IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 3, 26 pages.

https://doi.org/10.62056/aebn-4c2h

FINALLY: A Multi-Key FHE Scheme Based on NTRU and LWE

Jeongeun Park[1], Barry van Leeuwen[2] and Oliver Zajonc[2]

[1] Norwegian University of Science and Technology, Trondheim, Norway
[2] KU Leuven, COSIC, ESAT, Leuven, Belgium

Abstract. Multi-key fully homomorphic encryption (MKFHE), a generalization of fully homomorphic encryption (FHE), enables a computation over encrypted data under multiple keys. The first MKFHE schemes were based on the NTRU primitive, however these early NTRU based FHE schemes were found to be insecure due to the problem of over-stretched parameters. Recently, in the case of standard (non-multi key) FHE a secure version, called FINAL, of NTRU has been found. In this work we extend FINAL to an MKFHE scheme, this allows us to benefit from some of the performance advantages provided by NTRU based primitives. Thus, our scheme provides competitive performance against current state-of-the-art multi-key TFHE, in particular reducing the computational complexity from quadratic to linear in the number of keys.

1 Introduction

Fully homomorphic encryption (FHE) allows for computation on encrypted data without first decrypting. This has the advantage that a distrusting party can allow a computationally stronger party to perform computations without revealing that party's input or output. Multi-key fully homomorphic encryption (MKFHE) schemes take this a step further and allow for computations on encrypted data using multiple keys. As these keys can be held by different parties, multiple distrusting parties can have another party perform a computation on their combined inputs while preserving the privacy of the inputs.

Most FHE schemes prevent the security concerns of have deterministic encryption protocols by introducing a small random noise term during their encryption steps. This prevents the adversary from learning anything about the underlying message based solely on the ciphertext. This also means that in general, FHE decryption involves a rounding step to remove the noise if it's small enough. However, as FHE schemes are designed to be used in operations, the noise imparted during encryption is not the only noise we need to worry about. During addition, the noise likewise grows linearly, but during multiplication, the noise grows exponentially. To accommodate arbitrary sizes of computations, FHE schemes implement a bootstrapping protocol, which takes some ciphertext with large, but acceptable noise, and performs an operation that returns a ciphertext encrypting the same message but with a guaranteed smaller noise bound. Thus, FHE schemes must be able to support a single arbitrary operation followed by a run of the bootstrapping algorithm to be considered valid.

Beyond the noise growth from performing computations, it is known from prior work, [CCS19, KLSW21], that multi-key FHE schemes have significant error inflation compared to their single key counterparts, often in terms of the number of participating parties. To

E-mail: jeongeun.park@ntnu.no (Jeongeun Park), barry.vanleeuwen@kuleuven.be (Barry van Leeuwen), oliver.zajonc@esat.kuleuven.be (Oliver Zajonc)

Received: 2024-07-05 Accepted: 2024-09-02

accommodate this, MKFHE schemes may require larger parameter sizes than single-key FHE to support these extra players.

The first MKFHE scheme [LATV12] was based on the NTRU problem, which had the advantage of allowing ciphertext sizes to be nearly independent of the number of players. However, this construction was shown not to be secure due to algebraic attacks on the NTRU problem [ABD16, CJL16, Dv21], specifically on instantiations with large parameters. These large parameters were thought to be required to accommodate the noise growth of an MKFHE scheme, and thus research on NTRU-based FHE stalled. Despite NTRU-based MKFHE seeming infeasible, other approaches to MKFHE have been developed [LP19, CCS19, KKL+23], based on standard or ring LWE.

In this paper, we reopen the case for NTRU-based MKFHE by expanding on FINAL [BIP+22], a single key FHE scheme based on the NTRU problem introduced in 2022. FINAL adapted the FHEW [DM15] framework to the NTRU problem, which allowed the NTRU parameters to be kept small enough to ensure security against the aforementioned algebraic attacks on NTRU. We use these advantages to improve on the state-of-the-art by amortizing the key switching key generation phase and reducing the overall complexity of the bootstrapping phase.

As explained above, due to the constraints of NTRU parameters, a multi-key extension of FINAL might not be able to handle many parties. However, most practical applications of MKFHE are often optimized for two or three parties as is the case in [CDKS19, AH19]. The construction in this case is often between a server and a client or between two clients with an intermediary server, such as a machine learning scenario in which a (semi-honest) server receives input from clients and a predefined model (see Figure 1.)

In more detail, this scenario for two keys work as follows. A client wants to evaluate a machine learning algorithm of which parameters are owned by the model provider (server) on its "different set of" data, without interaction. Therefore, the model (parameters) is fixed, but multiple distinct evaluations are expected between the client and the server. Independently, other clients may want to do the same thing with the server. In this case, MKFHE for two keys perform better than other similar primitives such as mjulti party FHE due to its non-interactability and small enough size of ciphertexts.

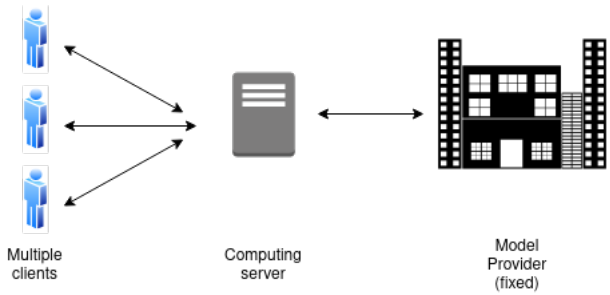

Figure 1: An example of a scenario with 2 keys: one for a client (left) and one for the model provider (right), along with an intermediary server who does not hold any keys.

1.1 Our contributions

We provide an efficient multi-key extension of NTRU based FHE based on thorough theoretical noise analysis and improving existing approaches like the following:

Preprocessable Lightweight Key Switching In section 4.4 we present a new key switching protocol based on the HybridProduct construction defined in [CCS19] and refined in [KMS24]. This new protocol avoids the use of the HybridProduct construction inline, as

Table 1: High level comparison of $MK-\mathsf{TFHE}$ protocols versus our protocol, where k is the number of players, N is the maximal degree of polynomials used in the encryption, and l is $\lceil \log_B(Q) \rceil$, where B is some base and Q is the modulus.

Protocol	Polynom. Mult.	Key Material Req.($k=2$)	Sec. Param.
Alg 2, [KMS24]	$O(Nlk + lk^2)$	215MB	2^{110}
Alg 1, [CCS19]	$O(Nlk^2)$	84MB	2^{110}
Ours	$O(Nlk)$	212MB	2^{128}

in [XTW+23], by only using it to compute the key-switching keys which are then used as the input to the key-switching protocol. This allows the key switching keys to be pre-processed ahead of time by the server. While this generation step does not become any more efficient, the cost can now be amortized over the number of bootstraps that need to be done without increasing the noise generated by the Key Switching protocol compared to earlier works.

Multi-key FHE combining LWE and NTRU In section 3.2, we present a multi-key variant of the GSW-like scheme based on the NTRU (also called NTRU-based GSW or NGS) presented in [BIP+22]. This adaptation allows to evaluate a function over multi-key LWE ciphertexts, under their multiple secret keys, and reduce the noise level from the resulting ciphertext at the same time, by running a bootstrapping protocol (blind rotation) which computes external products among NTRU ciphertexts (bootstrapping keys). Thanks to the nature of the NTRU problem, the size of the external product output does not increase. While a common reference string (CRS) is required in our construction, its use is limited to a pre-processing phase in the key switching phase of the bootstrapping. Hence future improvements may remove the protocol's dependency on a CRS without the need to re-establish a new protocol.

Faster Bootstrapping than $MK-\mathsf{TFHE}$ Compared to the current state-of-the-art within $MK-\mathsf{TFHE}$, [KMS24] and [CCS19], we bring improvements in both key size and number of polynomial multiplications on small player sets for equivalent parameter sizes as can be seen in Table 1. Furthermore, for any parameter sets and player numbers we reduced the computational complexity from quadratic in the number of players, k, to linear. As in [KMS24], it is possible to run the blind rotation for each party in parallel, and merged afterwards.

1.2 Concurrent work

In [XTW+23], the authors propose another multi-key FHE construction based on NTRU which provides a strong technical foundation which can be applied to FINAL. One of the key elements of this scheme is the introduction of keys with fixed Hamming weight, allowing for a significant reduction in the noise generated by this protocol. However, the use of fixed Hamming weight keys in NTRU was shown to have a damning security weakness in [KL23], rendering the aforementioned protocol unusable for multi-key FHE constructions.

Another recent work [AKÖ23], shows a high performance approach to $MK-\mathsf{TFHE}$, however due to the joint calculation of evaluation keys it can not be compared to this work, where no communication between the participating parties is required. A similar approach is likely possible with this scheme, but is left for future work.

A related line of work is that studying another primitive known as threshold multi-key FHE or multi-party FHE [AJL+12, Par21]. This primitive avoids the ciphertext expanding

in relation to the number of keys, but this setup requires at least one interaction among a fixed number of participants to generate a common public key, thus abandoning the dynamic property, i.e. the ability for players to join the computation arbitrarily.

2 Preliminaries

2.1 Notation

For a polynomial $g(X) \in \mathbb{K}[X]$ we denote by g_i the coefficient corresponding to the i'th power of X, where $\mathbb{K}$ is a ring or field. We denote vectors by lower-case bold letters, such that $\mathbf{a} = (a_1, \ldots, a_n)$, and matrices by upper-case bold letters, $\mathbf{M} \in M_{r\times c}(\mathbb{K})$, such that $M^{(i)}$ is the i'th row of $\mathbf{M}$ and $\mathbf{M}_{(j)}$ is the j'th column of $\mathbf{M}$. As both $M^{(i)}$ and $M_{(j)}$ are vectors a single index can be given by $\mathbf{M}_j^{(i)} = \mathbf{M}_{(j),i} = \mathbf{M}_{i,j}$.

Throughout the paper $\mathcal{R} := \mathbb{Z}[X]/\langle X^N + 1\rangle$ is the $2N$-th cyclotomic ring, where $N = 2^\kappa$ for some $\kappa \in \mathbb{Z}^+$. We define $\mathcal{R}_Q := \mathcal{R}/Q\mathcal{R} = \mathbb{Z}_Q[X]/\langle X^N + 1\rangle$ in the same vain. Note that any element $f \in R$, $R \in \{\mathcal{R}, \mathcal{R}_Q\}$, can be given as the smallest polynomial of at most degree N in the appropriate coset of R. This makes the vector representation of f given by $\phi(f) = (f_0, \ldots, f_n)$ well defined and so the infinity norm is likewise well defined: $\|f\| = \|\phi(f)\|$.

2.2 Distributions

For a random variable a we denote by $a \leftarrow D$ the action of sampling a from a distribution D. Similarly, we denote by $a \xleftarrow{\$} \mathbb{K}$, the sampling of an element of the set $\mathbb{K}$ uniformly randomly. Denote by G_σ the discrete Gaussian distribution over a ring $\mathcal{R}$ with variance σ^2. Moreover, by $\mathsf{Var}(a)$ we denote the variance of a.

A random variable, V, is called α-subgaussian if the moment generating function fulfills the following equation for some α and all $t \in \mathbb{R}$:

$$\mathbb{E}[\exp(t \cdot V)] \leq \frac{1}{2}\exp(\alpha^2 \cdot t^2).$$

We denote by χ_α the α-subgaussian distribution. We will use the fact that for $V \leftarrow \chi_\alpha$, $\mathsf{Var}(V) \leq \alpha^2$ and the following for a vector of random variables $\mathbf{a} := (a_1, a_2, \ldots, a_n)$ where $a_i \leftarrow \gamma_{\alpha_i}, i \in [n]$ for our noise analysis:

$$\mathsf{Var}(\mathbf{a}) = \max_{i\in[n]}(\mathsf{Var}(a_i)) = \max_{i\in[n]} \alpha_i.$$

Furthermore, we employ the benefit of subgaussian variables called the Pythagorean Additivity property, which means that for any $t, s \in \mathbb{Z}$ and $a \leftarrow \chi_\alpha$, $b \leftarrow \chi_\beta$, it holds that $\mathsf{Var}(t \cdot a + s \cdot b) = \sqrt{t^2 \cdot \alpha^2 + s^2 \cdot \beta^2}$. Lastly, for subgaussian random variables, whose mean is 0, $a, b \in R[X]$, it holds that $\mathsf{Var}(a \cdot b) \leq N \cdot \mathsf{Var}(\phi(a)) \cdot \mathsf{Var}(\phi(b))$.

