Blending FHE-NTRU keys – The Excalibur Property

Louis Goubin and Francisco José Vial Prado

Laboratoire de Mathématiques de Versailles
UVSQ, CNRS, Université Paris-Saclay
78035 Versailles, France

Abstract. Can Bob give Alice his decryption secret and be convinced that she will not give it to someone else? This is achieved by a proxy re-encryption scheme where Alice does not have Bob's secret but instead she can transform ciphertexts in order to decrypt them with her own key. In this article, we answer this question in a different perspective, relying on a property that can be found in the well-known modified NTRU encryption scheme. We show how parties can collaborate to *one-way-glue* their secret-keys together, giving Alice's secret-key the additional ability to decrypt Bob's ciphertexts. The main advantage is that the protocols we propose can be plugged directly to the modified NTRU scheme with no post-key-generation space or time costs, nor any modification of ciphertexts. In addition, this property translates to the NTRU-based multikey homomorphic scheme, allowing to equip a hierarchic chain of users with automatic re-encryption of messages and supporting homomorphic operations of ciphertexts. To achieve this, we propose two-party computation protocols in cyclotomic polynomial rings. We base the security in presence of various types of adversaries on the RLWE and DSPR assumptions, and on two new problems in the modified NTRU ring.

1 Introduction

Is it possible to avoid betrayal in a hierarchic scenario? Imagine a chain of users equipped with a public-key encryption scheme, where high level users can decrypt ciphertexts intended to all lower level users in the chain. This is trivial to construct using any public-key cryptosystem $\mathcal{E}$: just transfer low-level secret-keys to upper levels following the hierarchy. The evident drawback is that high-level users can betray their children and distribute their secrets to other parties. Using a proxy re-encryption procedure or multiple trapdoors is hence preferred, because parents do not have direct knowledge of their children's secrets. A proxy re-encryption scheme is a cryptosystem that allows a public transformation of ciphertexts such that they become decryptable to an authorized party. This is a particular case of a cryptosystem allowing delegation of decryption, which finds applications in mail redirection, for instance. In this article, we give a solution to the betrayal issue in another perspective, relying on a new property we found in the well-known modified NTRU encryption scheme, and which we refer to as

"Excalibur". Basically, this feature allows to generate a secret-key that decrypts encryptions under multiple public-keys and behaves like a regular key of the cryptosystem.

1.1 The Excalibur Property

A public-key encryption scheme $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M}$ has the Excalibur property if there is an algorithm that allows two users Alice and Bob with key-pairs $(\mathsf{sk}_A^{old}, \mathsf{pk}_A^{old})$ and $(\mathsf{sk}_B, \mathsf{pk}_B)$ respectively to forge a new key-pair for Alice $(\mathsf{sk}_A, \mathsf{pk}_A)$ such that

- Alice's key sk_A can decrypt ciphertexts in $\mathsf{Enc}(\mathsf{pk}_A, \mathcal{M}) \cup \mathsf{Enc}(\mathsf{pk}_B, \mathcal{M})$.
- Bob cannot decrypt ciphertexts in $\mathsf{Enc}(\mathsf{pk}_A, \mathcal{M})$.
- Alice cannot generate a secret-key sk'_B that is able to decrypt ciphertexts in $\mathsf{Enc}(\mathsf{pk}_B, \mathcal{M})$ but is not able to decrypt ciphertexts in $\mathsf{Enc}(\mathsf{pk}_A, \mathcal{M})$ (i.e. she cannot give away access to Bob's secret without leaking her own).

The intuition is that sk_A is a one-way expression of $(\mathsf{sk}_A^{old}, \mathsf{sk}_B)$. As Alice owns decryption rights over Bob's ciphertexts, this can be seen as *automatic* proxy re-encryption, in the sense that the re-encryption procedure is the identity. The idea is to "glue" Alice and Bob secret-keys together, resulting on a master key given to Alice. This Excalibur master key can be separated into factors only by Bob, hence the name of the feature: Bob plays the role of young Arthur, who is the only man in the kingdom able to separate Excalibur from the stone. Moreover, Alice can glue her key to an upper user's key, who inherits decryption over Bob's ciphertexts, and so forth, and if we suppose that no user is willing to give away own secrets, this achieves automatic N–hop re-encryption and sets a hierarchic chain.

We therefore have a scheme in which a single private-key can decrypt messages under multiple public-keys, and we will see that if a group of low-level users cheated in the joint key generation of this private-key (in order to sabotage or harden decryption), the secret-key holder may be able to trace it back to the wrongdoers, by simply testing decryptions and looking at the private-key's coefficients. In a sense, this is the inverse setting of a public-key traitor tracing scheme, where there are multiple secret-keys associated with a single public-key, and such that if a group of users collude in creating a new private-key achieving decryption with the public-key, it is possible to trace it to its creators, see for instance [4].

Three main advantages of this property over the trivial transfer of keys, over re-encryption schemes and over multiple trapdoor schemes are (i) there are no extra space or time costs: as soon as the keys are blended, the resulting key-pair acts as a fresh one and no ciphertext modification is necessary, (ii) our key generation procedure can be plugged directly into the (multikey) NTRU-based fully homomorphic encryption scheme, supporting homomorphic operations and automatic N-hop re-encryption and (iii) a user with a powerful key does not need to handle a "key ring" of secret-keys of her children; her key-pair $(\mathsf{sk}, \mathsf{pk})$ acts

as a regular NTRU key. In contrast, the classical proxy re-encryption scenario is more flexible; a user can agree a decryption delegation at any moment to any user, whereas in our proposal once the keys are blended, modifications in hierarchy involve new key generations. This is why our proposal is more suitable to a rigid pre-defined hierarchic scenario.

1.2 Modified NTRU

The NTRUEncrypt cryptosystem is a public-key encryption scheme whose security is based on short vector problems on lattices. Keys and ciphertexts are elements of the polynomial ring $\mathbb{Z}[X]/\langle\phi(x)\rangle$ where $\phi(x) = x^n - 1$, and coefficients are considered modulo a large prime q. This scheme was defined in 1996 by Hoffstein, Pipher, Silverman and gained much attention since its proposal because of its efficiency and hardness reductions. In [25], Stehlé and Steinfeld provided modifications to the scheme in order to give formal statistic proofs, which ultimately led to support homomorphic operations with an additional assumption in [23]. Among these modifications, we highlight the change of ring and parameters restrictions: $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$ where now $\phi(x) = x^n + 1$, n is a power of 2 (hence ϕ is the $2n$-th cyclotomic polynomial), and the large prime modulus is such that $x^n + 1$ splits into n different factors over $\mathbb{F}_q$ (namely, $q = 1 \mod 2n$). We will consider the modified NTRU scheme, but we believe that, possibly via a stretching of parameters, the original NTRU may also exhibit the Excalibur property.

1.3 Excalibur key generation

The way to glue two secret-keys is very simple: just multiply them together ! Indeed, the modified NTRU scheme offers a fruitful property: *If one replaces a secret-key with a small polynomial multiple of it, decryption still works.* If this polynomial multiple is itself a secret-key, then by symmetry decryption with the resulting key will be correct in the union of ciphersets decryptable by one key or another. However, addressing the main point of this article, parties must multiply the involved polynomials using multiparty protocols, since they do not want to trust individual secrets to each other. To achieve this joint key generation, we rely on multiparty protocols in the polynomial ring $R_q = \mathbb{F}_q[x]/(x^n + 1)$ in both the secret and shared setting. To this end, we describe two multiplication protocols between mutually distrusting Alice and Bob:

1. **Secret inputs setting** : Alice and Bob hold $f, g \in R_q$ respectively. They exchange random polynomials and at the end Alice learns $fg + r \in R_q$ where r is a random polynomial known by Bob, and Bob learns nothing.
2. **Additively shared inputs setting** : Alice and Bob hold $f_A, g_A \in R_q$ and $f_B, g_B \in R_q$ respectively such that $f = f_A + f_B$ and $g = g_A + g_B$. They exchange some random polynomials, and at the end Alice and Bob learn π_A, π_B respectively such that $\pi_A + \pi_B = fg \in R_q$. Revealing π_A or π_B to each other does not leak information about the input shares.

