
$n = 64, \log_2 q = 28$	0.22
$n = 128, \log_2 q = 32$	0.50
$n = 256, \log_2 q = 37$	1.16
$n = 512, \log_2 q = 41$	2.56
$n = 1024, \log_2 q = 46$	5.75

As for the space costs, Table 6 shows the size in KB of the polynomials produced by different parameters. Recall that in our schemes, all the elements of the cryptosystem (i.e., public and secret keys, re-encryption keys, and ciphertexts) are represented by the same kind of polynomials.

6. CONCLUSIONS

In this paper we describe NTRUReEncrypt, a highly-efficient proxy re-encryption scheme based on the NTRU cryptosystem. This scheme is bidirectional and multihop, but not collusion-resistant. The key strength of this scheme is its performance. Experimental results show that this scheme outperforms previous proposals by an order of magnitude, and there is room for even more improvement, for instance using parallelization techniques, as shown in [19]. We believe that the level of efficiency shown by NTRUReEncrypt opens up new practical applications of proxy re-encryption in constrained environments. In addition to this scheme, we propose PS-NTRUReEncrypt, a provably-secure variant that is CPA-secure under the hardness of lattice problems, namely the Ring-LWE problem.

With regard to future work, there are several areas with potential for improvement. The most pressing are achieving CCA-security and the definition of a unidirectional and collision-resistant scheme. Another subject is improving the parameters of NTRUReEncrypt, since this could decrease the probability of decryption failures after multiple re-encryptions. Additionaly, it would be interesting to come up with better bounds for the provably-secure version and an analysis of the selection of parameters based on the best known attacks. Finally, we want also to extend the experimental analysis to other lattice-based proxy re-encryption schemes.

7. ACKNOWLEDGMENTS

This work was partly supported by the Junta de Andalucía through the project FISICCO (P11-TIC-07223) and by the Spanish Ministry of Economy and Competitiveness through the PERSIST project (TIN2013-41739-R). The first author has been funded by a FPI fellowship from the Junta de Andalucía through the project PISCIS (P10-TIC-06334).

8. REFERENCES

[1] Java implementation of NTRUEncrypt and NTRUSign. http://tbuktu.github.io/ntru/.

[2] ANSI X9.98: Lattice-based polynomial public key establishment algorithm for the financial services industry. Technical report, ANSI, 2010.

[3] C. Aguilar-Melchor and P. Gaborit. A lattice-based computationally-efficient private information retrieval protocol. *Cryptol. ePrint Arch., Report*, 446, 2007.

[4] Y. Aono, X. Boyen, L. T. Phong, and L. Wang. Key-private proxy re-encryption under LWE. In *Progress in Cryptology–INDOCRYPT 2013*, pages 1–18. Springer, 2013.

[5] G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. *Topics in Cryptology–CT-RSA 2009*, pages 279–294, 2009.

[6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, pages 29–44, 2005.

[7] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[8] D. V. Bailey, D. Coffin, A. Elbirt, J. H. Silverman, and A. D. Woodbury. NTRU in constrained devices. In *Cryptographic Hardware and Embedded Systems—CHES 2001*, pages 262–272. Springer, 2001.

[9] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for key management — part 1: General. Technical report, 2005.

[10] E. Barker, L. Chen, A. Roginsky, and M. Smid. Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. NIST special publication 800-56A (Revision 2), NIST, May 2013.

[11] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. *Advances in Cryptology—EUROCRYPT'98*, pages 127–144, 1998.

[12] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 185–194. ACM, 2007.

[13] C. Chu and W. Tzeng. Identity-based proxy re-encryption without random oracles. *Information Security*, pages 189–202, 2007.

[14] A. De Caro. Java Lattice Based Cryptography Library (jLBC). http://gas.dia.unisa.it/projects/jlbc/.

[15] A. De Caro and V. Iovino. jPBC: Java pairing based cryptography. In *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, pages 850–855. IEEE, 2011.

[16] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[17] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[18] M. Green and G. Ateniese. Identity-based proxy re-encryption. In *Applied Cryptography and Network Security*, pages 288–306. Springer, 2007.

[19] J. Hermans, F. Vercauteren, and B. Preneel. Speed records for ntru. In *Topics in Cryptology-CT-RSA 2010*, pages 73–88. Springer, 2010.

[20] J. Hoffstein, J. Pipher, and J. H. Silverman. Ntru: A ring-based public key cryptosystem. In *Algorithmic number theory*, pages 267–288. Springer, 1998.

[21] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *Information Theory, IEEE Transactions on*, 57(3):1786–1802, 2011.