A specific distribution which the NTRU-section of the protocol will use is the FINAL distribution, which is a subgaussian distribution with variance $\leq \frac{1}{2}$ given as follows:

$$\Pr(f_i = x) = \begin{cases} \frac{1}{4}, & x = -1 \\ \frac{1}{2}, & x = 0 \\ \frac{1}{4}, & x = 1 \end{cases}.$$

We denote by $D_\mathcal{R}$ the distribution over $\mathcal{R}$ where each of the coefficients is independently drawn from the FINAL distribution.

2.3 NTRU Problems

Following standard definitions NTRU we define $\mathbb{M} = \{0, \pm 1\}$ to be the set such that $\forall m : m \leftarrow \mathbb{M}$ according to the FINAL distribution. Similarly to [BIP+22, Dv21], we define the *anti-circulant* version of the NTRU problem.

Definition 1 (Anti-circulant NTRU)**.** Let $N > 0, Q > 1$ be integers and let $\mathcal{R} = \mathbb{Z}[X]/\langle X^N + 1\rangle$. Let $\sigma > 0$ be a real number and let $g, f \in G_\sigma^n$ and f invertible in $\mathcal{R}_Q$. The (computational) $(N, Q, \sigma) -$ NTRU problem is to recover f and g given $h := g \cdot f^{-1} \mod Q$.

The decisional $(N, Q, \sigma) -$ NTRU problem is to distinguish between h and $r \xleftarrow{\$} \mathcal{R}_Q$.

2.4 Gadget Decomposition

For fixed integers Q, B let $l = \lceil \log_B(Q) \rceil$. The l-dimensional column vector defined by $\mathfrak{g}_{Q,B} := (B^0, \ldots, B^{l-1})$ is called the gadget vector. For any identity matrix $\mathbf{I}_k$ we can then define the gadget matrix $\mathbf{G}_{k,Q,B} = \mathbf{I}_k \otimes \mathfrak{g}_{Q,B}$. To avoid confusion we will write $\mathfrak{g} = \mathfrak{g}_{Q,B} = \mathbf{G}_{1,Q,B}$ dropping parameters if they are clear from context. For any integer $z \in \mathbb{Z}_Q$ represented in the symmetric interval, $[-Q/2, Q/2)$, define its signed decomposition as $\mathfrak{g}^{-1}(z) = (z_0, \ldots, z_{l-1})$ such that $|z_i| < B/2$ for each $i \in [0, \ldots, l-1]$. It is easy to see that $\mathfrak{g}^{-1}(z) \cdot \mathfrak{g} = z$.

This gadget decomposition extends to polynomials in the natural way. For any $f \in \mathcal{R}_Q$ we may define

$$\mathbf{G}^{-1}_{n,Q,B}(f) = \sum_{i=0}^{n-1} \mathbf{G}^{-1}_{n,Q,B}(f_i) \cdot X^i.$$

From this definition it is clear that $\mathbf{G}^{-1}_{n,Q,B}(f) \cdot \mathbf{G}_{n,Q,B} = f$, fulfilling the requirement of a gadget matrix.

2.5 Single Key NGS

As introduced in FINAL, [BIP+22], we recall the definition of the single key NGS scheme, as well as some important properties of the noise generation within the scheme. To analyze the noise of the scheme FINAL assumes that for every $r \in \mathcal{R}_Q : \mathfrak{g}^{-1}(r)$ is γ-subgaussian for some $\gamma = O(B)$ as in [DM15, CGGI20, CDKS19], where $\mathfrak{g}^{-1}$ is the gadget decomposition as defined above.

Definition 2. Given a security parameter λ, the NGS scheme consists of four probabilistic polynomial time algorithms: Setup, KeyGen, SEnc, and VEnc.

- NGS.Setup(λ): Upon input of a security parameter λ returns the public parameters (N, Q, ζ, B, l), where B is the decomposition base and $l = \lceil \log_B(Q) \rceil$
- NGS.KeyGen(ζ, N): Upon input of N and ζ, KeyGen samples $f' \leftarrow \chi_\zeta^N$ until $f^{-1} = (4 \cdot f' + 1)^{-1}$ exists in $\mathcal{R}_Q$. Then it outputs $\mathsf{sk} = f$.
- $\mathsf{NGS.SEnc}_{\mathsf{sk}}(m)$: Upon input of secret key f and a plaintext message $m \in \mathbb{F}_3[X]$, NGS.SEnc samples $g \leftarrow \chi_\zeta^N$ and sets $\Delta = \left\lfloor \frac{Q}{4} \right\rceil$. It then outputs

$$c = \frac{g}{f} + \Delta \cdot m.$$

This is called a *scalar encryption of* m.

- $\mathsf{NGS.VEnc}_{\mathsf{sk}}(m)$: Upon input of secret key sk and a plaintext message m, $\mathsf{NGS.VEnc}$ samples $g_i \leftarrow \chi_\zeta^N$ for $i \in (0, l-1)$ and sets $\mathbf{g} = (g_1, \ldots, g_{l-1})$. It then outputs

$$\mathbf{c} = \frac{\mathbf{g}}{f} + \mathfrak{g} \cdot m.$$

  This is called a *vector encryption of* m.

The NGS scheme is equipped with one operation: an external product. This product takes as input a vector encryption of m' and a scalar encryption of m and outputs a scalar encryption of $m \cdot m'$.

To see this let c and $\mathbf{c}$ be as defined in Definition 2 such that $c = \mathsf{NGS.SEnc}(m)$ and $\mathbf{c} = \mathsf{NGS.VEnc}(m')$. Then the external product $\boxdot : \mathcal{R}_Q \times \mathcal{R}_Q \to \mathcal{R}_Q$ is defined as

$$c \boxdot \mathbf{c} = \mathfrak{g}^{-1}(c) \cdot \mathbf{c}.$$

To analyze the noise of this external product we first observe these two basic properties:

Definition 3 (Noise of a Scalar Ciphertext)**.** Let $c = g/f + \Delta \cdot m$, then the noise of c is denoted and defined as

$$\mathsf{err}(c) := c \cdot f - \Delta \cdot m = g + (f-1) \cdot \Delta \cdot m.$$

This can be interpreted as a polynomial over $\mathbb{Z}[X]$ with coefficients in $\left[-\frac{Q}{2}, \frac{Q}{2}\right]$.

Definition 4 (Noise of a Vector Ciphertext)**.** Let $\mathbf{c} = \mathbf{g}/f + \mathfrak{g} \cdot m$, then the noise of $\mathbf{c}$ is denoted and defined as

$$\mathsf{err}(\mathbf{c}) := \mathbf{c} \cdot f - \mathfrak{g} \cdot m \cdot f = \mathbf{g}.$$

This can be interpreted as a polynomial over $\mathbb{Z}[X]$ with coefficients in $\left[-\frac{Q}{2}, \frac{Q}{2}\right]$.

These observations result in a rigorous analysis of a chain of external products between a single scalar NGS ciphertext and k vector NGS ciphertexts.

Lemma 1 (Noise of a sequence of external products [BIP+22])**.** *Let $c_0 = g_0/f + \Delta \cdot m_0$ encrypting binary polynomial $m_0 \in \mathcal{R}_Q$ and let, for every $i \in \{1, \ldots, k\}$, $\mathbf{c}_i = \mathbf{g}_i/f + \Delta \cdot m_i$ encrypting binary polynomial $m_i \in \mathcal{R}_Q$. If $\mathsf{ct} = c_0 \boxdot_{i=1}^k \mathbf{c}_i$, then*

$$\mathsf{Var}(\mathsf{err}(\mathsf{ct})) \leq N \cdot l \cdot \gamma^2 \cdot \sum_{i=1}^{k} \mathsf{Var}(\mathbf{g}_i) + \mathsf{Var}(g_0) + 4 \cdot \zeta^2.$$

Moreover, if all ciphertexts, $c_0, \mathbf{c}_1, \ldots, \mathbf{c}_k$ are fresh, then

$$\mathsf{Var}(\mathsf{err}(\mathsf{ct})) \leq \left(4 + \left((k+1) \cdot N \cdot l \cdot \gamma^2\right)\right) \cdot \zeta^2.$$

To see this let c and $\mathbf{c}$ be as defined in Definition 2 such that c encrypts m_s and $\mathbf{c}$ encrypts m_v. Then the external product $\boxdot : \mathcal{R}_Q \times \mathcal{R}_Q \to \mathcal{R}_Q$ is defined as

$$c \boxdot \mathbf{c} = \mathfrak{g}^{-1}(c) \cdot \mathbf{c}.$$

3 Multi-key Homomorphic Encryption

To shift from the single key encryption scheme described in FINAL to a multi key version of the same scheme there are two components that have to be 'translated': the NGS-scheme, which is imperative for the bootstrapping mechanism, and the base LWE scheme.

3.1 Multi-key LWE

To instantiate a multi-key version of the LWE-scheme we must replicate the functionality of the LWE scheme when used under multiple keys. Such a scheme is not novel, e.g. [MW16, PS16], however introduces a few important concepts. Note that when $k = 1$, this is just the well-known single-key variant of LWE.

Definition 5 (Multi-Key LWE)**.** Given a security parameter λ, the Multi-Key LWE scheme consists of 5 probabilistic polynomial time algorithms: $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$, and NAND.

- $\mathsf{MKLWE.Setup}(\lambda)$: Upon input of a security parameter λ returns the public parameters (q, p, σ). Denote by $\Delta := q/4$.
- $\mathsf{MKLWE.KeyGen}(\sigma)$: Upon input of q, p, and σ, $\mathsf{MKLWE.KeyGen}$ randomly samples $s_i \in \mathbb{F}_2$, $i \in \{0, \ldots, n-1\}$ such that $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{F}_2^n$.
- $\mathsf{MKLWE.Enc}_{\mathbf{s}}(m)$: Upon input of a key $\mathbf{s}$ and a message $m \in \mathbb{F}_2$, $\mathsf{MKLWE.Enc}$ generates $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, and $e \leftarrow G_\sigma$. Then, it computes $b = \mathbf{a} \cdot \mathbf{s} + \Delta m + e$. This algorithm then outputs a vector of length $k+1$, $\mathsf{ct} = (b, 0, \ldots, 0, \mathbf{a}, 0, \ldots, 0)$ where $\mathbf{a}$ is in the position corresponding to the party generating the encryption, i.e. the i'th position for $\mathcal{P}_i$.
- $\mathsf{MKLWE.NAND}(\mathsf{ct}_1, \mathsf{ct}_2)$: Upon input of two ciphertexts of the form

  $$ct_i = (b_i, \mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,k}),$$

  $\mathsf{MKLWE.NAND}$ outputs

  $$\mathsf{ct} = \left(\frac{5q}{8} - b_1 - b_2, \mathbf{a}_{1,1} + \mathbf{a}_{2,1}, \ldots, \mathbf{a}_{1,k} + \mathbf{a}_{2,k}\right).$$

  We denote a ciphertext encrypting m under multiple keys (such as $\mathsf{sk}_1, \ldots, \mathsf{sk}_k$) as $\mathsf{MKLWE.Enc}_{\mathsf{sk}_1,\ldots,\mathsf{sk}_k}(m)$, which is generated through a NAND gate with ciphertexts encrypted under different keys.
- $\mathsf{MKLWE.Dec}(\mathbf{s}_1, \ldots, \mathbf{s}_k, \mathsf{ct})$: Upon input of keys $\mathbf{s}_1, \ldots, \mathbf{s}_k$ and ciphertext ct of the form $(b, \mathbf{a}_1, \ldots, \mathbf{a}_k)$ $\mathsf{MKLWE.Dec}$ computes

  $$m^* = \left\lfloor \frac{b - \sum_{i=1}^{k} (\mathbf{a}_i \cdot \mathbf{s}_i)}{\Delta} \right\rceil.$$

The MKLWE scheme is correct if

$$\mathsf{MKLWE.Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_k, (\mathsf{MKLWE.Enc}_{\mathsf{sk}_1,\ldots,\mathsf{sk}_k}(m)) = m.$$

Note that every fresh ciphertext is encrypted under a single key and only combined in the function evaluation. That this scheme is correct is clear, except that it remains to show that the $\mathsf{MKLWE.NAND}$ decrypts correctly.