Let us illustrate how to use these protocols in Alice's key generation. Suppose that Bob keys were previously generated. Generating Alice's secret-key is fairly easy: Informally, if $\beta \in R_q$ is Bob's secret-key, let Alice and Bob sample random $\alpha_A, \alpha_B \in R_q$ respectively, with small coefficients. They perform the first protocol on inputs $f = \alpha_A$ and $g = \beta$, and Bob chooses $r = \alpha_B\beta$. At the end, Alice learns $\gamma = \alpha_A\beta + \alpha_B\beta = \alpha\beta \in R_q$, and Bob learns nothing. One may stop here and let Alice compute her public-key $\mathsf{pk}_A = 2h\gamma^{-1} \in R_q$ for suitable $h \in R_q$, but she may cheat and generate other NTRU fresh keys $(\mathsf{sk}_A, \mathsf{pk}'_A)$ and then distribute freely Bob's secret γ. This is why the public-key is also generated jointly, and moreover, the public-key will be generated before the secret-key, this way Alice must first commit to a public-key pk_A.

1.4 Fully Homomorphic Encryption

Fully Homomorphic Encryption schemes allow public processing of encrypted data. Since Gentry's breakthrough in [10–12], there has been considerable effort to propose FHE schemes that are efficient [1, 2, 7, 14–18, 20], secure [2, 6, 8, 9, 13], and having other properties [7, 9, 13, 19]. We highlight the existence of Multi-key FHE schemes, in which some ciphertexts can only be decrypted with the collaboration of multiple key-holders. This was first constructed in [23], and it reduces the general multiparty computation problem to a particular instance. We encourage the reader to see the latest version of this article.

All of the above schemes have a PPT encryption algorithm that adds random "noise" to the ciphertext, and propose methods to add and multiply two ciphertexts. With these methods they give an (homomorphic) evaluation algorithm of circuits. The noise in ciphertexts grows with homomorphic operations (especially with multiplication gates) and after it reaches a threshold, the ciphertext can no longer be decrypted. Thus, only circuits of bounded multiplicative degree can be evaluated: these schemes are referred to as leveled FHE schemes. Gentry proposed a technique called "bootstrapping" that transform a ciphertext into one of smaller noise that encrypts the same message, therefore allowing more homomorphic computations. This (algorithmically expensive) technique remains the only known way to achieve pure FHE scheme from a leveled FHE scheme. In order to do this, the decryption circuit of the leveled scheme must be of permitted depth and the new scheme relies on non-standard assumptions.

Nevertheless, leveled FHE schemes with good *a priori* bounds on the multiplicative depth do satisfy most applications requirements, see [22,27]. We suggest that the use of our protocols in the LATV scheme use the leveled version, but as pointed out in [23], the scheme can be transformed into a fully homomorphic scheme by boostrapping and modulus reduction techniques, both adaptable to the use of Excalibur keys.

1.5 FHE and bidirectional multi-hop re-encryption paradigm

It has been widely mentioned (for instance in the seminal work [11]) that a fully homomorphic encryption scheme allows bidirectional multi-hop proxy re-

encryption. The argument is similar to the celebrated bootstrapping procedure: let c be an encryption of m using Bob's secret-key s_B. First publish τ, an encryption of s_B under Alice's public-key, then homomorphically run the decryption circuit on c and τ, the result is an encryption of m decryptable by Alice's secret-key. However, we point out that this is *pure* re-encryption only if Alice never gets access to τ, since she can decrypt and learn s_B directly. This restriction tackles the pure re-encryption definition, and in light of this the NTRU-based FHE scheme with the Excalibur property may be a starting point to clear out this paradigm (as it satisfies the pure definition, but fails to be bidirectional).

1.6 Our contributions

In this article, we propose a key generation protocol that allows to glue NTRU secret-keys together in order to equip a hierarchic chain of users, such that a given user has the ability to decrypt all ciphertexts intended to all lower users in the chain, and she cannot give away secrets without exposing her own secret-key. This procedure can be plugged directly into the (multikey) FHE-scheme by Lopez-Alt et.al., it is compatible with homomorphic operations and has no space costs or ciphertext transformations, and important users do not have to handle key rings. To achieve this, we describe two-party computations protocols in cyclotomic polynomial rings that may be of independent interest. We base the semantic security on the hardness of RLWE and DSPR problems, and the semi-honest and malicious security in a new hardness assumption which we call "Small Factors Assumption". In this assumption we define the "Small GCD Problem" and we show that any algorithm solving this problem can be used to break the semantic security of the modified NTRU scheme.

2 Preliminaries

2.1 Notation

Let q be a large prime. We let the set $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$ represent the equivalence classes of $\mathbb{Z}/q\mathbb{Z}$, and both notations $[x]_q$ or $x \mod q$ represent modular reduction of x into this set. For a ring A, $A^\times$ stands for the group of units (or invertible elements) of A, $\langle a \rangle$ or (a) is the ideal generated by $a \in A$. Also, we denote by $\mathbb{F}_k$ the finite field of k elements, for $k = q^l \in \mathbb{Z}$. The notation $e \leftarrow \xi$ indicates that the element e is sampled according to the distribution ξ, and $e \xleftarrow{R} S$ means that e was sampled from the set S using the uniform distribution. Similarly, $A \overset{R}{\subset} S$ means that each $a \in A$ was sampled uniformly at random on S. Finally, let $R \overset{def}{=} \mathbb{Z}[x]/(x^n + 1)$, we identify an element of R with its coefficient vector in $\mathbb{Z}^n$, and for $v(x) = v_0 + v_1 x + \cdots + v_{n-1} x^{n-1}$ in R, we denote by $||v||_\infty, ||v||_2$ its l_∞, l_2 norm respectively.

2.2 The quotient ring R_q

Operations in the modified NTRU scheme are between elements of $R_q \stackrel{def}{=} \mathbb{F}_q[x]/(x^n+1)$, the ring of polynomials modulo $\Phi_{2n}(x) = x^n + 1$ (i.e. Φ_{2n} is the $2n$–th cyclotomic polynomial) and coefficients in $\mathbb{F}_q$, where n is a power of 2 and q is a large prime. Addition and multiplication of polynomials are performed modulo $\Phi_{2n}(x)$ and modulo q. The ring R_q is not a unique factorization domain, in fact, small units of this ring serve as NTRU secret-keys. The Chinese remainder theorem shows that the group of units is large, and thus $y = ru \in R_q$ where $r \in R_q$ is a random element and u is a unit is a good masking of u: it is unfeasible to recover u from y for large n. Let us collect some lemmas related to the set of invertible elements of R_q.

Lemma 2.2.1 *Let $q \geq 3$ be a prime number and $\Phi_n(x) \in \mathbb{Z}[x]$ be the n–th cyclotomic polynomial. Then $\Phi_n(x)$ is irreducible over $\mathbb{F}_q$ if and only if q is a generator of the group $(\mathbb{Z}/n\mathbb{Z})^\times$.*

Lemma 2.2.2 *If $n > 2$ is a power of 2, then $(\mathbb{Z}/2n\mathbb{Z})^\times$ is not cyclic and therefore $\Phi_{2n}(x) = x^n + 1$ is not irreducible over $\mathbb{F}_q$. In addition, $x^n + 1$ decomposes into l distinct irreducible factors over $\mathbb{F}_q$ for prime $q \geq 3$: Let $(\phi_i)_{i=1}^l \subset \mathbb{F}_q[x]$ respectively such that $x^n + 1 = \prod_{i=1}^l \phi_i(x)$ over $\mathbb{F}_q$. Then we have a ring isomorphism*

$$\pi : \frac{\mathbb{F}_q[x]}{(x^n+1)} \to \prod_{i=1}^{l} \frac{\mathbb{F}_q[x]}{(\phi_i(x))} \text{ where } \frac{\mathbb{F}_q[x]}{(\phi_i(x))} \simeq \mathbb{F}_{q^{\deg \phi_i}}.$$

Corollary 2.2.3 $\mathrm{Card}(R_q^\times) = \prod_{i=1}^l \left(q^{\deg \phi_i} - 1\right)$.