[22] A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th symposium on Theory of Computing*, pages 1219–1234. ACM, 2012.

[23] X. Lv, B. Yang, and C. Pei. Efficient traitor tracing scheme based on ntru. In *Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005. Sixth International Conference on*, pages 120–124. IEEE, 2005.

[24] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):43, 2013.

[25] D. Stehlé and R. Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *Advances in Cryptology–EUROCRYPT 2011*, pages 27–47. Springer, 2011.

[26] D. Stehlé and R. Steinfeld. Making NTRUEncrypt and NTRUSign as secure as standard worst-case problems over ideal lattices. *IACR Cryptology ePrint Archive*, 2013:4, 2013.

[27] J. Weng, R. H. Deng, S. Liu, and K. Chen. Chosen-ciphertext secure bidirectional proxy re-encryption schemes without pairings. *Information Sciences*, 180(24):5077–5089, 2010.

[28] W. Whyte, N. Howgrave-Graham, J. Hoffstein, J. Pipher, J. Silverman, and P. Hirschhorn. IEEE P1363.1: Draft standard for public-key cryptographic techniques based on hard problems over lattices. Technical report, IEEE, 2008.

[29] K. Xagawa and K. Tanaka. Proxy re-encryption based on learning with errors. In *Proceedings of the 2010 Symposium on Cryptography and Information Security (SCIS 2010)*, 2010.

APPENDIX

A. SECONDARY LEMMAS

In this section we present two secondary lemmas, both presented in [26]. The first lemma provides bounds for the secret keys generated using the key generation algorithm:

LEMMA 1. *[26, Lemma 3.6] Let n be a power of 2, q a prime so that $q = 1 \mod 2n$, and $deg(p) \leq 1$. Let $\sigma \geq \sqrt{n \log n} \cdot q^{1/n}$. The secret key polynomials f and g returned by the KeyGen algorithm satisfy $\|f\| \leq 4\sqrt{n}\|p\|\sigma$ and $\|g\| \leq \sqrt{n}\sigma$, with probability $\geq 1 - 2^{-n+3}$.*

The second lemma provides bounds to the product of arbitrary polynomials of $\mathcal{R}$ with samples from a distribution from Ψ_α:

LEMMA 2. *[26, Lemma 2.10] Let $y, r \in \mathcal{R}$, with r fixed and y sampled from a distribution from Ψ_α, with $\alpha q \geq n^{0.25}$. Then: $Pr\left[\|yr\|_\infty \geq \alpha q n^{-0.25} \omega(\log n) \cdot \|r\|\right] \leq n^{-\omega(1)}$*

B. PERFORMANCE EVALUATION OF PROXY RE-ENCRYPTION SCHEMES

In addition to the schemes shown in the experimental comparison, we also implemented two more schemes from Ateniese et al [7] and Libert and Vergnaud [21], both of them unidirectional and the latter secure against chosen ciphertext attacks. However, being based on pairings, these schemes have proven to be much slower, so we drop them from our main analysis. Since we are unaware of any previous work that benchmarks actual implementations of proxy re-encryption schemes, we decided to include this evaluation as an appendix.

We implemented these additional schemes using the jPBC library [15], a pairing-based cryptography library for Java. As for the cryptographic details, we used a supersingular curve of the form $y^2 = x^3 + x$, with a 256-bit group order and 3072 bits for the field size, which achieves 128 bits of security [16][9]. Additionally, we have made extensive use of exponentiation and pairing preprocessing of frequently-used elements for efficiency reasons. This is a very useful functionality of jPBC that allows the preprocessing of exponentiation operations for efficiency reasons, since there are several elements that are frequently exponentiated, such as the generators of the groups and the public keys.

Table 7: Computation time of several proxy re-encryption schemes (in ms)

Scheme	Enc.	Dec.	Re-Enc.
NTRUReEncrypt	0.43	1.22	1.15
Aono et al [4]	1.17	0.47	20.5
BBS [11]	11.07	11.21	11.48
Weng et al [27]	22.52	11.89	22.29
Ateniese et al [7]	22.76	13.76	83.52
Libert and Vergnaud [21]	155.27	443.87	386.93

Table 7 shows the results obtained from our experiments, including the schemes from Ateniese et al. and from Libert and Vergnaud. For the particular case of re-encryption, which is the most common operation for most applications of proxy re-encryption, it can be seen that NTRUReEncrypt outperforms the others. Note also that the security level for the proxy re-encryption schemes is 128 bits, while the provided by NTRUReEncrypt with the used parameters is 256 bits; if we increase the parameters in the rest of the schemes in order to achieve 256 bits of security, then these times would be much higher.