Lemma 2 (Correctness of LWE-NAND)**.** *For two ciphertexts $\mathbf{c}_1$, $\mathbf{c}_2$ of the form $\mathbf{c}_i = (b_i, \mathbf{a}_1, \ldots, \mathbf{a}_k)$, $b_i = \sum_{i=1}^{k} (\mathbf{a}_i \cdot \mathbf{s}_i) + \Delta \cdot m_i + \mathbf{e}$ and secret keys, sk_i. Consider $\mathbf{c} = \mathsf{MKLWE.NAND}(\mathbf{c}_1, \mathbf{c}_2)$. If $\|\mathbf{e}_1 + \mathbf{e}_2\| \leq \frac{q}{4}$, then the protocol $\mathsf{MKLWE.Dec}_{\mathsf{sk}_1,\ldots,\mathsf{sk}_k}(\mathbf{c})$ outputs $\mathsf{NAND}(m_1, m_2)$.*

Proof. Let $\mathbf{c}$ be as above with the NAND-gate as described in Definition 5. Then

$$\begin{aligned}
&\mathsf{MKLWE.Dec}_{\mathsf{sk}_1,\ldots,\mathsf{sk}_k}(\mathbf{c}) \\
&\quad= \mathsf{MKLWE.Dec}_{\mathsf{sk}_1,\ldots,\mathsf{sk}_k}((\mathsf{MKLWE.NAND}(\mathbf{c}_1,\mathbf{c}_2)\ ) \\
&\quad= \mathsf{MKLWE.Dec}_{\mathsf{sk}_1,\ldots,\mathsf{sk}_k}\left(\frac{5q}{8} - b_1 - b_2, \mathbf{a}_{1,1} + \mathbf{a}_{2,1}, \ldots, \mathbf{a}_{1,k} + \mathbf{a}_{2,k}\right) \\
&\quad= \left(\frac{5q}{8} - b_1 - b_2, \mathbf{a}_{1,1} + \mathbf{a}_{2,1}, \ldots, \mathbf{a}_{1,k} + \mathbf{a}_{2,k}\right) \cdot (1, \mathsf{sk}_1, \ldots, \mathsf{sk}_k) \\
&\quad= \frac{5q}{8} - \Delta(m_1 - m_2) + \mathbf{e}_1 + \mathbf{e}_2.
\end{aligned}$$

Recall that $\mathsf{NAND}(m_1, m_2)$ can be written symmetrically as $(1 - \Delta \cdot m_1 m_2)\frac{q}{2}$, which means that the error produced by the above equation is given by

$$\frac{5q}{8} - \Delta(m_1 - m_2) + \mathbf{e}_1 + \mathbf{e}_2 - (1 - \Delta \cdot m_1 m_2)\frac{q}{2} = \pm\frac{q}{8} + \mathbf{e}_1 + \mathbf{e}_2.$$

Hence we can see that MKLWE.NAND is a regular encryption of $\mathsf{NAND}(m_0, m_1) = 1 - m_0 m_1$, producing an error of at most

$$\left|\pm\frac{q}{8} + \mathbf{e}_1 + \mathbf{e}_2\right| = \frac{q}{8} + \frac{q}{16} + \frac{q}{16} = \frac{q}{4}.$$

□

RLWE The ring-based variant of LWE, known as RLWE, is well-known and not all that different from LWE. We note here the major differences and notational changes, and refer you to [CCS19] for a more complete description. The main differences are as follows. We have our public values $\mathbf{a}_i, b \in \mathcal{R}_Q$. We denote party $\mathcal{P}_i$'s key by $z_i \in \mathcal{R}_Q$ instead of s_i. The coefficients of z_i and any error terms are sampled from χ_σ. We denote by $\mathbf{z} = (1, z_1, \ldots, z_k)$ and by $\langle \cdot, \cdot \rangle : \mathcal{R}_Q^n \times \mathcal{R}_Q^n \to \mathcal{R}_Q$ the external product between vectors over $\mathcal{R}_Q$. A multi-key RLWE ciphertext is then of the form $\mathbf{c} = (\mathbf{c}_0, \ldots, \mathbf{c}_k) \in R^{k+1}$ where decryption happens by computing $\langle \mathbf{c}, \mathbf{z} \rangle$ followed by rounding.

3.2 Multi-key NGS-scheme

The multi-key variant of the NGS scheme as introduced in FINAL, [BIP+22], is going to perform the same functionality as it does in the original paper. The NGS scheme will function as a transitional scheme which allows our scheme to implement an efficient bootstrapping procedure. It is clear to see in what follows that $k = 1$ produces (nearly identically) the single key NGS scheme (definition 2).

Definition 6 (Multi-key NGS)**.** Given a security parameter λ the multi-key NGS-scheme consists of four probabilistic polynomial time (PPT) algorithms: Setup, KeyGen, SEnc, VEnc.

- $\mathsf{MKNGS.Setup}(\lambda)$: Upon input of the security parameter λ this outputs a tuple (N, Q, μ, B, l), where B is a base used to decompose the ciphertexts and $l := \lfloor \log_B(Q) \rceil$.
- $\mathsf{MKNGS.KeyGen}(N, Q, \mu)$: Upon input of (N, Q, μ), samples $f_i' \leftarrow D_{\mathcal{R}}$ and computes $f_i = 4 \cdot f_i' + 1$ until f^{-1} exists in $\mathcal{R}_Q$. Output $\mathsf{sk}_i := f_i$.
- $\mathsf{MKNGS.SEnc}(f, m)$: Upon input of a secret key f and a message $m \in \mathbb{M}[X]$, samples random elements $g \leftarrow D_{\mathcal{R}}$ and outputs $c := \frac{g}{f} + \Delta \cdot m \in \mathcal{R}_Q$ where $\Delta = \lfloor \frac{q}{4} \rceil$. c is called a scalar encryption of m.

- $\mathsf{MKNGS.VEnc}(f, m)$: Upon input of a secret key f and a message $m \in \mathbb{M}[X]$, samples $g_i \leftarrow D_{\mathcal{R}}$ such that $\mathbf{g} = (g_0, \ldots, g_{l-1})$ and outputs $\mathbf{c} := \frac{\mathbf{g}}{f} + \mathfrak{g} \cdot m \in \mathcal{R}_Q^l$, where $\mathfrak{g} = \mathbf{G}_{1,Q,B} = (B^0, \ldots, B^{l-1})$. $\mathbf{c}$ is called a vector encryption of m.

Due to the nature of the Multi Key FINAL scheme the MKNGS scheme does not require multi-key encryption as the transitory encryption used during bootstrapping will always be under a single key. It is easy to verify that this defines the same functionality as the NGS-scheme, however the key interest of that scheme is that there exists an external product.

Furthermore, while the message space is ostensibly $\mathbb{M}[X]$, the messages in our scheme will always come from the message space of MKLWE. Thus, while we may handle ciphertexts that contain a polynomial as a message, we will only ever read the 0'th coefficient and we will expect this value to not be -1 at time of keyswitching to MKLWE.

A Note on Parameter Selection for NTRU-based Schemes As briefly mentioned in the introduction, there exist attacks [ABD16, CJL16, Dv21] on NTRU schemes with large parameters. Parameters susceptible to such attacks are called overstretched, and occur when, for modulus Q and dimension N, $Q > 0.004 \cdot N^{2.484}$ [Dv21]. In Section 5, we list a number of sets of parameters which are valid for our scheme without being overstretched.

3.2.1 Multi-Key External Product

The external product, [BIP+22], is a key feature in the bootstrapping procedure of FINAL and therefore needs to be replicated in the Multi-Key setting through adaptation of the processes described in [BIP+22, CCS19].

Given a scalar encryption $c = \sum_{i=1}^{k} \frac{g_i}{f_i} + \Delta \cdot m_s$ and a vector encryption $\mathbf{c} := \sum_{j=1}^{k} \frac{\mathbf{g}_j}{f_j} + \mathfrak{g} \cdot m_v$, the external product is defined as

$$c \boxdot \mathbf{c} = \mathfrak{g}^{-1}(c) \cdot \mathbf{c} = \mathfrak{g}^{-1}(c) \cdot \sum_{j=1}^{k} \left(\frac{\mathbf{g}_j}{f_j} \right) + \sum_{i=1}^{k} \left(\frac{g_i}{f_i} \right) \cdot m_v + \Delta \cdot m_s \cdot m_v.$$

It is easy to see that this is a correct scalar NGS encryption of $m = m_s \cdot m_v$ given the noise term is small enough.

For a scalar NGS ciphertext c and a sequence of vector NGS encryptions $\mathbf{c}_j$ for $j \in [t]$, we denote by

$$c \boxdot_{j=1}^{t} \mathbf{c}_j$$

the sequence of external products

$$((((c \boxdot \mathbf{c}_1) \boxdot \mathbf{c}_2) \boxdot \ldots) \boxdot \mathbf{c}_t).$$

3.2.2 Noise Analysis of (Multi-Key) External Products

As in FINAL, our analysis will be an average-case analysis, where the key assumption, that $\mathfrak{g}^{-1}(a)$ is γ-subgaussian for all $a \in \mathcal{R}_Q$, is identical. This allows us to evaluate the multi-key external product. First, we define the noise of a scalar and vector ciphertexts before we proceed with the analysis of the external product.

Definition 7 (Noise in a Scalar Multi-Key Ciphertext)**.** Let $c = \mathsf{MKNGS.SEnc}_{f_1,\ldots,f_n}(m) = \sum_{i=1}^{k} \frac{g_i}{f_i} + \Delta \cdot m$. Then the noise of c is defined as

$$\mathsf{err}(c) = c \cdot \prod_{i=1}^{k} f_i - \Delta \cdot m = \sum_{i=1}^{k} \left(g_i \prod_{j \neq i} f_j \right) + \prod_{i=1}^{k} (f_i - 1) \cdot \Delta \cdot m.$$

Similarly, we can define the noise of a vector ciphertext.

Definition 8 (Noise in a Vector Multi-Key Ciphertext)**.** Let $\mathbf{c} = \mathsf{MKNGS.VEnc}_{f_1,\ldots,f_k}(m) = \sum_{j=1}^{k} \frac{\mathbf{g}_j}{f_j} + \mathfrak{g} \cdot m$. Then the noise of $\mathbf{c}$ is defined as

$$\mathsf{err}(\mathbf{c}) = \mathbf{c} \cdot \prod_{i=1}^{k} f_i - \prod_{i=1}^{k} f_i \cdot \mathfrak{g} \cdot m = \sum_{i=1}^{k} \left(\mathbf{g}_i \prod_{j \neq i} f_j \right).$$

In the case of our instantiation of MKNGS using the FINAL distribution, we note that both $\mathbf{g}_i$ and f'_j are identically distributed, pairwise independent polynomials of at most degree N. Hence,

$$\mathsf{Var}\left(\sum_{i=1}^{k} \left(\mathbf{g}_i \prod_{j \neq i} f_j \right) \right) \leq k \cdot N^k \cdot \mathsf{Var}(4 \cdot f'_i + 1)^k \leq k \cdot \left(\sqrt{8} N \right)^k.$$

Lemma 3 (Noise Variance of a (Fresh) Multi-Key Scalar Ciphertext)**.** *Let*

$$c = \mathsf{MKNGS.SEnc}_{f_1,\ldots,f_k}(m) = \sum_{i=1}^{k} \frac{g_i}{f_i} + \Delta \cdot m \in \mathcal{R}_Q.$$

Then the variance of the error contained in c, $\mathsf{err}(c)$, is given by

$$\mathsf{Var}(\mathsf{err}(c)) \leq \mathsf{Var}\left(\sum_{i=1}^{k} \left(\prod_{j \neq i} f_j \cdot g_i \right) \right) + (4 \cdot d \cdot \mathsf{Var}(f'))^{k-1}$$

if m is a monomial in $\mathcal{R}_Q$ of the form $\alpha \cdot X^k$ for some $k \in \mathbb{Z}$ and $\alpha \in \mathbb{F}_2$, and

$$\mathsf{Var}(\mathsf{err}(c)) \leq \mathsf{Var}\left(\sum_{i=1}^{k} \left(\prod_{j \neq i} f_j \cdot g_i \right) \right) + (4 \cdot d \cdot \mathsf{Var}(f'))^{k-1} \cdot N$$

if m is a binary or ternary polynomial with $\deg(m) \leq N - 1$.