The proofs are straightforward. In the original modifications in [25], $q = 1 \mod 2n$ and hence x^n+1 splits into n distinct linear factors, yielding $\mathrm{Card}(R_q)^\times = (q-1)^n$.

2.3 Bounded Gaussian samplings on $\mathbb{Z}[x]/(x^n+1)$

Let n be a power of 2 and q a prime number, $R = R_0 \stackrel{def}{=} \frac{\mathbb{Z}[x]}{(x^n+1)}$ and as before $R_q \stackrel{def}{=} \frac{\mathbb{F}_q[x]}{(x^n+1)}$. The modified NTRU scheme uses a particular distribution in R_q, which we refer to as K-bounded by rejection Gaussian, serving to sample both message noises and secret-keys. Definitions follow.

Definition 2.3.1 *Let $\mathcal{G}_r$ be the Gaussian distribution over R, centered about 0 and of standard deviation r.*

Sampling from $\mathcal{G}_r$ can be done in polynomial time, for instance approximating with Irwin-Hall distributions. Consider the following definitions from [23]:

Definition 2.3.2 *A polynomial $e \in R$ is called K-bounded if $||e||_\infty < K$.*

Definition 2.3.3 *A distribution is called K-bounded over R if it outputs a K-bounded polynomial.*

Definition 2.3.4 *(K-bounded by rejection Gaussian) Let $\bar{\mathcal{G}}_K$ be the distribution $\mathcal{G}_{K/\sqrt{n}}$ that repeats sampling if the output is not K-bounded.*

Lemma 2.3.5 (Expansion factors for $\phi(x) = x^n + 1$, from [23]) *For any polynomials $s, t \in R$,*

$$||s \cdot t \mod \phi(x)||_2 \leq \sqrt{n} \cdot ||s||_2 \cdot ||t||_2,$$
$$||s \cdot t \mod \phi(x)||_\infty \leq n \cdot ||s||_\infty \cdot ||t||_\infty.$$

Corollary 2.3.6 *Let χ be a K-bounded distribution over R and let $s_1, \ldots, s_l \leftarrow \chi$. Then $\prod_{i=1}^{l} s_i$ is $(n^{l-1}K^l)$-bounded.*

3 Modified NTRU encryption

We review the modified NTRU encryption scheme as presented in [23], and we insist on the multi-key property. The message space is $\{0, 1\}$ and the ciphertext space is $R_q = \frac{\mathbb{F}_q[x]}{(x^n+1)}$. Let q be a large prime, $0 < K \ll q$, n be a power of 2 and $\bar{\mathcal{G}}_K$ be the K-bounded by rejection discrete Gaussian. A key-pair $(\mathsf{sk}, \mathsf{pk})$ is a tuple of polynomials in R_q, the secret-key being K-bounded.

$\underline{\mathsf{Keygen}(1^\kappa)}$:

Step 1. Sample a polynomial $f \leftarrow \bar{\mathcal{G}}_K$. Set $\mathsf{sk} = 2f + 1$, if sk is not invertible in R_q start again.
Step 2. Sample a polynomial $g \leftarrow \bar{\mathcal{G}}_K$ and set $\mathsf{pk} = 2g \cdot \mathsf{sk}^{-1} \in R_q$.
Step 3. Output $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$: Sample polynomials $s, e \leftarrow \bar{\mathcal{G}}_K$. For message $m \in \{0, 1\}$, output $c = m + 2e + s \cdot \mathsf{pk} \mod q$.

$\underline{\mathsf{Dec}(\mathsf{sk}, c)}$: For a ciphertext $c \in R_q$, compute $\mu = c \cdot \mathsf{sk} \in R_q$ and output $m = \mu \mod 2$.

3.1 The multikey property

We describe a decryption property that states that one can decrypt a ciphertext with the secret-key required for decryption, or a small polynomial multiple of it.

Lemma 3.1.1 *Let $(f, h) \leftarrow \mathsf{Keygen}(1^\kappa)$, $m \in \{0, 1\}$ and let $c \leftarrow \mathsf{Enc}(h, m)$. Let $\theta \in R$ be a M-bounded polynomial satisfying $\theta \mod 2 = 1$. If $M < (1/72)(q/n^2K^2)$, then*

$$\mathsf{Dec}(f, c) = \mathsf{Dec}(\theta \cdot f, c) = m.$$

Proof: There exist K-bounded polynomials s, e such that $c = m + hs + 2e$. Decryption works since

$$[fc]_q = [fm + fhs + 2fe]_q = [fm + 2gs + 2fe]_q$$

and supposing there is no wrap-around modulo q in the latter expression, we have $[fc]_q \mod 2 = fc \mod 2 = m$. If we replace f by $\theta \cdot f$ and try to decrypt, we have $\theta fc = \theta fm + 2\theta gs + 2\theta fe$, and then again, if there is no wrap-around modulo q (i.e. if M is small enough), $\theta fc \bmod 2 = m$ is verified. To ensure that there is no wrap-around modulo q, one has to give an *a priori* relation between K, n and M. In fact, using corollary 2.3.6, we have $||gs||_\infty < nK^2$ and $||fe||_\infty < n(2K+1)K$, and thus

$$||fc||_\infty < 2nK^2 + 2n(2K+1)K + K.$$

Decryption using f is correct if $2nK^2 + 2n(2K+1)K + K < q/2$, and decryption using θf is correct if $nM(2nK^2 + 2n(2K+1)K + K) < q/2$. Therefore, decryption using f is ensured by $36nK^2 < q/2$, decryption using θf is ensured by $36n^2MK^2 < q/2$. □

Corollary 3.1.2 (The multikey property) *Let (f_1, h_1) and (f_2, h_2) be valid keys, $m_1, m_2 \in \{0,1\}$ and let $c_1 \leftarrow \mathsf{Enc}(h_1, m_1)$, $c_2 \leftarrow \mathsf{Enc}(h_2, m_2)$. Let $\tilde{f} \leftarrow f_1 \cdot f_2 \in R_q$. Then*

$$\mathsf{Dec}(\tilde{f}, c_1) = m_1,\ \mathsf{Dec}(\tilde{f}, c_2) = m_2$$

provided that K is small enough,

Proof: Apply 3.1.1 with $f = f_1$ and $\theta = f_2$ for the first equation and $f = f_2, \theta = f_1$ for the second. □

We can of course extend this facts to show that a highly composite key of the form $\tilde{f} = \prod_{i=1}^{l} f_i \in R_q$ can decrypt all messages decryptable by any of f_i: Just apply lemma 3.1.1 with $f = f_i$ and $\theta = \tilde{f}/f_i$, provided good *a priori* bounds: In fact $||\tilde{f}||_\infty \leq n^{l-1}K^l$, therefore decryption with this key is ensured by $n^{l-1}K^l \ll q$.

4 Hardness assumptions

The modified NTRU-FHE scheme semantic security is based on the celebrated *Ring Learning With Errors problem* (RLWE) and the new *Small Polynomial Ratio problem* (SPR). For the original modified NTRU parameters, the decisional SPR problem reduces to RLWE, but not a single homomorphic operation can be assured. A stretch of parameters is needed to overcome this, though it severely harms the statistic proofs of Stehlé and Steinfeld. The *DSPR assumption* states that the decisional SPR problem with stretched parameters is computationally hard. We adopt this same assumption, and in addition, we base the security of the honest-but-curious model on two problems that involve decomposing a polynomial into bounded factors. In the first, one wants to factorize a polynomial in R_q into two K-bounded polynomials, given the information that this is possible. In the second, one wants to extract a common factor of two polynomials such that the remaining factors are K-bounded. We first describe the DSPR assumption and then our *"Small Factors"* assumption.

4.1 Small Polynomial Ratio Problem, from [23]

In [25] Stehlé and Steinfeld based the security of the modified NTRU encryption scheme on the Ring Learning With Errors (RLWE) problem [24]. They showed that the public-key $\mathsf{pk} = 2g \cdot \mathsf{sk}^{-1} \in R_q$ is *statistically close to uniform* over R_q, given that g and $f' = (\mathsf{sk} - 1)/2$ were sampled using discrete Gaussians. Their results holds if (a) n is a power of 2, (b) $x^n + 1$ splits over n distinct factors over R_q (i.e. $q = 1 \mod 2n$) and (c) the Gaussian error distribution has standard deviation of at least $\text{poly}(n)\sqrt{q}$. However, these distributions seem too wide to support homomorphic operations in the NTRU-FHE scheme. To overcome this, authors in [23] defined an additional assumption which states that if the Gaussian is contracted, it is still hard to distinguish between a public-key and a random element of R_q (even if the statistic-closeness result does not hold).

Definition 4.1.1 (DSPR Assumption) *Let $q \in \mathbb{Z}$ be a prime integer and $\bar{\mathcal{G}}_K$ denote the K-bounded discrete Gaussian distribution over $R_0 = \mathbb{Z}[X]/(x^n + 1)$ as defined in 2.3.4. The decisional small polynomial ratio assumption says that it is hard to distinguish the following two distributions on R_q: (1) A polynomial $h = [2gf^{-1}]_q \in R_q$ where f', g were sampled with $\bar{\mathcal{G}}_K$ and $f = 2f'+1$ is invertible over R_q, and (2) a polynomial $u \xleftarrow{R} R_q$ sampled uniformly at random.*

Finally, in a work by Bos et.al. [5], authors achieved to base the security on RLWE alone, alas achieving multikey FHE for a constant number of keys, a property inherent to any FHE scheme (as proved in the latest version of [23]).

4.2 Small factorizations in the quotient ring

In addition to the RLWE and DSPR assumptions, we rely the semi-honest security on the hardness of the following problems. Let us define the distribution $\bar{\mathcal{G}}_K^{\times}$ which samples repeatedly from $\bar{\mathcal{G}}_K$ until the output is invertible over R_q.

Small Factors Problem: *Let $a, b \leftarrow \bar{\mathcal{G}}_K^{\times}$ and let $c(x) = a(x) \cdot b(x) \in R_q$. Find $a(x)$ and $b(x)$, given $c(x)$ and a test routine $T : R_q \to \{0, 1\}$ that outputs 1 if the input is in $\{a, b\}$ and 0 otherwise.*

$\bar{\mathcal{G}}_K^{\times}$–GCD Problem: *Let $a, b \leftarrow \bar{\mathcal{G}}_K^{\times}$, and $y \xleftarrow{R} R_q$. Let $u(x) = a(x) \cdot y(x) \in R_q$ and $v(x) = b(x) \cdot y(x) \in R_q$. Find $a(x), b(x)$ and $y(x)$, given $u(x), v(x)$ and a test routine $T : R_q \to \{0, 1\}$ that outputs 1 if the input is in $\{a, b, y\}$ and 0 otherwise.*

Proposition 4.2.1 *An algorithm solving the $\bar{\mathcal{G}}_K^{\times}$–GCD problem can be used to break the semantic security of the NTRU scheme.*

Proof: Given only a public-key of the form $\mathsf{pk} = [2ab^{-1}]_q$ where a is the secret-key, sample $p \xleftarrow{R} R_q$ and define $(u', v') = (ab^{-1}p, p)$. Define also $T : R_q \to \{0, 1\}$ that for input $\alpha \in R_q$, samples random $r \xleftarrow{R} \{0, 1\}$, checks if $\mathsf{Dec}(\alpha, \mathsf{Enc}(\mathsf{pk}, r)) \stackrel{?}{=}$

r and outputs 1 if α pass several such tests. Note that $u' = ay'$ and $v' = by'$ for $y = b'^{-1}p$, therefore seeding u', v', T to such algorithm outputs a, b, y'. □

Small Factors Assumption: *For the modified NTRU parameters, it is unfeasible to solve the small factors problem.*

In the absence of a formal proof, let us motivate the hardness of the small factors problem. The SF problem is equivalent to solve a quadratic system of equations over $\mathbb{F}_q$ with additional restrictions on the unknowns. Indeed, each coefficient of $c(x)$ is a quadratic form on coefficients of $a(x), b(x)$:

$$c_k = \sum_{i=0}^{n-1} a_i b_{k-i \bmod n} \cdot \sigma_k(i) \mod q,$$

where $\sigma_k(i) = +1$ if $i \leq k$ and -1 otherwise, the unknowns a_i, b_j follow a Gaussian distribution about 0 and are bounded in magnitude by K. As $K \ll q$, one can consider the equations over the integers. This results in a Diophantine quadratic system of n equations in $2n$ variables. Quadratic systems of m equations with n unknowns can be the Achilles heel for strong cryptographic primitives, as they can be attacked in the very overdetermined ($m \geq n(n-1)/2$) or very underdetermined ($n \geq m(m+1)$) cases in fields with even characteristic. In [26], authors adapt an algorithm of Kipnis-Patarin-Goubin [21] to odd characteristic fields and show a gradual change between the determined case ($m = n$, $\exp(m)$ runtime) and the massively underdetermined case ($n \geq m(m+1)$, $\text{poly}(n)$ runtime). According to their analysis, our system ($n = 2m$) escapes the polynomial-time scope. Let us write the system in clear:

$$\forall i \in \{0, \ldots, n-1\}, ||a_i||_\infty < K \text{ and } ||b_i||_\infty < K,$$

$$\begin{cases} c_0 = a_0b_0 - a_1b_{n-1} - a_2b_{n-2} - \ldots - a_{n-1}b_1, \\ c_1 = a_0b_1 + a_1b_0 - a_2b_{n-1} - \ldots - a_{n-1}b_2, \\ c_2 = a_0b_2 + a_1b_1 + a_2b_0 - \ldots - a_{n-1}b_3, \\ \vdots \\ c_{n-1} = a_0b_{n-1} + a_1b_{n-2} + a_2b_{n-3} + \ldots + a_{n-1}b_0. \end{cases}$$

As this is an underdetermined system, the linearization $Z_{i,j} = a_i b_j$ results in a linear system with too many degrees of freedom to select the correct solution; this is not better than guessing in the initial quadratic system. On the other hand, this system presents cyclic anti-symmetry, which one could exploit to find a solution. However, it is not clear how to use the additional symmetry to make progress in finding solutions (this is also the case when trying to solve lattice problems in the particular case of ideal lattices).

From another point of view, we are given an element c in the intersection $\langle a \rangle \cap \langle b \rangle$ of ideals of $R = \mathbb{Z}[x]/(x^n + 1)$, and a test routine $T : R \to \{0, 1\}$ that outputs 1 if the input is b and 0 otherwise (in our scenario, the test routine is to simply try out the extracted key $\beta = b$ via decryptions). An algorithmic issue arises again: There is a degree of freedom of one ring unit in the small

factors problem, and an algorithm must exclude trivial factorizations of c: for instance, if nothing was required for a and b other than invertibility, the size of the candidates list for (a, b) is at least the number of units of R_q, since it contains all pairs $(a, b) = (cu, cu^{-1})$ for invertible $u \in R_q$. Using the K-boundedness of a, b, the list is to be reduced rejecting all incorrect pairs. To optimize up the rejection, we suggest a study of the distribution $\chi \stackrel{def}{=} (\bar{\mathcal{G}}_K)^{-1}$, which samples e according to $\bar{\mathcal{G}}_K$ and then outputs $e^{-1} \in R_q$ if e is invertible.