Proof. Recall that $\Delta = \frac{q}{4} + \epsilon$ for some $|\epsilon| < \frac{1}{2}$. Since $c \in \mathcal{R}_Q$ it holds that

$$c \cdot \prod_{i=1}^{k} f_i - \Delta \cdot m = \sum_{i=1}^{k} \left(g_i \cdot \prod_{i \neq j} f_j \right) + \prod_{i=1}^{k} (4 f'_j + 1) \cdot \epsilon \cdot m.$$

Note that $\prod_{i=1}^{k} (4 \cdot f'_j + 1) = 4^k f'_1 f'_2 \ldots f'_k + \ldots + 1$ and so

$$= \sum_{i=1}^{k} \left(g_i \cdot \prod_{i \neq j} f_j \right) + \left(4^k f'_1 f'_2 \ldots f'_k + \ldots + 1 \right) \cdot \epsilon \cdot m.$$

As $|\epsilon| < \frac{1}{2}$ variance of the noise is given by

$$\mathsf{Var}(\mathsf{err}(c)) \leq \mathsf{Var}\left(\sum_{i=1}^{k} \left(g_i \cdot \prod_{i \neq j} f_j \right) \right) + 4^k \mathsf{Var}(f'_1 \ldots f'_k) \cdot \|m\|_2^2$$

for a monomial $m = \alpha + X^t$ with $t \in \mathbb{Z}$ and $\alpha \in \mathbb{F}_2$, $\|m\|_2^2 < 1$ and for a binary or ternary polynomial for which $\deg(m) < N$, $\|m\|_2^2 \leq N$. Noting that each f'_i is a polynomial such that $\deg f'_i \leq d$ concludes the proof. □

Given these we can bound the noise of a fresh scalar ciphertext in MKNGS quite straightforwardly to $k \cdot \left(\sqrt{8}N\right)^k + (2 \cdot N)^k$ in the case that m is a monomial and $k \cdot \left(\sqrt{8}N\right)^k + (2 \cdot N)^k \cdot N$ otherwise.

Lemma 4 (Noise growth of a Multi-Key External Product)**.** *Let* $c := \sum_{i=1}^{k} \frac{g_i}{f_i} + \Delta \cdot u \in \mathcal{R}_Q$ *be a scalar encryption of* u *and let* $\mathbf{c} := \sum_{j=1}^{k} \frac{\mathbf{g}_j}{f_j} + \mathfrak{g} \cdot v$ *be a vector encryption of* $v \in \mathbb{F}_2$ *or* $v \in \{\pm b \cdot X^k : b \in \mathbb{F}_2, k \in \mathbb{N}\}$ *, such that each encryption uses at most* k *keys drawn from the set of keys* $\mathcal{F} = \{f_1, \ldots, f_k\}$*, i.e.* $\forall i, j \in \{1, \ldots, k\} : f_i, f_j \in \mathcal{F}$*. Then*

$$\mathsf{Var}(c \boxdot \mathbf{c}) = N \cdot l \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c})) + \mathsf{Var}(\mathsf{err}(c)).$$

Proof. From Section 3.2.1 we know that

$$c \boxdot \mathbf{c} = \mathfrak{g}^{-1}(c) \cdot \sum_{j=1}^{k} \left(\frac{\mathbf{g}_j}{f_j}\right) + \sum_{i=1}^{k} \left(\frac{g_i}{f_i}\right) \cdot v + \Delta \cdot u \cdot v.$$

By considering $u \cdot v = m$ this is a valid scalar NGS ciphertext. Thus by Lemma 3, we obtain that

$$\mathsf{err}(c \boxdot \mathbf{c}) = \mathsf{Var}\left(\mathfrak{g}^{-1}(c)\left(\sum_{i=1}^{k} \mathbf{g}_i \prod_{i \neq j} f_j\right) + \left(\sum_{i=1}^{k} g_i \prod_{i \neq j} f_j\right) \cdot \|v\|_2^2\right) + \left(4 \cdot d \cdot \mathsf{Var}(f')\right)^k$$

where m is a polynomial. Given the assumption that $\mathfrak{g}^{-1}(a)$ is γ-subgaussian l component decomposition into polynomials for any a and the fact that $\|v\|_2^2 < 1$ we obtain that

$$\mathsf{err}(c \boxdot \mathbf{c}) \leq N \cdot l \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c})) + \mathsf{Var}(\mathsf{err}(c))$$

which is exactly as claimed. □

Remark 1. Note that in Lemma 4, there is no limitation set on the noise terms given in c and $\mathbf{c}$ despite drawing the keys from the same key set. Thus it is possible to apply the lemma to ciphertexts which share no keys at all by letting $\mathcal{F}$ be the union of the associated sets of keys.

This lemma and the following observation immediately lead to the following corollary.

Corollary 1. *Let* $c_0 = \mathsf{MKNGS.SEnc}_{\mathsf{sk}_{1,0},\ldots,\mathsf{sk}_{k,0}}(m_0)$*, such that* $m_0 \in \mathbb{F}_2$*, and let* $\mathbf{c}_j = \mathsf{MKNGS.VEnc}_{\mathsf{sk}_{1,j},\ldots,\mathsf{sk}_{k,j}}(m_j)$ *such that for* $i, j \in \{1, \ldots, t\}$*,* $\mathsf{Var}(\mathsf{err}(\mathbf{c}_j)) = \mathsf{Var}(\mathsf{err}(\mathbf{c}_i))$*. Then*

$$\mathsf{Var}\left(\mathsf{err}(c \boxdot_{j=1}^{t} \mathbf{c}_j)\right) \leq t \cdot N \cdot l \cdot \gamma^2 \cdot \mathsf{Var}\left(\mathsf{err}(\mathbf{c}_j)\right) + \mathsf{Var}\left(\mathsf{err}(c)\right).$$

Proof. This follows immediately from Lemma 4 and Lemma 1. □

As a result of Corollary 1 and Remark 1 we observe that, given k players, each with t fresh single key Vector NGS-ciphertexts, $\mathbf{c}_{i,j}$, $i \in [\{1, \ldots, k\}]$, $j \in \{1, \ldots t\}$ such that $\mathbf{c}_{i,j} = \mathsf{NGS.VEnc}_{\mathsf{sk}_i}(m_j)$ and a single key Scalar NGS ciphertext $c_0 = \mathsf{NGS.SEnc}_{\mathsf{sk}_0}(m_0)$, the total noise of $\mathsf{ct} = c_0 \boxdot_{i,j=1}^{k,t} \mathbf{c}_{i,j}$ is given by $\mathsf{Var}\left(\mathsf{err}(\mathsf{ct})\right) = t \cdot k \cdot N \cdot l \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_{i,j})) + \mathsf{Var}(\mathsf{err}(c_0))$.

3.3 Key Switching

The penultimate step of the bootstrapping algorithm, see Section 4, will be a key switching from NGS to LWE. The protocol to perform this keyswitching, Π_{KS} defined in Figure 6, requires 3 pieces to get running. First, we require the key switching key defined originally in [BIP+22] which will be computed using Π_{kskGen} in Figure 5. Second, to compute this key switching key, we'll need a keyswitching protocol between different MKLWE keys. Third, and finally, we use the single key variant of the keyswitching protocol from NGS to LWE within our multi-key keyswitching protocol. In this section, we introduce all three of these preliminary parts required for Π_{KS}.

3.3.1 FINAL's key switching key

First, we recall key switching key used in [BIP+22] which proves to be useful in the multi-key space.

Definition 9 (Key switching key)**.** Let $c = \frac{g}{f} + \Delta m$ be an NGS scalar ciphertext such that $\phi(c) = (c_0, \ldots, c_{N-1})$ is the coefficient vector of c. Then the key switching key is given by

$$\mathsf{ksk} = (\mathbf{A}, b := \mathbf{A} \cdot \mathbf{s} + \mathbf{e} + G \cdot \phi(f)),$$

where $\mathbf{A} \in \mathbb{Z}_q^{(N \cdot l) \times n}$ and $\mathbf{e} \leftarrow \chi_{\sigma_e}^{N \cdot l}$.

3.3.2 Single-key keyswitching from NGS to LWE

Within our algorithm to keyswitch from multi-key NGS to multi-key LWE, we'll use the following function from [BIP+22].

Definition 10. Let $\mathsf{ksk} = (\mathbf{A}, b)$ and c be defined as above in definition 9. Let $\mathbf{y} = \mathfrak{g}^{-1}(\phi(c))$ and compute $\mathsf{KS}_{\mathsf{NGS}\to\mathsf{LWE}}(c, \mathsf{ksk}) := (\mathbf{y} \cdot \mathbf{A}, \mathbf{y} \cdot \mathbf{b})$.

This function takes as input an NGS ciphertext encrypting some message $m \in \mathbb{M}[X]$ (where the 0'th coefficient is in $\mathbb{F}_2$) under the key f and returns an LWE ciphertext encrypting $m_0 \in \mathbb{F}_2$ under key $\mathbf{s}$, where m_0 is m evaluated at 0. This is explicitly treated in [BIP+22] which shows the validity of the output.

3.3.3 Key switching within MKLWE

During our bootstrapping, we will use a slightly modified version of FINAL's key switching algorithm to convert a multi-key NGS ciphertext to a multi-key LWE ciphertext. However, we cannot directly compute key switching keys in the same way as FINAL due to the existence of multiple parties. Therefore, we introduce an MKLWE key switching algorithm in Figure 2 for use in our new key switching key generation algorithm in Section 4.4.

3.4 Hybrid Product

The hybrid product was introduced in [CCS19] and improved in [KMS24]. It allows, at the cost of significant noise growth, to securely compute a multiplication of an underlying message, x, with the secret key f_i of one of the parties. By repeated application of the hybrid product, this allows us to obtain an encryption of $x \cdot \prod_{i=1}^{n} f_i$. This product will be used in Section 4.4.1 to correctly compute the required key switching keys.

Definition 11 (Hybrid Product Key Generation)**.** $\mathsf{HPKGen}(z_i, f_i)$: on input (z_i, f_i) by the i-th player, where $z_i \leftarrow \chi_\sigma^n$ is their RLWE key and $f_i \leftarrow D_{\mathcal{R}}$ is their NGS key, sample

MKLWE **Key Switching protocol,** Π_{KSLWE}

Input: An MKLWE ciphertext $\mathbf{c}$, k Key switching keys $(\mathsf{ksk}_{\mathsf{LWE}})_i = (\mathbf{b}_{i,j}, \mathbf{A}_{i,j})$.

Output: MKLWE ciphertext $\mathbf{c}' = (b', \mathbf{a}'_1, \ldots, \mathbf{a}'_k)$.

1. Parse $\mathbf{c}$ as $(b, \mathbf{a}_1, \ldots, \mathbf{a}_k)$ and $\mathbf{a}_i = (a_{i,1}, \ldots, a_{i,N})$.
2. For $i \in [k], j \in [N]$
   (a) $b'_i = \sum_{j=1}^{N} \mathfrak{g}^{-1}(a_{i,j})^T \cdot \mathbf{b}_{i,j}$
   (b) $\mathbf{a}'_i = \sum_{j=1}^{N} \mathfrak{g}^{-1}(a_{i,j})^T \cdot A_{i,j}$

Figure 2: The MKLWE Key switching protocol as introduced in [CCS19].

noise vectors $\mathbf{e}_{i,0}, \mathbf{e}_{i,1}, \mathbf{e}_{i,2} \leftarrow \chi_\mu^n$. Generate random vector $\mathbf{d}_{i,1} \leftarrow \chi_\mu^n$ and random RLWE key r_i and compute

$$\mathbf{b}_i = -z_i \cdot \mathbf{a} + \mathbf{e}_{i,0}; \quad \mathbf{d}_{i,0} = -z_i \cdot \mathbf{d}_{i,1} + r_i \cdot \mathfrak{g} + \mathbf{e}_{i,1}; \quad \mathbf{d}_{i,2} = r_i \cdot \mathbf{a} + f_i \cdot \mathfrak{g} + \mathbf{e}_{i,2},$$

where $\mathbf{a}$ is a common reference string (CRS). Output $\mathbf{b}_i$ and $hpk_i = (\mathbf{d}_{i,0}, \mathbf{d}_{i,1}, \mathbf{d}_{i,2})$.

To get an intuitive idea of what the hybrid product keys achieve it is easy to see that they behave like LWE ciphertexts, such that $\mathbf{b}_i$ encrypts 0 under the key z_i, $\mathbf{d}_{i,0}$ encrypts $r_i \cdot \mathfrak{g}$ under the key z_i, and $\mathbf{d}_{i,2}$ encrypts $f_i \cdot \mathfrak{g}$ under the key r_i.