5 Two-party multiplication protocols in R_q

In this section we introduce two protocols to jointly achieve multiplication in the quotient ring between two mutually distrusting parties. We distinguish two settings, the "secret inputs" (which is the classical MPC scenario) and the "shared inputs" which supposes that both parties have additive shares of some elements. The latter setting, however, can be regarded as a classical MPC computing a quadratic expression of the inputs.

5.1 Secret inputs setting

Alice and Bob hold $f \in R_q$ and $g \in R_q$ respectively. The following protocol allows them to multiply these elements: Alice will learn $fg + r \in R_q$ where r is a polynomial chosen by Bob. The reason of this is that if Alice learns fg, she can compute $g = fg/f$. The utility of this protocol may seem questionable, in the sense that it transfers Alice's obliviousness from g to r, nevertheless we will see that careful selection of r will allow the two parties to generate Alice's NTRU keys. This protocol is inspired on [3], where authors propose a protocol to compute scalar products as a building block to perform much more complex functionalities. It is detailed in algorithm 1.

Algorithm 1 TMP

Require: Alice holds $f \in R_q$, Bob holds $g \in R_q$. Let p, m be public integers.
Ensure: Alice learns $fg + r \in R_q$ where Bob knows $r \in R_q$

1: Alice generates m random polynomials $\{f_1, \ldots, f_m\} \stackrel{R}{\subset} R_q$ such that $\sum_{i=1}^{m} f_i = f$.
2: Bob generates m random polynomials $\{r_1, \ldots, r_m\} \stackrel{R}{\subset} R_q$ and $r \stackrel{def}{=} \sum_{i=1}^{m} r_i$.
3: **for** $i = 1, \ldots, m$ **do**
4: Alice generates a secret random number $k, 1 \leq k \leq p$.
5: Alice generates random polynomials $v_1, \cdots, v_p$, sets $v_k = f_i$, and send all these polynomials to Bob.
6: Bob computes the products and masks them: For all $j = 1, \ldots, p$ $z_{i,j} = v_j g + r_i$.
7: Alice extracts $z_{i,k} = f_i g + r_i$ from Bob with a 1–out–of–p OT protocol.
8: **end for**
9: Alice computes $\sum_{i=1}^{m} z_{i,k} = fg + r$.

Note that throughout the protocol, Bob always computed products of random polynomials, and to guess the value of f he has to perform $\approx p^m$ additions.

Lemma 5.1.1 *If it is not feasible to compute $O(p^m)$ additions in R_q, and if the RLWE assumption holds for $q, \phi(x) = x^n + 1$ and uniform χ over R_q,* TMP *securely outputs $fg + r$ to Alice and r to Bob in the presence of semi-honest parties.*

Proof: In this model, both parties follow exactly the protocol but try to learn as much information as possible from their transcript of the protocol. Let view_A, view_B be the collection of learned elements by Alice and Bob respectively. We have that view_B contains only polynomials $v_j^{(i)}$ indistinguishable from uniform (since they were sampled by semi-honest Alice), and these elements are independent from Bob's input, samplings, and computations. Therefore, to learn f, he needs to perform $\approx p^m$ additions. On the other hand Alice wants to learn g or r and she only has m pairs of the form $(f_i, f_i g + r_i)$ (and the output which is the component-wise sum of these), which by the RLWE assumption are indistinguishable from (f_i, u_i) for uniform u_i. In other words, the view of each adversary contains her input, her output, and a list of polynomials indistinguishable from random by construction. We can construct simulators $\mathcal{S}_A, \mathcal{S}_B$ of protocol TMP for both parties, and it follows immediately that the views of Alice and Bob are indistinguishable from the simulators. □

Remark: If both parties are malicious but they do not want to leak their own inputs, at the end of the protocol they learn nothing about the other party's input.

This holds because Alice may deviate from the samplings, but she sends pm random elements computationally hiding f, Bob will process these pm elements (deviating as much as he wants from the actual required computation) and send m elements computationally hiding g and r to Alice via the OT protocol, thus Bob learns nothing. In this case, deviations from the protocol may cause the output to be incorrect. We do not worry much about this as soon as Bob's input is safe, since we will see that it will result in invalid keys for Alice and the honest party will know that the other is malicious.

5.2 Shared inputs setting

In this setting, two parties share two elements of R_q additively, and they want to compute shares of the product of these elements. Let Alice and Bob hold x_A, y_A and x_B, y_B respectively such that

$$x = x_A + x_B \quad \text{and} \quad y = y_A + y_B.$$

We propose a protocol SharedTMP, at the end of which Alice and Bob will learn additive shares π_A, π_B respectively of the product:

$$\pi_A + \pi_B = xy \in R_q.$$

Algorithm 2 SharedTMP

Require: Alice holds $(x_A, y_A) \in R_q^2$, Bob holds $(x_B, y_B) \in R_q^2$ such that $x = x_A + x_B$, $y = y_A + y_B$

Ensure: Alice learns $\pi_A \in R_q$, Bob learns $\pi_B \in R_q$ such that $\pi_A + \pi_B = xy$

1: Alice samples $r_A \xleftarrow{R} R_q$, Bob samples $r_B \xleftarrow{R} R_q$
2: Alice and Bob perform $\mathsf{TMP}(x_A, y_B)$ using Bob's randomness r_B, thus Alice learns $u_A = x_A y_B + r_B$ and Bob learns nothing.
3: Bob and Alice perform $\mathsf{TMP}(x_B, y_A)$ using Alice's randomness r_A, thus Bob learns $u_B = x_B y_A + r_A$ and Alice learns nothing.
4: Alice computes the share $\pi_A = x_A y_A + u_A - r_A \in R_q$
5: Bob computes the share $\pi_B = x_B y_B + u_B - r_B \in R_q$

Note that $\pi_A + \pi_B = (x_A + x_B)(y_A + y_B) = xy$. Since they only communicate in steps 2 and 3, security is reduced to two independent instances of the TMP protocol. We also have the following observation:

Lemma 5.2.1 *Let Alice and Bob perform* SharedTMP *on some non-trivial inputs, learning at the end π_A and π_B respectively. Even if Alice reveals π_A to Bob, he cannot deduce Alice's inputs.*

Proof: This follows directly from the randomness of $u_A - r_A$. □

6 Excalibur key generation

We present our main contribution, three protocols $\mathsf{Keygen}_{\mathsf{pk}}, \mathsf{Keygen}_{\mathsf{sk}}$ and a validation protocol, to be performed by Alice and Bob that will generate the public and the (blended) private-key of Alice, in that order. Let us first give an informal outline of the protocol. Bob has already generated his key-pair $(\beta, 2h\beta^{-1}) \in R_q \times R_q$. They want to compute a new key-pair $(\mathsf{sk}_A, \mathsf{pk}_A) = (\alpha\beta, 2g(\alpha\beta)^{-1}) \in R_q \times R_q$ for Alice, which correctly decrypts encryptions under pk_B since it contains the factor β.

- Excalibur generation of pk_A
  1. They share polynomials α, g, r of R_q additively, such that $\alpha = 1 \mod 2$.
  2. They perform SharedTMP to obtain shares of $\alpha r, gr$. Alice reveals her shares to Bob.
  3. Bob computes $2(gr) \cdot (\alpha r)^{-1} \cdot \beta^{-1} = 2g(\alpha\beta)^{-1}$ in R_q and broadcasts the result.
- Excalibur generation of sk_A (to be performed after publication of pk_A)
  1. Let $\alpha_A + \alpha_B = \alpha$ denote the same additive sharing of α than in the previous steps, where Alice holds α_A and Bob holds α_B. Alice and Bob perform TMP on entries α_A, β respectively, and Bob chooses $r = \alpha_B\beta$ as the randomness in the protocol.
  2. At the end of the protocol, Alice learns $\alpha_A\beta + r = \alpha\beta = \mathsf{sk}_A \in R_q$, and Bob learns nothing.

– Validation protocol : Alice and Bob run tests to be convinced that the keys are well formed and behave as claimed.

The protocols are described formally in algorithms 3, 4 and 6.

Algorithm 3 Excalibur $\mathsf{Keygen}_{\mathsf{pk}}$

Require: Bob already has his own key-pair $(\mathsf{sk}_B, \mathsf{pk}_B) = (\beta, 2h\beta^{-1}) \in R_q \times R_q$.
Ensure: A public-key for Alice pk_A
1: Alice and Bob sample random shares of elements in R_q:
   – Alice samples $s_A \leftarrow \bar{\mathcal{G}}_K, r_A \xleftarrow{R} R_q, g_A \leftarrow \bar{\mathcal{G}}_K$
   – Bob samples $s_B \leftarrow \bar{\mathcal{G}}_K, r_B \xleftarrow{R} R_q, g_B \leftarrow \bar{\mathcal{G}}_K$
   Let $\alpha = 2(s_A + s_B) + 1, r = r_A + r_B, g = g_A + g_B$ denote the shared elements.
2: Alice and Bob perform $\mathsf{SharedTMP}$ twice to obtain shares of $z = \alpha \cdot r$ and $w = g \cdot r$. Alice reveals her shares, thus Bob learns z, w.
3: Bob checks: If z is not invertible in R_q, restart the protocol.
4: Bob computes $2w(z\beta)^{-1} = 2g(\alpha\beta)^{-1}$ and publishes it as pk_A, along with a NIZK proof showing that z, w come from step 2 and that pk_A is well-formed.
5: Alice verifies Bob's proof. If it is not correct, abort the protocol.

If protocol 3 was carried out properly, a ciphertext encrypted with pk_A is correctly decrypted by any secret-key having the factor $\alpha\beta$ and reasonable coefficient size. Remark that in step 2, Bob received the element $z = \alpha \cdot r$: this does not allow to deduce a functional equivalent of the secret-key $\alpha\beta$, since r has large coefficients. Also, chances are overwhelmingly high that this element is in fact invertible in view of section 2.2.

Algorithm 4 Excalibur $\mathsf{Keygen}_{\mathsf{sk}}$

Require: Bob's secret-key β and the same sharing of $\alpha = 2(s_A + s_B) + 1$ than in protocol 3.
Ensure: A secret-key for Alice $\mathsf{sk}_A = \alpha\beta$
1: Bob computes $r := (2s_B + 1)\beta \in R_q$
2: Alice and Bob perform the protocol $\mathsf{TMP}(2s_A, \beta)$, and Bob uses r as the random polynomial. At the end Alice knows $2s_A\beta + r = \alpha\beta \in R_q$.

Once the keys are generated, they must pass a series of decryption and a well-formedness test. This is described in algorithms 5 and 6. First, Alice checks if her new secret-key works as expected, and then she convinces Bob, via a game of decryptions that she is indeed capable of decrypting ciphertexts encrypted under pk_A and under pk_B. As we will see, this validation protocol avoids malicious activity.

Algorithm 5 Validation function (performed by Alice)

function VALIDATE($\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B$)
  for i from 1 to k **do**
    $\mu \xleftarrow{R} \{0,1\}$
    $\mu_1 \leftarrow \mathsf{Dec}(\mathsf{sk}_A, \mathsf{Enc}(\mathsf{pk}_A, \mu))$
    $\mu_2 \leftarrow \mathsf{Dec}(\mathsf{sk}_A, \mathsf{Enc}(\mathsf{pk}_B, \mu))$
    if $\mu_1 \neq \mu$ or $\mu_2 \neq \mu$ **then** output `reject`
    end if
  end for
  if $||\mathsf{sk}_A||_\infty > n(2K+1)^2$ or $||\mathsf{sk}_A \cdot \mathsf{pk}_B||_\infty > 2(2K+1)$ **then** output `size warning`
  end if
  output `accept`
end function

Algorithm 6 Validation protocol (performed by Alice and Bob)

Require: Alice holds $(\mathsf{sk}_A, \mathsf{pk}_A)$ and Bob holds $(\mathsf{sk}_B, \mathsf{pk}_B)$

1: Alice runs VALIDATE($\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B$). If the output is `reject`, abort.
2: Bob picks $2k$ random messages $(m_1^{(A)}, \ldots, m_k^{(A)})$ and $(m_1^{(B)}, \ldots, m_k^{(B)})$, and for each $i = 1, \ldots, k$ he computes ciphertexts $c_i^{(A)} = \mathsf{Enc}(\mathsf{pk}_A, m_i^{(A)}), c_i^{(B)} = \mathsf{Enc}(\mathsf{pk}_B, m_i^{(B)})$. He send all ciphertexts to Alice.
3: For each $i = 1, \ldots, k$, Alice compute $\mu_i^{(A)} = \mathsf{Dec}(\mathsf{sk}_A, c_i^{(A)}), \mu_i^{(B)} = \mathsf{Dec}(\mathsf{sk}_A, c_i^{(B)})$. She sends all plaintexts to Bob.
4: For each $i = 1, \ldots, k$, Bob checks if $\mu_i^{(A)} = m_i^{(A)}$ and $\mu_i^{(B)} = m_i^{(B)}$.

7 Security

We first discuss the honest-but-curious model, where the protocol is strictly followed but parties try to learn secrets. Then we look at the malicious model, where one party does not follow the protocol properly, in order to steal secrets or to sabotage the key generation.

7.1 Honest-but-curious model

In this model, we suppose that Alice and Bob follow exactly the instructions in algorithms 3, 4, 5 and 6 but they try to learn about each other's secret with all collected information.

Proposition: *If Alice is able to extract Bob's key from the protocol, she can solve the Small Factors Problem or the $\bar{\mathcal{G}}_K^{\times}$-GCD Problem. If Bob is able to extract Alice's key, he can solve the $\bar{\mathcal{G}}_K^{\times}$-GCD Problem.*

Proof: Let us focus first in Bob's chances on learning α (or a functional equivalent of the form $\theta \cdot \alpha$ for small $\theta \in R_q$). Recall that

$$\begin{cases} \mathsf{pk}_B = 2h\beta^{-1}, \\ \mathsf{pk}_A = 2g(\alpha\beta)^{-1}, \\ z = \alpha \cdot r, \\ w = g \cdot r, \\ \alpha = 2(s_A + s_B) + 1. \end{cases}$$

Let us focus on *Bob's view of the protocol:*

$$V = \{(s_B, r_B, g_B, z_B, w_B, z_A, w_A, \beta, \mathsf{pk}_B), \mathsf{pk}_A, \alpha \cdot r, g \cdot r\} \subset R_q.$$

What Bob is curious about: Any element of the set

$$U = \{\alpha, g, r, s_A, g_A, r_A\} \subset R_q.$$

The parentheses in V indicate that he sampled or received the elements contained, and the rest are results of joint computation. The knowledge of any element in U allows Bob to deduce Alice's secret-key α, and only the last three elements $\mathsf{pk}_A, \alpha \cdot r, g \cdot r$ of V depend on elements in U. Thus, extracting α is equivalent to solve the following system of equations in the unknowns $(X, Y, Z) = (\alpha, r, g)$:

$$\begin{cases} b_1 = XY, \\ b_2 = ZY, \\ b_3 = ZX^{-1}, \end{cases}$$

where $b_1 = \alpha \cdot r, b_2 = g \cdot r, b_3 = \beta\mathsf{pk}_A/2$. We can eliminate the third equation noting that $b_1 b_3 = b_2$, and thus Bob faces the Small GCD Problem of section 4.2.

Let us now focus in Alice chances of learning Bob's secret.