Definition 12 (Hybrid Product)**.** $\mathsf{HybridProduct}(\mathbf{c}, \mathsf{hpk}_i, \{\mathbf{b}_j\}_{j\in\{1,\ldots,k\}})$: on input of the hybrid product keys and RLWE ciphertext $\mathbf{c} = (\mathbf{c}_0, \ldots, \mathbf{c}_k) \in R^{k+1}$, for $j \in \{0, \ldots, k\}$, let $\mathbf{b}_0 = -\mathbf{a}$ and compute:

$$\mathbf{u}_{i,j} = \langle \mathfrak{g}^{-1}(\mathbf{c}_j), \mathbf{d}_{i,2} \rangle; \qquad \mathbf{v}_i = \langle \Sigma_{j=0}^{k} \mathfrak{g}^{-1}(\mathbf{c}_j), \mathbf{b}_i \rangle.$$

Let $\mathbf{c}'_j = \mathbf{u}_{i,j}$ for $j \in [k]$. Now let $\mathbf{c}'_0 = \mathbf{c}'_0 + \langle \mathfrak{g}^{-1}(\mathbf{v}_i), \mathbf{d}_{i,0} \rangle$ and $\mathbf{c}'_i = \mathbf{c}'_i + \langle \mathfrak{g}^{-1}(\mathbf{v}_i), \mathbf{d}_{i,1} \rangle$. Then, return the MK-RLWE ciphertext $\mathbf{c}' = (\mathbf{c}'_0, \ldots, \mathbf{c}'_k) \in \mathcal{R}^{(k+1)}$.

The following lemma proves that on input of an MKLWE ciphertext encrypting x, the hybrid product outputs an MKLWE ciphertext encrypting $x \cdot f_i$ and bounds the noise growth.

Lemma 5. *Let* $\mathbf{c} = (\mathbf{c}_0, \ldots, \mathbf{c}_{i-1}, 0, \ldots, 0) \in R^{k+1}$ *such that* $\langle \mathbf{c}, \mathbf{z} \rangle = x + e$ *where* $\mathbf{z} = (1, z_1, \ldots, z_k) \in R^{k+1}$*. Let* hpk_i *and* $\{b_j\}_{j\in\{1,\ldots,k\}}$ *be computed by* HPKGen*. Then* $\mathbf{c}' := \mathsf{HybridProduct}(\mathbf{c}, \mathsf{hpk}_i, \{b_j\}_{j\in\{1,\ldots,k\}})$ *has the form* $\mathbf{c}' = (c'_0, c'_1, \ldots, c'_i, 0, \ldots, 0)$*. Furthermore, the following holds*

$$\langle \mathbf{c}', \mathbf{z} \rangle =: x \cdot f_i + f_i \cdot e + e', \tag{1}$$

where $\mathbf{z} = (1, z_1, \ldots, z_k) \in R^{k+1}$*. The new error is bounded by*

$$\begin{aligned} \mathsf{Var}(e') &= (2 + 2 \cdot (i-1) \cdot N \cdot l \cdot \mathsf{Var}(f)) \cdot \gamma^2 \cdot \mu^2 \\ &\leq 2i \cdot N \cdot l \cdot \mathsf{Var}(f) \cdot \gamma^2 \cdot \mu^2 \leq 16i \cdot N \cdot l \cdot \gamma^2 \cdot \mu^2. \end{aligned}$$

Proof. We first prove that $c'_j = 0$ when $j > i$. Next, we prove equation 1. Finally, we show the value of e'.

Let $k \geq j > i$. Then,

$$c'_j = \mathbf{u}_{i,j} = \langle \mathfrak{g}^{-1}(c_j), \mathbf{d}_{i,2} \rangle = \langle 0, \mathbf{d}_{i,2} \rangle = 0.$$

Next, we compute

$$\begin{aligned}\langle \mathbf{c}', \mathbf{z}\rangle =& \mathbf{u}_{i,0} + \Sigma_{j=1}^{k} z_j \cdot \mathbf{u}_{i,j} + \langle \mathfrak{g}^{-1}(\mathbf{v}_i), \mathbf{d}_{i,0}\rangle + \mathbf{z}_i \cdot \langle \mathfrak{g}^{-1}(\mathbf{v}_i), \mathbf{d}_{i,1}\rangle \\ =& f_i \cdot \langle \mathbf{c}, \mathbf{z}\rangle + \langle \mathfrak{g}^{-1}(c_0), e_{i,2}\rangle + \Sigma_{j=1}^{i-1} z_i \langle \mathfrak{g}^{-1}(c_j), e_{i,2}\rangle + \langle \mathfrak{g}^{-1}(v_i), e_{i,1}\rangle \\ &+ \Sigma_{j=1}^{i-1} \langle \mathfrak{g}^{-1}(c_j), e_{i,0} \cdot r_i\rangle.\end{aligned}$$

Thus, equation 1 trivially holds. Furthermore, for all $i \in \{1, \ldots, k\}$, it holds that

$$\mathsf{Var}(e') = \mathsf{Var}(\mathfrak{g}^{-1}(c_0)\mathbf{e}_{i,2} + \mathfrak{g}^{-1}(v_i)\mathbf{e}_{i,1} + \Sigma_{j=1}^{i-1} r_i \mathfrak{g}^{-1}(c_j)\mathbf{e}_{i,0} + \Sigma_{j=1}^{i-1} z_j \cdot \mathfrak{g}^{-1}(c_j)\mathbf{e}_{i,2})$$

As $\mathbf{e}_{i,j}$ is a vector and $\mathfrak{g}^{-1}(a)$ is a vector of length at most l for any a, the simplification

$$\mathsf{Var}(e') = 2 \cdot \gamma^2 \cdot \mu^2 + (i-1) \cdot \gamma^2 \cdot \mu^2 \cdot \mathsf{Var}(r_i) \cdot \zeta_1 + (i-1) \cdot \gamma^2 \cdot \mu^2 \cdot \mathsf{Var}(z_j) \cdot \zeta_2$$

can immediately be made, where ζ_1 and ζ_2 are correction factors depending on r_i and z_j. Specifically, as r_i and z_j are both keys drawn according to the FINAL distribution they are both polynomials, therefore $\zeta_1 = \zeta_2 = N \cdot l$ and so

$$\begin{aligned}\mathsf{Var}(e') &= (2 + 2 \cdot (i-1) \cdot N \cdot l \cdot \mathsf{Var}(f)) \cdot \gamma^2 \cdot \mu^2 \\ &\leq 2i \cdot N \cdot l \cdot \mathsf{Var}(f) \cdot \gamma^2 \cdot \mu^2 \leq 16i \cdot N \cdot l \cdot \gamma^2 \cdot \mu^2\end{aligned}$$

where the final inequality holds as $\mathsf{Var}(f) = \mathsf{Var}(4 \cdot f' + 1) = 8$. □

3.5 Modulus Switching

As the last step of our bootstrapping, we will need to switch the modulus of an MKLWE ciphertext. Let the randomized rounding function [DM15] be as follows.

Definition 13 (Randomized Rounding Function)**.** Let $Q, q \in \mathbb{Z}$, and $1 < q < Q$. The randomized rounding function $[\cdot]_{Q:q} : \mathbb{Z}_Q \to Z_q$ is defined as $[z]_{Q:q} = \lfloor q \cdot z/Q \rfloor + B$ where $B \in \{0, 1\}$ is a Bernoulli random variable with $\Pr[B = 1] = (q \cdot z/Q) - \lfloor q \cdot z/Q \rfloor \in \{0, 1\}$

This definition naturally extends to vectors, matrices and polynomials by performing rounding element-wise. Note also that the rounding error ϵ is $1-$subgaussian.

Modulus switching from Q to q is done by applying the Randomized Rounding Function to the LWE ciphertext. We denote $\mathsf{ModSwitch}(c, Q, q) = [c]_{Q:q}$. Furthermore, the following lemma holds regarding the noise growth of this operation.

Lemma 6. *For any* MKLWE *ciphertext* $\mathbf{c}$ *of message* m *under keys* $\{\mathbf{s}_i\}_{1 \leq i \leq k}$ *with noise* σ *and modulus* Q *and* q*,* $\mathsf{ModSwitch}(\mathbf{c}, Q, q)$ *is an* MKLWE *ciphertext of message* m *under keys* $\{\mathbf{s}_i\}_{1 \leq i \leq k}$ *with modulus* q *and noise at most*

$$\frac{q^2}{Q^2}\sigma + (kn+1)\sqrt{2\pi}.$$

Proof. Let $\mathbf{c} = (b, \mathbf{a})$ and $\mathsf{ModSwitch}(\mathbf{c}) = (b', \mathbf{a}')$. We have that for each $i \in \{1, \ldots, k\}$ and each $j \in \{1, \ldots, n\}$, $a'_{i,j} = \frac{q}{Q} a_{i,j} + r_{i,j}$ and $b' = \frac{q}{Q} b + r_0$ for independent $\sqrt{2\pi}$-subgaussian rounding errors $r_0, r_{1,1}, r_{1,2}, \ldots, r_{k,n}$. It follows that c' is a MKLWE encryption of m with error

$$\begin{aligned}\mathsf{err}(\mathbf{c}') =& b' - \Sigma_{i=1}^{k} \mathbf{a}'_{\mathbf{i}} \cdot \mathbf{s_i} - qm \\ =& \frac{q}{Q}\mathsf{err}(\mathbf{c}) + r_0 - \Sigma_{i=1}^{n}\Sigma_{j=1}^{k} r_{i,j} \\ =& \frac{q}{Q}\mathsf{err}(\mathbf{c}) + R\end{aligned}$$

where R is $(kn+1)\sqrt{2\pi}$-subgaussian. Thus $\mathsf{Var}(\mathsf{err}(\mathbf{c}')) \leq \frac{q^2}{Q^2}\mathsf{Var}(\mathsf{err}(\mathbf{c})) + (kn+1)\sqrt{2\pi}$. □

Bootstrapping key generator Π_{BSK}

Input: LWE Secret Key $\mathbf{s}_i$ belonging to a player $\mathcal{P}_i$

Output: Bootstrapping key bsk_i

1. $\mathbf{s}_i = (s_{i,0}, \ldots, s_{i,n-1})$
2. for $j \in \{0, \ldots, n\}$ do
   $\mathsf{bsk}_{i,j} \leftarrow \mathsf{NGS.VEnc}_{f_i}(\mathbf{s}_{i,j})$
3. $\mathsf{bsk}_i = (\mathsf{bsk}_{i,0}, \ldots, \mathsf{bsk}_{i,n-1})$

Figure 3: The bootstrapping key generator, Π_{BSK}, using MKLWE-style binary keys.

4 Bootstrapping

After the MKLWE scheme has produced a ciphertext encrypting $m = f(m_1, \ldots, m_k)$ using the MKLWE scheme's NAND gates, the next step is reducing the noise through a bootstrapping procedure. To this end, assume that $\mathsf{ct} := (b, \mathbf{a}_1, \ldots, \mathbf{a}_k)$ is an MKLWE ciphertext where $b = \sum_{i=1}^{k} \mathbf{a}_i \cdot \mathbf{s}_i + \Delta m + e$ where the ciphertext error, e, is E-subgaussian for an appropriate E.

The bootstrapping procedure consists of three steps. First, we take this ct as input and convert this to an MKNGS Scalar Ciphertext of the form $c = \sum_{i=1}^{k} \frac{g_i}{f_i} + \Delta \cdot m$. Second, we keyswitch this MKNGS ciphertext and keyswitch back to an MKLWE ciphertext. Third, as the ciphertext output by the keyswitching will still have the MKNGS modulus, we ModSwitch back to the MKLWE modulus. The bootstrapping algorithm is defined in detail in Figure 4.

4.1 Generating Bootstrapping Keys

Each party needs to provide the server with a bootstrapping key to allow for the conversion from MKLWE to MKNGS. To achieve this we define a new bootstrapping key generation algorithm, Π_{BSK}, in Figure 3.

Every party, $\mathcal{P}_i$, has an LWE key, $\mathbf{s}_i$, and an NGS key, f_i. To obtain the bootstrapping key for party $\mathcal{P}_i$, bsk_i, we simply encrypt each coefficient of $\mathbf{s}_i$, $s_{i,j}$, under the appropriate party's NGS key to obtain

$$\mathsf{bsk}_i = (\mathsf{bsk}_{i,j})_{j \in \{0,\ldots,n-1\}} = \left(\mathsf{NGS.VEnc}_{f_i}(s_{i,j})\right)_{j \in \{0,\ldots,n-1\}}.$$

Each party can pre-process the generation of these keys at the start of the protocol. As the bootstrapping keys, $\mathsf{bsk}_{i,j}$, are fresh NGS vector encryptions we conclude that $\mathsf{Var}(\mathsf{err}(\mathsf{bsk}_{i,j})) = \mathsf{Var}(\mathbf{g}) = \max_j \mathsf{Var}(g_j)$.