Alice's view of the protocol: $W = \{(s_A, r_A, g_A, z_A, w_A), \mathsf{pk}_B, \mathsf{pk}_A, \alpha\beta\} \subset R_q$.
What Alice is curious about: Any element of the set
$Q = \{\alpha, \beta, h, s_B, \{w_B, z_B\}, \{w, z\}\}$.

First, extracting α or β directly from $\alpha \cdot \beta$ is exactly the small factors Problem. Using the only three sensitive elements of W, she faces the following system of equations in $(X, Y, Z) = (h, \beta, \alpha)$

$$\begin{cases} a_1 = XY^{-1}, \\ a_2 = (ZY)^{-1}, \\ a_3 = ZY, \end{cases}$$

where $a_1 = \mathsf{pk}_B/2, a_2 = \mathsf{pk}_A/2g, a_3 = \alpha\beta$. After elimination of the third equation since $a_3 = a_2^{-1}$, Alice also faces the small GCD problem (actually, mapping $Y \mapsto Y^{-1}, Z \mapsto Z^{-1}$ yields to the same gcd problem faced by Bob). □

7.2 Security against one malicious party

We consider the presence of one malicious adversary, a party that deviates as much as she wants from the protocol, but has a list of paramount objectives which she is not willing to sacrifice. We suppose that one of the two parties strictly follows the protocol and the other one is malicious, given the objectives below. We consider the presence of only one somewhat malicious adversary, given that both parties have concurrent objectives (for instance, Bob is trying to protect his key, and Alice to extract it from the protocols). In other words,

What curious Alice wants:

(A1) A functional secret-key sk_A associated with pk_A,
(A2) such that sk_A decrypts encryptions under pk_B,
(A3) protecting elements of $U = \{g_A, r_A, s_A\}$ from Bob and
(A4) to learn β.

What curious Bob wants:

(B1) To give Alice a functional secret-key sk_A associated with pk_A with decryption rights on $\mathsf{Enc}(\mathsf{pk}_B, \mathcal{M})$.
(B2) to protect elements of $Q = \{\beta, h, s_B, \{w_B, z_B\}\}$ from Alice,
(B3) (if malicious) overloading Alice's secret-key sk_A to have large coefficients, and
(B4) to learn α.

We will show that either the keys will be correctly generated or one party will not fulfill all of her objectives.

Malicious Alice, semi-honest Bob Suppose that Bob is strictly following the protocol and Alice may deviate from the protocol but wants to fulfill (A1) to (A4). Let us summarize Alice's participation in the key generation:

1. *Samples* $s_A \leftarrow \bar{\mathcal{G}}_K, r_A \xleftarrow{R} R_q, g_A \leftarrow \bar{\mathcal{G}}_K$.
   Trivial samplings of these elements may ultimately leak α to Bob. For instance, if $s_A = 0$, $\alpha = 2s_B + 1$, if $r_B = 0$, $z/r_B = \alpha$, if $g_A = 0$ $g = w/g_B$. Also, if s_A or g_A have large coefficients, there is risk of mod q wrap-around in the decryption procedure with sk_A. As she is sampling only shares of elements, she cannot force algebraic relations with them: regardless of her samples, α, g, r will remain indistinguishable from random.
2. *Participates in* $\mathsf{SharedTMP}((2s_A, r_A), (2s_B + 1, r_B))$ *and learns* z_A*, participates in* $\mathsf{SharedTMP}((g_A, r_A), (g_B, r_B))$ *and learns* w_A*, then sends* z_A, w_A *to Bob.*
   As discussed in section 5, TMP and SharedTMP are secure if Bob is honest, in the sense that either Alice learns the correct output, or either she learns indistinguishable from random elements, but she learns nothing about Bob's input. She is limited to alter the inputs of both instances of SharedTMPand then giving wrong z_A or w_A to Bob. Nevertheless if she inputs different r_A's in both protocols or if she changes the values of z_A, w_A before sending them to Bob, from the linearity of shares and the randomness of Bob's entries it follows that this sabotages the relation $wz^{-1} = g\alpha^{-1}$, needed for correctness of decryption. In other words, in order to ensure (A1) and (A2), she is forced to maintain the input r_A for both instances of SharedTMPand send the correct output to Bob.
3. *Participates in* $\mathsf{TMP}(\{2s_A\}, \{2s_B + 1\})$ *and learns* $\alpha\beta$.
   If she uses the correct value of $2s_A$ (i.e. the same as in step 2), she learns the correct output $\alpha\beta$. If she inputs another value $x \neq 2s_A$, she does learn a functional equivalent of Bob secret (namely, $(x + 2s_B + 1)\beta$), but she is not able to decrypt encryptions under the already published pk_A, failing the verification procedure.

Malicious Bob, semi-honest Alice Now suppose the inverse case, where Alice follows the protocol strictly and Bob is protecting β and guessing α, deviating as much as he wants from the protocol but fulfilling (B1) to (B4). We begin by saying that (B3) is unavoidable (unless the presence of a zero-knowledge proof that Bob's polynomials are of the right size), but Alice can tell if Bob overloaded the secret-key $\alpha\beta$ simply looking at the coefficients. Let us now summarize Bob's participation in the key generation:

1. *Samples* $s_B \leftarrow \bar{\mathcal{G}}_K, r_B \xleftarrow{R} R_q, g_B \leftarrow \bar{\mathcal{G}}_K$.
   Trivial sampling may compromise sensible elements as before. He must ensure the randomness of α, r if he wants to protect these elements, and on the other hand Alice will know if he deviates from a K-bounded sampling (just looking at the coefficients in $\alpha \cdot \beta$. Therefore, he gains nothing in deviating from a K-bounded sampling.

2. *Participates in* $\mathsf{SharedTMP}((2s_A, r_A), (2s_B + 1, r_B))$ *and learns* z_B*, participates in* $\mathsf{SharedTMP}((g_A, r_A), (g_B, r_B))$ *and learns* w_B*.*

   As noted in section 5, because of Alice's randomness in SharedTMP, either Bob obeys the protocol an receive the correct outputs, either he deviates and receives random outputs, from which he cannot deduce secret values and which sabotage key generation. Also, if he uses different r_B's in both instances, Alice will not be able to decrypt since the decryption relation $wz^{-1} = g\alpha^{-1}$ is not fulfilled (and he remains oblivious of r_A, not being able to force this relation). Hence, he is forced to follow SharedTMP and use the same r_B in both instances if he wants to fulfill (B1).

3. *Receives* z_A, w_A*, learning* z, w*. Checks if* z *is invertible and publishes* $\mathsf{pk}_A = 2w(z\beta)^{-1}$*. He then participates in* $\mathsf{TMP}(\{2s_A\}, \{2s_B + 1\})$*, chooses* $R = (2s_B + 1)\beta$ *and learns nothing.*

   Suppose that he published pk'_A as Alice's public-key and participated in the TMP instance with generic values, indicated by an apostrophe. At the end, Alice knows pk'_A and sk'_A. She will run the validate function of algorithm 5 to check (i) if sk'_A has the expected coefficient size, (ii) if $\mathsf{sk}'_A \cdot \mathsf{pk}'_A = 2g'$ for a vector $g' \in \chi$ and (iii) if she is able to decrypt encryptions of messages under pk'_A and pk_B. If she is indeed able to decrypt encryptions under pk_B, then sk'_A contains the factor β, thus by randomness of s_A, $\beta' = \theta\beta$ and $R' = \omega\beta$ for small θ and ω. Also, as long as the polynomials sk'_A and $\mathsf{sk}'_A \cdot \mathsf{pk}'_A$ are of the right form, she does not care about how Bob computed pk'_A, as decryption of encryptions under pk'_A work as claimed. If on the contrary a single decryption fails or if sk'_A or $\mathsf{sk}'_A \cdot \mathsf{pk}'_A$ have large coefficients, she can claim one of the following Bob's wrongdoings: Either he did not include β, either he included $\theta\beta$ for too large θ, either he sabotaged entirely the key generation in a change of input or inside a multiparty multiplication protocol. This allows to conclude that if Bob fails to give what is expected, the output keys will be rejected by Alice, who discovers Bob's maliciousness after the validation protocol 6.

We should point out another strategy that Bob could maliciously try. When generating Alice's secret-key, he could simply ignore Alice's input share $2s_A$, and thus the protocol gives Alice the key $\mathsf{sk}'_A = \alpha'\beta$, for an $\alpha' \in R_q^\times$ of Bob's choice. Bob, who received no output from the protocol, can reconstruct this key, thus gaining Alice's secret. However, this key will be rejected by Alice since it cannot be associated with the previously generated $\mathsf{pk}_A = 2g(\alpha\beta)^{-1}$. To avoid this rejection, Bob should have published $\mathsf{pk}'_A = 2g'(\alpha'\beta)^{-1}$ instead, but it is easy to see that this publication would contradict the NIZK proof of step 4 of algorithm 3: Because of the way SharedTMP works, Bob has no way of choosing α' of his choice in the expression $z = \alpha r$. In view of this, passing the validation protocol with such a key is overwhelmingly unlikely.

8 Extensions

8.1 Chains of keys

Suppose Alice and Bob perform the latter protocols, such that Alice has now a private-key of the form $\mathsf{sk}_A = \alpha\beta$ where β is Bob's secret-key. Alice can repeat the protocol with a third user Charlie (with slight coefficients size modifications at the validation protocol), who at the end receives a pair of keys of the form $(\mathsf{sk}_C, \mathsf{pk}_C) = (\alpha\beta\gamma, 2g_C(\alpha\beta\gamma)^{-1})$. As his secret-key contains the factors β and $\alpha\beta$, he can decrypt both Bob's and Alice's ciphertexts. This shows that easy modifications to the protocol allows to generate a chain of users, each one inheriting the previous user decryption rights. From corollary 3.1.2, it is easy to see that the length of such a chain is at most $\approx \log(q/nK)$ to ensure decryptions (this matches the maximum number of keys on the multikey LATV FHE scheme for the same parameters). We point out that intersecting chains are also possible, meaning that a user can glue her secret-keys to two or more upper-level users and even if they collude they are not able to extract his key. This comes from an easy generalization of our $\bar{\mathcal{G}}_K^{\times}$–GCD problem.

8.2 Plugging in LATV-FHE

Because of the form of an Excalibur key, i.e. $(\mathsf{sk}, \mathsf{pk}) = (\prod_{i=1}^{r} \alpha_i, 2g \prod_{i=1}^{r} \alpha_i^{-1})$, the inclusion of our protocols into the Multikey FHE scheme from [23] is immediate. The only missing element are the evaluation keys, which can be generated easily by the secret-key holder after the (Excalibur) key generation: they are "pseudo-encryptions" of the secret-key sk under the public-key pk. This achieves a somewhat homomorphic encryption scheme in the chain of users, where in addition they can combine ciphertexts generated by any public-key.

9 Conclusion

In this article, we proposed a new protocol to generate NTRU keys with additional decryption rights, allowing to form a hierarchic chain of users. We motivated such a procedure because it avoids betrayal naturally, and since it applies to the FHE-NTRU scheme, it may contribute to clear the bootstrapping-like re-encryption paradigm, since it is to our knowledge the first FHE scenario featuring (the pure definition of) proxy re-encryption. In this light, it concurs with other proxy re-encryption schemes, as, while being rigid, ciphertext transformation is no necessary at all, since decryption rights are defined in key-generation time. We used two-party computation protocols as building blocks, and relied the semantic security on the well-known RLWE and DSPR assumptions, and security in presence of semi-honest parties on a hardness assumption in cyclotomic polynomial rings.

Acknowledgments

We would like to thank Pablo Schinke Gross for suggesting the term Excalibur and the INDOCRYPT 2016 anonymous reviewers for their helpful comments. This work has been supported in part by the FUI CRYPTOCOMP project.

References

1. Alperin-Sheriff, J., Peikert, C.: Practical Bootstrapping in Quasilinear Time, pp. 1–20. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
2. Alperin-Sheriff, J., Peikert, C.: Faster Bootstrapping with Polynomial Error, pp. 297–314. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
3. Atallah, M.J., Du, W.: Secure multi-party computational geometry. In: International Workshop on Algorithms and Data Structures. pp. 165–179. Springer-Verlag (2001)
4. Boneh, D., Franklin, M.: An Efficient Public Key Traitor Tracing Scheme, pp. 338–353. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
5. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme, pp. 45–64. Springer Berlin Heidelberg (2013)
6. Brakerski, Z.: Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP, pp. 868–886. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
7. Brakerski, Z., Gentry, C., Halevi, S.: Packed ciphertexts in lwe-based homomorphic encryption. In: Public-Key Cryptography PKC 2013, Lecture Notes in Computer Science, vol. 7778, pp. 1–13. Springer Berlin Heidelberg (2013)
8. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science. pp. 97–106. FOCS '11, IEEE Computer Society, Washington, DC, USA (2011)
9. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Proceedings of the 31st Annual Conference on Advances in Cryptology. pp. 505–524. CRYPTO'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2033036.2033075
10. Gentry, C.: Computing on encrypted data. In: Proceedings of the 8th International Conference on Cryptology and Network Security. pp. 477–477. CANS '09, Springer-Verlag, Berlin, Heidelberg (2009)
11. Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. thesis, Stanford, CA, USA (2009), aAI3382729
12. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing. pp. 169–178. STOC '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1536414.1536440
13. Gentry, C., Halevi, S.: Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In: Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science. pp. 107–109. FOCS '11, IEEE Computer Society, Washington, DC, USA (2011)

14. Gentry, C., Halevi, S.: Implementing gentry's fully-homomorphic encryption scheme. In: Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology. pp. 129–148. EUROCRYPT'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2008684.2008697
15. Gentry, C., Halevi, S., Peikert, C., Smart, N.P.: Ring switching in bgv-style homomorphic encryption. In: Proceedings of the 8th International Conference on Security and Cryptography for Networks. pp. 19–37. SCN'12, Springer-Verlag, Berlin, Heidelberg (2012)
16. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Proceedings of the 15th International Conference on Practice and Theory in Public Key Cryptography. pp. 1–16. PKC'12, Springer-Verlag, Berlin, Heidelberg (2012)
17. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques. pp. 465–482. EUROCRYPT'12, Springer-Verlag, Berlin, Heidelberg (2012)
18. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic Evaluation of the AES Circuit, pp. 850–867. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
19. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J. (eds.) Advances in Cryptology CRYPTO 2013, Lecture Notes in Computer Science, vol. 8042, pp. 75–92. Springer Berlin Heidelberg (2013)
20. Halevi, S., Shoup, V.: Algorithms in helib. In: Garay, J., Gennaro, R. (eds.) Advances in Cryptology CRYPTO 2014, Lecture Notes in Computer Science, vol. 8616, pp. 554–571. Springer Berlin Heidelberg (2014)
21. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes, pp. 206–222. Springer Berlin Heidelberg (1999)
22. Lauter, K., Lopez-Alt, A., Naehrig, M.: Private computation on encrypted genomic data. Tech. rep. (2014), http://research.microsoft.com/apps/pubs/default.aspx?id=219979
23. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: In STOC. pp. 1219–1234 (2012)
24. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: In Proc. of EUROCRYPT, volume 6110 of LNCS. pp. 1–23. Springer (2010)
25. Stehlé, D., Steinfeld, R.: Making NTRU as Secure as Worst-Case Problems over Ideal Lattices, pp. 27–47. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
26. Thomae, E., Wolf, C.: Solving Underdetermined Systems of Multivariate Quadratic Equations Revisited, pp. 156–171. Springer Berlin Heidelberg (2012)
27. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Secure pattern matching using somewhat homomorphic encryption. In: Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop. pp. 65–76. CCSW '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2517488.2517497