4.2 Adopting the Binary CMux

The first step of our bootstrapping is converting our input, an MKLWE ciphertext, to an MKNGS ciphertext. We do this using a series of binary CMux gates. The binary CMux gate is well described, [CGGI20], and is given by

$$\mathsf{CMux}_{i,j}(c) = \mathfrak{g} + (X^c - 1) \cdot \mathsf{NGS.VEnc}_{f_i}(s_{i,j}),$$

where $\mathfrak{g}$ is a noiseless encryption of 1. Note that the encryption of $s_{i,j}$ is a single key vector NGS encryption under the key f_i. Note also that

$$\mathsf{CMux}_{i,j}(c_{i,j}) = \mathfrak{g}(1 + X^{c_{i,j}} s_{i,j} - s_{i,j}) + \frac{\mathbf{g}}{f_i}(X^{c_{i,j}} - 1) = \mathsf{NGS.VEnc}_{f_i}(X^{c_{i,j} s_{i,j}}).$$

The Bootstrapping Algorithm, Π_{BS}

Input: $\mathsf{ct} = (b, \mathbf{a_1}, \ldots, \mathbf{a_k})$, an MKLWE ciphertext encrypting m.

Output: ct', an MKLWE ciphertext encrypting the same m.

1. Parse $\mathsf{ct} = (\mathsf{ct}_0, \ldots, \mathsf{ct}_k)$
2. $(c_0) \leftarrow \left\lfloor \frac{4 \cdot N \cdot \mathsf{ct}_0}{q} \right\rceil$
3. $(\mathbf{c_1}, \ldots, \mathbf{c_k}) \leftarrow \left(\left\lfloor \frac{4 \cdot N \cdot \mathsf{ct}_i}{q} \right\rceil \right)_{i \in \{1,\ldots,k\}}$
4. $\mathsf{ACC} \leftarrow \left\lfloor \frac{Q}{8} \right\rceil \cdot X^{N/2} \cdot \sum_{i=0}^{N-1} X^i$
5. $\mathsf{ACC} \leftarrow \mathsf{ACC} \boxdot \mathfrak{g} \cdot X^{c_0}$
6. for $i \leftarrow 1$ to k do
   (a) for $j \leftarrow 0$ to $n-1$ do
      i. $c_{mux} \leftarrow \mathsf{CMux}_{i,j}(c_{i,j})$
      ii. $\mathsf{ACC} \leftarrow \mathsf{ACC} \boxdot c_{mux}$
7. $\mathsf{ACC} \leftarrow \mathsf{ACC} + \left\lfloor \frac{Q}{8} \right\rceil \cdot \sum_{i=0}^{N-1} X^i$
8. $\mathsf{ACC} \leftarrow \Pi_{\mathsf{KS}}(\mathsf{ACC}, (\mathsf{ksk}_0, \ldots, \mathsf{ksk}_k))$
9. $\mathsf{ct}' \leftarrow \mathsf{ModSwitch}(\mathsf{ACC}, Q, q)$

Figure 4: Multi-Key NGS Bootstrapping algorithm in the style of FINAL.

Where the final encryption of $X^{c_{i,j} s_{i,j}}$ under f_i has noise $\mathbf{g}(X^{c_{i,j}} - 1)$. As $s_{i,j} \in \mathbb{F}_2$ it can easily be observed that the final equality holds:

- If $s_{i,j} = 0$ then $X^{c_{i,j} s_{i,j}} = X^0 = 1$ while $1 + X^{c_{i,j}} s_{i,j} + s_{i,j} = 1$.
- If $s_{i,j} = 1$ then $X^{c_{i,j} s_{i,j}} = X^{c_{i,j}}$, while $1 + X^{c_{i,j}} s_{i,j} - s_{i,j} = X^{c_{i,j}}$.

Thus we have that the message encrypted by $\mathsf{CMux}_{i,j}(c_{i,j})$ lies in the message space $\mathbb{M}$.

Now let E_{bsk_i} be the error of the bootstrapping key used in the binary CMux gate, the noise of the CMux gate can be easily computed.

Lemma 7. *Let* $E_{\mathsf{bsk}_i} = \mathsf{Var}(\mathsf{err}(\mathsf{bsk}_i))$ *and let* $E_{\mathsf{CMux}_{i,j}}$ *be the noise of the output of the* CMux *gate. Then* $E_{\mathsf{CMux}_{i,j}} \leq 2 \cdot E_{\mathsf{bsk}_{i,j}}$.

Proof. First note that $\mathsf{bsk}_i = (\mathsf{bsk}_{i,j})_{j \in \{0,\ldots,n-1\}}$ and so $E_{\mathsf{bsk}_i} = \max_j \left(E_{\mathsf{bsk}_{i,j}} \right)$. Since all $\mathsf{bsk}_{i,j} = \mathsf{NGS.VEnc}_{f_i}(s_{i,j})$ are fresh ciphertexts, that means $\mathsf{Var}(\mathsf{err}(\mathsf{bsk}_{i,j})) = \mathsf{Var}(g)$. Hence, as the message contained in $\mathsf{CMux}_{i,j}(c_{i,j})$ is given by $1 + X^{c_{i,j} s_{i,j}} - s_{i,j}$ we have

$$\begin{aligned} \mathsf{Var}(\mathsf{err}(\mathsf{CMux}_{i,j}(c_{i,j}))) &= \mathsf{Var}\left(\frac{\mathbf{g}}{f_i} (X^{c_{i,j}} - 1) \right) \\ &= \mathsf{Var}(\mathbf{g}) \cdot \| X^{c_{i,j}} - 1 \|_2^2 \\ &\leq 2 \cdot \mathsf{Var}(\mathbf{g}) \end{aligned}$$

and as $\mathsf{Var}(\mathbf{g})$ is exactly the error of the bootstrapping key, we obtain the desired noise. □

4.3 Applying the CMux Gate

In Figure 4 we describe the bootstrapping protocol in full. Given our input $\mathsf{ct} = (ct_0, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, we can pre-process this to obtain $(c_0, \mathbf{c}_1, \ldots, \mathbf{c}_k)$ such that

$$c_0 = \left\lfloor \frac{4 \cdot N \cdot ct_0}{q} \right\rceil, \mathbf{c}_i = \left\lfloor \frac{4 \cdot N \cdot (-\mathsf{ct}_i)}{q} \right\rceil, \text{ where } i \in \{1, \ldots, k\}.$$

Now set $\tilde{\mathbf{c}}_0 = \mathfrak{g} \cdot X^{c_0}$, which can be seen as a noiseless Vector NGS encryption of X^{c_0}. We would like to apply the CMux gates to obtain the result of the following useful multiplication:

$$\begin{aligned}
\tilde{\mathbf{c}} \cdot \prod_{\substack{i \in [k] \\ j \in [n-1]}} \mathsf{CMux}_{i,j}(c_{i,j}) &= \mathfrak{g} \cdot X^{c_0} \cdot \prod_{\substack{i \in [k] \\ j \in [n-1]}} \left(\mathfrak{g}(X^{c_{i,j} s_{i,j}}) + \frac{\mathbf{g}}{f_i}(X^{c_{i,j}} - 1) \right) \\
&= \mathsf{NGS.VEnc}_{f_1, \ldots, f_n} \left(X^{\sum_{(i,j)} c_{i,j} s_{i,j}} \right) \\
&= \mathsf{NGS.VEnc}_{f_1, \ldots, f_n} \left(X^{\Delta \cdot m + e} \right).
\end{aligned}$$

Note that the $\mathbf{c}_i$ defined in step 1 and 2 of Π_{BS} are polynomials of at most degree n. Therefore, the total number of elements processed in the CMux gates is $(k \cdot n) + 1$.

However, simply computing this multiplication is not directly possible as we have no procedure to multiply two vector ciphertexts together. Instead, we use the external products defined in Section 3.2 with an accumulator $\mathsf{ACC} := \left\lfloor \frac{Q}{8} \right\rceil \cdot X^{N/2} \cdot \sum_{i=0}^{N-1} X^i \pmod{X^N + 1}$ on the left. This accumulator can be observed as a noiseless Scalar NGS encryption and therefore allows for the use of the external products. Using this accumulator, as is common in [GPV23, MS18, CGGI20], we obtain the following statement:

Theorem 1. *Let* $\mathsf{ct} = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k)$ *and let* $c_0, \mathbf{c}_1, \ldots, \mathbf{c}_k$, bsk_i, *and* ACC *be as defined above. Then*

$$\mathbf{c}' := (\mathsf{ACC} \boxdot \tilde{\mathbf{c}}_0) \boxdot_{i,j} \mathsf{CMux}_{i,j}(c_{i,j}) + \left\lfloor \frac{Q}{8} \right\rceil \cdot \sum_{i=0}^{N-1} X^i = \mathsf{NGS.SEnc}_{f_1, \ldots, f_k}(m),$$

with $\Delta = \frac{Q}{4}$, *such that* $\mathsf{Var}(\mathsf{err}(\mathbf{c}') \leq (n \cdot k + 1) \cdot N \cdot l \cdot \gamma^2$.

Proof. We first show correctness, then we will bound the noise of $\mathbf{c}'$. Let $\mathsf{ct} = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k)$ be a valid MKLWE ciphertext and let $c_0, \ldots, \mathbf{c}_k$ be as given. Then let the accumulator, ACC, be defined as

$$\mathsf{ACC} = \left\lfloor \frac{Q}{8} \right\rceil \cdot X^{N/2} \cdot \sum_{i=0}^{N-1} X^i$$

such that we can use the external product in the first step to obtain

$$\mathsf{ACC} \boxdot \mathfrak{g} \cdot X^{c_0} = \left\lfloor \frac{Q}{8} \right\rceil \cdot X^{N/2 + c_0} \cdot \sum_{i=0}^{N-1} X^i.$$

Then, applying the CMux in repetition over all pairs (i, j), $i \in [k]$, $j \in [n]$, in an identical way we obtain

$$\mathsf{ACC} = \left\lfloor \frac{Q}{8} \right\rceil \cdot X^{N/2} \cdot \sum_{i=0}^{N-1} X^i \cdot X^{\lfloor \frac{4N}{q} c_0 \rceil} \cdot \prod_{i=1}^{n} X^{\lfloor \frac{4N}{q} c_i s_i \rceil}.$$

Now consider $c_0 - \sum_{i=1}^{n} c_i s_i = \Delta \cdot m + e$. Since $\Delta = \lfloor \frac{q}{4} \rceil$ we obtain that

$$\frac{4 \cdot N}{q} (\Delta \cdot m + e) = N \cdot \left(m + \frac{e}{\Delta} \right).$$

By assumption on the error bound $\left\lfloor \frac{e}{\Delta} \right\rceil = 0$ and so we obtain the following:

$$\mathsf{ACC} = \left\lfloor \frac{Q}{8} \right\rceil \cdot X^{N/2} \cdot \sum_{i=0}^{N-1} X^i \cdot (-1)^m.$$

Recall from Section 3.2 they we are only interested in the 0'th coefficient. We observe that the 0'th coefficient occurs at $i = \frac{N}{2}$, obtaining $X^{N/2} \cdot X^{N/2} = X^N = -1$, resulting in

$$\mathsf{ACC} = \left\lfloor \frac{Q}{8} \right\rceil - 1 \cdot (-1)^m.$$

It is easily observed that this is a valid encryption, $\mathsf{MKNGS.SEnc}(2m-1)$, therefore by adding a noiseless MKNGS encryption of 1, $\mathsf{MKNGS.SEnc}(1) = \frac{Q}{8} \cdot \sum_{i=1}^{N-1} X^i$, we obtain

$$\mathsf{ACC} = \mathsf{MKNGS.SEnc}(2m)$$

with $\Delta = \frac{Q}{8}$. Equivalently, this can be seen as $\mathsf{ACC} = \mathsf{MKNGS.SEnc}(m)$ with $\Delta = \frac{Q}{4}$, which is the format we desired.

We now bound the noise generated by $\mathbf{c}'$. As there are $(n \cdot k) + 1$ total CMux multiplications, we obtain from Corollary 1 that the noise of this sequence is bounded by:

$$\mathsf{Var}\left(\mathsf{err}(\mathbf{c}')\right) \leq (n \cdot k + 1) \cdot N \cdot l \cdot \gamma^2 \cdot \mathsf{Var}\left(\mathsf{err}(\mathsf{CMux}_{i,j}(c_{i,j}))\right) + \mathsf{Var}\left(\mathsf{err}(T(x))\right).$$

By Lemma 7 we note that $\mathsf{Var}\left(\mathsf{err}(\mathsf{CMux}_{i,j}(c_{i,j}))\right) \leq 2 \cdot \mathsf{Var}(g)$, hence

$$\begin{aligned}\mathsf{Var}\left(\mathsf{err}(\mathbf{c}')\right) \leq & (n \cdot k + 1) \cdot N \cdot l \cdot \gamma^2 \cdot 2 \cdot \mathsf{Var}(g) + \mathsf{Var}\left(\mathsf{err}(T(X))\right) \\ \leq & (n \cdot k + 1) \cdot N \cdot l \cdot \gamma^2\end{aligned}$$

where the last inequality holds as $\mathsf{Var}(g) = \frac{1}{2}$ by the definition of the FINAL distribution and $T(X)$ being a noiseless encryption. □

4.4 Switching back to LWE

Having obtained the result of the CMux, we must now switch the ciphertext and the keys back from MKNGS to MKLWE. To do this, we adapt the single-key key switching algorithm from FINAL [BIP+22], see also Section 3.3.

The main adaptation that we need to introduce is that we use MKLWE instead of single key LWE, which means that the key switching key has to have noise entries for each of the parties, hence $\mathsf{ksk} = (\mathsf{ksk}_0, \ldots, \mathsf{ksk}_k)$ where $\mathsf{ksk}_0 := \sum_{i=1}^{k} \mathsf{ksk}_i \cdot s_i + e + \mathfrak{g} \cdot \prod_j f_j$. However, first we must generate the key switching keys themselves.

4.4.1 Generating the key switching keys:

To instantiate this, assume the use of a CRS, denoted here by $\mathbf{a}$, which is used as implicit input in what follows. Each player, P_i, starts by locally generating the hybrid product keys $\mathbf{b}_i$ and $\mathsf{hpk}_i = (\mathbf{d}_{i,0}, \mathbf{d}_{i,1}, \mathbf{d}_{i,2})$ by computing $\mathsf{HPKGen}(z_i, f_i)$ as outlined in Section 3.4. Each player must also compute $\mathsf{ksk}_{\mathsf{LWE}i}$ as in Π_{KSLWE} in Section 3.3.

These are then public values which can be sent to the server, allowing it to compute ksk_0. To do this, let

$$\mathbf{x}_1 := (\mathbf{d}_{1,2} + \langle \mathfrak{g}^{-1}(-\mathbf{a}), \mathbf{d}_{1,0} \rangle, \langle \mathfrak{g}^{-1}(-\mathbf{a}), \mathbf{d}_{1,1} \rangle, 0, \ldots, 0) \in \mathcal{R}_Q^{k+1}$$

be an RLWE ciphertext encrypting $\mathfrak{g} \cdot f_1$ under the keys $\{z_i\}_{1 \leq i \leq k}$. The server computes $\mathbf{x}_i := \mathsf{HybridProduct}\left(\mathbf{x}_{i-1}, \mathsf{hpk}_i, \{\mathbf{b}_j\}_{j \in [k]}\right)$. for each $i \in \{2, \ldots, k\}$ until it obtains $\mathbf{x}_k$.

Key switching key generation, Π_{kskGen}

Input: b_i, $\mathsf{hpk}_i = (d_{i,0}, d_{i,1}, d_{i,2})$, and $ksk_{LWE,i}$

Output: The key switching key: $(\mathsf{ksk}_0, \ldots, \mathsf{ksk}_k)$.

1. $\mathbf{x}_1 \leftarrow (\mathbf{d}_{1,2} + \langle \mathfrak{g}^{-1}(-\mathbf{a}), \mathbf{d}_{1,0} \rangle, \langle \mathfrak{g}^{-1}(-\mathbf{a}), \mathbf{d}_{1,1} \rangle, 0, \ldots, 0) \in R_Q^{k+1}$
2. for $i \in \{2, \ldots, k\}$ do:
   - $\mathbf{x}_i \leftarrow \mathsf{HybridProduct}(\mathbf{x}_{i-1}, \mathsf{hpk}_i, \{\mathbf{b}_j\}_{1 \le j \le k})$
3. Parse $\mathbf{x}_k := (\mathbf{b}, \mathbf{a}_1, \ldots, \mathbf{a}_k) \in \mathbb{Z}^{N(k+1)}$.
4. $\mathbf{x}'_k \leftarrow (b_0, \mathbf{a}_1, \ldots, \mathbf{a}_k) \in \mathbb{Z}^{kN+1}$
5. $\mathsf{ksk} \leftarrow \Pi_{\mathsf{KSLWE}}(\mathbf{x}'_k, \{\mathsf{ksk}_{LWE,i}\}_{1 \le i \le k})$
6. **Out:** $(\mathsf{ksk}_0, \ldots, \mathsf{ksk}_k) := \mathsf{ksk}$

Figure 5: Computing the Key switching Keys

By Lemma 5, at the end of the Hybrid Product chain we obtain a multi-key RLWE ciphertext $(\mathbf{c}'_0, \mathbf{c}'_1, \ldots, \mathbf{c}'_k) \in \mathcal{R}_Q^{k+1}$, where c'_0 contains a term $\mathfrak{g} \cdot \Pi_{i=1}^k f_i$ in the zero'th coefficient. To obtain an MKLWE ciphertext encrypting $\mathfrak{g} \cdot \Pi_{i=1}^k f_i$ we compute $\phi(c'_i)$ for all $i \in \{1, \ldots, k\}$ and set

$$\hat{\mathbf{c}} = \left(\left(\phi\left(c'_0\right) \right)_0, \phi(c'_1), \ldots, \phi(c'_k) \right).$$

However, we still need to use the multi-key LWE key-switching from Section 3.3 to switch the keys from the RLWE keys to $\mathbf{s}_1, \ldots, \mathbf{s}_k$. The full computation can be seen in Figure 5.

Note that when decrypting x_1 we get

$$\langle x_1, \mathbf{z} \rangle = \mathfrak{g} \cdot f_1 + \mathfrak{g}^{-1}(a) \cdot \mathbf{e}_{0,1} + \mathbf{e}_{1,2}.$$

Let $e'_1 = \mathfrak{g}^{-1}(a) \cdot \mathbf{e}_{0,1} + \mathbf{e}_{1,2}$. By repeatedly applying Lemma 5 while denoting e' in the formation of x_i by e'_i, the following holds for $i \in \{2, \ldots, k\}$:

$$\langle x_i, \mathbf{z} \rangle = \mathfrak{g} \prod_{j=1}^{i} f_j + \Sigma_{j=1}^{i-1} \left(\prod_{l=j+1}^{i} (f_l) \cdot e'_j \right) + e'_i. \tag{2}$$

Lemma 8. *Let $|\mathcal{P}| = k$ and let $\mathbf{b}_i$ and hpk_i be as defined above. Computing the key switching keys, $\mathsf{ksk}_1, \ldots, \mathsf{ksk}_k$, as in Π_{kskGen}, yields a noise given by*

$$\mathsf{err}_{\mathsf{ksk}} \le N^{k-1} \cdot (\gamma^2 + 1) \cdot \mu^2 + 64 \cdot (k-2) \cdot (3k-1) \cdot N^{k-1} \cdot l \cdot \gamma^2 \cdot \mu^2 + N \cdot l \cdot \gamma^2 \cdot \mu^2.$$

Hence, for $k > 0$, the error is bounded by $O\left(k^2 \cdot N^{k-1} \cdot l \cdot \gamma^2 \cdot \mu^2\right)$.

Proof. From lemma 5, we know the values of e'_i for all $i \in [k]$. This evaluation can then be inserted into the generalized formula 2, where it is trivial to see that the error is greatest at $i = k$. Therefore, to compute the final error of the Hybrid Product stage we only need

Key switching protocol, $\Pi_{\mathsf{KS}}(c)$

Input: An MKNGS ciphertext $c = \Sigma_i g_i/f_i + \Delta m$ where $\mathsf{ksk} = (\mathsf{ksk}_0, \ldots, \mathsf{ksk}_k)$ is obtained from Π_{HP}

Output: An LWE ciphertext $(b, A_1, \ldots, A_n)$ encrypting m.

1. for $i \in \{1, \ldots, k\}$ do:
   - $a_i \leftarrow \mathsf{KS}_{\mathsf{NGS}\to\mathsf{LWE}}(c, \mathsf{ksk}_i)$
2. $\mathbf{a} = (a_1, \ldots, a_k)$
3. $b \leftarrow \mathsf{KS}_{\mathsf{NGS}\to\mathsf{LWE}}(c, \mathsf{ksk}_0)$
4. **Out:** $c' := (b, \mathbf{a})$

Figure 6: Key switching from NGS to LWE

to compute the error at $i = k$:

$$\begin{aligned}
\mathsf{Var}\left(\langle x_i, \mathbf{z}\rangle - \mathfrak{g}\prod_{j=1}^{k} f_j\right) &= \mathsf{Var}\left(\sum_{j=1}^{k-1}\left(\prod_{l=j+1}^{k}(f_l)\cdot e'_j\right) + e'_k\right)\\
&\le \mathsf{Var}\left(\prod_{l=2}^{k} f_l \cdot e'_1\right) + \mathsf{Var}\left(\left(\left((k-2)\cdot\prod_{l=3}^{k} f_l\right)+1\right)\cdot e'_k\right)\\
&\le N^{k-1}\cdot(\gamma^2+1)\cdot\mu^2 + (k-2)\cdot(N^{k-2}\cdot 8)\cdot\left(16k\cdot N\cdot l\cdot\gamma^2\cdot\mu^2\right)\\
&\le N^{k-1}\cdot(\gamma^2+1)\cdot\mu^2 + 128\cdot(k-2)\cdot k\cdot N^{k-1}\cdot l\cdot\gamma^2\cdot\mu^2.
\end{aligned}$$

That the penultimate inequality holds follows from the fact that e'_i is largest at $i = k$ and the evaluation of the product of multiple f_l. From [CCS19], we know that the MKLWE Key Switching protocol described in Section 3.3 induces an additive error of at most $N \cdot l \cdot \gamma^2 \cdot \mu^2$, realizing the stated error bound. □

Observe that, up until now, the key-switching keys have had no dependency on the encrypted ciphertexts. Hence, we may assume that the key-switching keys have been pre-computed.

4.4.2 Key switching Protocol:

Having obtained the correct key-switching keys, the key switching protocol in Figure 6 can be executed. This uses the $\mathsf{KS}_{\mathsf{NGS}\to\mathsf{LWE}}$ sub-protocol defined in Section 3.3.2.

After the key switching steps we obtain an MKLWE ciphertext under the keys $\mathbf{s}_1, \ldots, \mathbf{s}_k$ with public parameters $\mathbf{a}_i = y \cdot \mathsf{ksk}_i$ for each party P_i, such that

$$b = y \cdot \left(\sum_{i=1}^{k} \mathsf{ksk}_i \cdot \mathbf{s}_i + e + \mathfrak{g}\cdot\prod_j f_j\right),$$

and $\mathsf{err}(b) = b - \sum_{i=1}^{k} \mathbf{a}_i \cdot \mathbf{s}_i - \Delta m$ by the definition of the error of an MKLWE ciphertext. Therefore we can conclude that the error introduced by key switching is as follows:

Lemma 9. *Let c be an* MKNGS.SEnc *ciphertext, such that $c = \sum_{i=1}^{k} \frac{g_i}{f_i} + \Delta m$, with error E and let $c' = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k)$ be the output of Π_{KS} on c. Then*

$$\mathsf{Var}\left(\mathsf{err}(c')\right) \le N^k \cdot \mu^2 \cdot l \cdot \left(\gamma^4 + \gamma^2 + 128\cdot(k-2)\cdot k\cdot l\cdot\gamma^4\right) + E + 8\cdot N^{k-1}.$$

Moreover, for $k > 0$, the error is bounded by $O(k^2 \cdot N^k \cdot l^2 \cdot \gamma^4 \cdot \mu^2 + E)$.

Proof. It is easy to observe that

$$\begin{aligned}\mathsf{err}(b) &= y \cdot e + \prod_{j=1}^{k} f_j \cdot \phi(c) \\ &= y \cdot e + \sum_{i=1}^{k} g_i \cdot \prod_{j \neq i} f_j + \prod_{j=1}^{k} f_j \cdot \phi(m).\end{aligned}$$

And by the decryption equation for an MKLWE ciphertext,

$$\begin{aligned}\mathsf{Var}\left(\mathsf{err}(b)\right) &= \mathsf{Var}\left(y \cdot e\right) + \mathsf{Var}\left(\sum_{i=1}^{k} g_i \cdot \prod_{j \neq i} f_j\right) + \mathsf{Var}\left(\prod_{j=1}^{k} f_j \cdot \phi(m)\right) \\ &\leq \mathsf{Var}(\mathfrak{g}^{-1}(\phi(c)) \cdot e) + \mathsf{Var}(c) + \mathsf{Var}\left(\prod_{j=1}^{k} f_j\right)\end{aligned}$$

where the final inequality holds as $m \in \mathbb{F}_2$, hence $\|m\|_2^2 \leq 1$. Moreover, $\mathsf{Var}(y) = \gamma^2$, $\mathsf{err}_{\mathsf{ksk}}$ is the error introduced by the key switching key, as identified in 4.4.1. Since $\Pi_{i=1}^{k} f_j$ is $k-1$ multiplications of a polynomial of at most degree N we obtain that

$$\begin{aligned}\mathsf{Var}(\mathsf{err}(b)) &\leq N \cdot l \cdot \gamma^2 \cdot \mathsf{err}_{\mathsf{ksk}} + E + \cdot N^{k-1}\mathsf{Var}(f_j) \\ &\leq N^k \cdot l \cdot \gamma^2 \cdot (\gamma^2+1) \cdot \mu^2 + 128 \cdot (k-2) \cdot k \cdot N^k \cdot l^2 \cdot \gamma^4 \cdot \mu^2 + E + 8 \cdot N^{k-1}\end{aligned}$$

which yields a complexity of $O(k^2 \cdot N^k \cdot l^2 \cdot \gamma^4 \cdot \mu^2 + E)$ for any $k > 0$. □

We conclude the bootstrapping by modswitching from the MKNGS modulus to the MKLWE modulus using the modswitching protocol introduced in Section 3.5.

4.5 Bootstrapping Noise

With the construction of bootstrapping completed we can now consider the error before and after bootstrapping.

Theorem 2. *Let $\mathcal{P} = \mathcal{P}_1, \ldots, \mathcal{P}_k$ such that k is the number of participating parties, each equipped with keys $(\mathbf{s}_i, \mathsf{bsk}_i, f_i)$. Moreover, let $c = \mathsf{MKLWE}_{\mathbf{s}_1, \ldots, \mathbf{s}_k}(m)$ such that $\mathsf{Var}(\mathsf{err}(c)) = E_{\mathsf{LWE}}$. Then $\Pi_{\mathrm{BS}}(c)$ produces $c' = \mathsf{MKLWE}(m)$ encrypted under the LWE keys $\mathbf{s}_1, \ldots, \mathbf{s}_n$ such that*

$$\mathsf{Var}\left(\mathsf{err}\left(\Pi_{\mathrm{BS}}(c)\right)\right) \leq \frac{q^2}{Q^2}\left(\mathsf{Var}(\mathsf{err}(\Pi_{\mathrm{KS}}(c)))\right) + (kn+1)\sqrt{2\pi}.$$

Moreover, the error while working in the MKNGS scheme does not exceed $(n \cdot k+1) \cdot N \cdot l \cdot \gamma^2$.

Proof. Let c be as defined above. Then by Theorem 1 we obtain that the error after step 6 is given by

$$\mathsf{Var}\left(\mathsf{err}(\mathsf{ACC})\right) = (n \cdot k + 1) \cdot N \cdot l \cdot \gamma^2.$$

As this is the final step in the MKNGS scheme we conclude that this is the total size of the error within this scheme.

Then, by Lemma 9 we obtain that the key switching algorithm introduces extra noise within the MKLWE encryption amounting to a total of

$$\begin{aligned}\mathsf{Var}\left(\mathsf{err}\left(\Pi_{\mathrm{KS}}(c)\right)\right) \leq & N^k \cdot l \cdot \gamma^2 \cdot (\gamma^2+1) \cdot \mu^2 + 128 \cdot (k-2) \cdot k \cdot N^k \cdot l^2 \cdot \gamma^4 \cdot \mu^2 \\ & + 2 \cdot (n \cdot k + 1) \cdot N \cdot l \cdot \gamma^2 + 8 \cdot N^{k-1}.\end{aligned}$$

Let $c' = \Pi_{\mathrm{KS}}(c)$. Using ModSwitch we obtain from Lemma 6

$$\mathsf{Var}\left(\mathsf{err}(\mathsf{ModSwitch}(c'))\right) \leq \frac{q^2}{Q^2}\left(\mathsf{Var}(\mathsf{err}(\Pi_{\mathrm{KS}}(c)))\right) + (kn+1)\sqrt{2\pi}.$$

A simple analysis yields the stated results. □

4.6 Heuristics

Set $\mathsf{Var}\left(\mathsf{err}(\Pi_{\mathrm{BS}}(c)\right) = E_{\mathsf{final}}$. It is clear from the previous section that given ct encrypting $f(m_1, \ldots, m_n)$ the bootstrapping algorithm outputs a ciphertext ct′ encrypting the same message.

The noise obtained after bootstrapping behaves as a Gaussian distribution, therefore we can apply the Central Limit Theorem and its accompanying heuristic to conclude that with overwhelming probability $\|\mathsf{err}(\mathsf{ct}')\| \leq 6 \cdot \sqrt{E_{\mathsf{final}}}$.

With the result from Theorem 2 we then conclude that with overwhelming probability

$$\|\mathsf{err}(\mathsf{ct}_{\mathsf{NGS}})\| \leq 6 \cdot \sqrt{(n \cdot k + 1) \cdot N \cdot l \cdot \gamma^2}. \tag{3}$$

Similarly for the MKNGS-scheme, we can prove that our NTRU-modulus Q is asymptotically less than the fatigue point found in [Dv21], $Q < \mathcal{O}(N^{2.484})$.

Theorem 3. *If the* NGS *ciphertexts of the protocol in Figure 4 satisfy equation 3 except with negligible probability and* $Q = \mathcal{O}(N^{\frac{3}{2}})$*, then these ciphertexts can be correctly decrypted except with negligible probability.*

Proof. As $n = \mathcal{O}(N)$, $l = \mathcal{O}(\log Q) = \mathcal{O}(\log N)$ and $\gamma^2, k \in \mathcal{O}(1)$, we have that the noise of NGS ciphertexts in the protocol in Figure 4 are $\mathcal{O}(N^{\frac{3}{2}})$ except with negligible probability, by Eq 3. For correct decryption, we require that this noise is less than $Q/4$. Thus it is sufficient to select $Q = \mathcal{O}(N^{\frac{3}{2}})$. □

5 Analysis on the maximum supported players with concrete parameters

In this section we analyze this noise for concrete parameter sets and determine the maximum number players each of these parameter sets can sustain.

Due to the construction of the scheme there are two natural possible noise overload points: Within the MKNGS scheme or within the MKLWE scheme after bootstrapping. Practically, we show in Theorem 2 that the MKLWE noise always dominates. Using the analysis and code provided by Ducas and Van Woerden, [Dv21], given the available parameters (Q, σ^2, the distribution of the MKNGS keys, and a desired security level $\lambda = 128$) an adaptation of the analysis in [BIP+22] provides a host of parameter sets which would be suitable, albeit not necessarily practical. Specifically, we were able to find the maximum values of $\log Q$ for each of our selected values for N with 128 bits of security. LWE parameters are set using the LWE Estimator, [APS15], to ensure that the (q, n)-pairs achieve 128 bits of security without being susceptible to the attacks in [ABD16, CJL16, Dv21]. Furthermore, as between key switching and modulus switching the server has access to a MKLWE ciphertext using parameters n and Q, these are also tested using the LWE estimator to ensure the same level of security.

Table 2 contains the tested possibilities for the selection of parameters along with the maximal number of players each set can support. Additionally, for each parameter set we computed the probability of a decryption error in each of the encryption schemes. In the table we include the greatest rate of error and denote which protocol is the cause for these

Table 2: Noise and possible number of players for various sets of parameters

N	Q	n	q	ε	k_{max}	error rate
2^{11}	2^{30}	660	2^{35}	2^{22}	2	$E_{\mathsf{MKLWE}} < 2^{-200}$
2^{12}	2^{45}	1210	2^{51}	2^{24}	4	$E_{\mathsf{MKLWE}} < 2^{-200}$
2^{13}	2^{67}	2060	2^{73}	2^{28}	7	$E_{\mathsf{MKLWE}} < 2^{-200}$
2^{14}	2^{100}	3310	2^{107}	2^{29}	11	$E_{\mathsf{MKLWE}} < 2^{-200}$
2^{15}	2^{149}	5160	2^{157}	2^{31}	17	$E_{\mathsf{MKLWE}} < 2^{-30}$

error rates - i.e. has the burden of limitation. For clarity, by ε we denote the LWE error bound on a single ciphertext.

We can see that for all parameter sets $q > Q$ and that the number of supported participating players is ≥ 2, which is sufficient for most practical situations as a server with no inputs is not counted as a player.

The size of our bootstrapping keys is $N \cdot n \cdot l \cdot \log_2(q)$ bits of which there are k. Similarly, the size of the key switching key is $N \cdot n \cdot l \cdot \log_2(q) \cdot (k+1)$ bits. By using our parameters, as in the first line of 2 and setting $k = 2$, we can compare with the smallest parameter sets of [KMS24] and [CCS19]. We note that the total size of the keys used in our work (212MB) is similar to that of [KMS24] (215MB) and about double that of [CCS19] (84MB), however we do provide 128 bit security where the predecessors only provide 110 bit security.

Where a significant gain is made compared to the state-of-the-art in MKFHE based on TFHE, [KMS22, CCS19], and even multi-party homomorphic encryption based on TFHE, [AKÖ23], is in the complexity. To see this let n be the dimension of the base scheme (MKLWE) and N be that of the bootstrapping scheme (MKNGS). Moreover, let $l = \log_B(Q)$ for a decomposition base B and bootstrapping scheme modulus Q.

Note that each gadget decomposition requires l polynomial multiplications, hence the claimed complexity of $\mathcal{O}(Nk + k^2)$ and $\mathcal{O}(Nk^2)$ gadget decompositions in [KMS24] and [CCS19] respectively needs to be extended by l for good comparison. Compared to our bootstrapping phase, which requires, respectively, $\mathcal{O}(nlk)$ and $\mathcal{O}(Nlk)$ to compute Π_{BS} and Π_{KS}. Since $N > n$ this results in a total complexity of $\mathcal{O}(Nlk)$. Note that this is the optimal complexity, as is achieved in the version of MKFHE as presented in [AKÖ23], however their protocol requires smaller parameters and their key switching is significantly less costly due to the exploitation of communication between parties.

A significant factor in this is the generation of the key-switching keys that are used in the bootstrapping protocol. The new construction allows the protocol to reuse these keys throughout a computation, which means the generation of the key-switching keys can be extracted from the protocol. Amortizing the $2k^2l + 4kl$ polynomial multiplications over the number of bootstraps reduces the complexity of key-switching key generation to $\mathcal{O}(1)$ in all practical cases.

6 Conclusion

In this work, we showed that it is possible to obtain Multi-Key Fully Homomorphic Encryption construction based on the FINAL scheme, [BIP+22], while keeping the parameters under the fatigue point, [Dv21]. In doing so, we provided a thorough analysis of the noise growth in both Single-Key and Multi-Key FINAL and showed that our scheme reduces the computational complexity to the optimal $O(Nkl)$ polynomial multiplications. We achieved this, among other things, by redefining how the Hybrid Product construction, [CCS19], is used, which allows the cost of key switching key generation to be amortized over the

number of bootstraps.

Due to the advantages of NTRU-based ciphertext, the size of intermediate value during homomorphic gate operation does not grow in the number of keys, our scheme gives up to 40% less smaller key size and computation time than LWE based MKFHE. Note that our parameter sets are merely proof of concept and a closer analysis may reveal more nuanced parameters which could result in significant increases in efficiency.

7 Acknowledgments

This work was supported by Cyber Security Research Flanders with reference number VR20192203 and by the FWO under an Odysseus project GOH9718N. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Cyber Security Research Flanders or the FWO.

References

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Berlin, Heidelberg, August 2016. doi:10.1007/978-3-662-53018-4_6.

[AH19] Asma Aloufi and Peizhao Hu. Collaborative homomorphic computation on data encrypted under multiple keys, 2019. arXiv:1911.04101.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Berlin, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_29.

[AKÖ23] Yavuz Akin, Jakub Klemsa, and Melek Önen. A practical TFHE-based multikey homomorphic encryption with linear complexity and low noise growth. In Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis, editors, *ESORICS 2023, Part I*, volume 14344 of *LNCS*, pages 3–23. Springer, Cham, September 2023. doi:10.1007/978-3-031-50594-2_1.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. URL: https://doi.org/10.1515/jmc-2015-0016, doi:doi:10.1515/jmc-2015-0016.

[BIP+22] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. FINAL: Faster FHE instantiated with NTRU and LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 188–215. Springer, Cham, December 2022. doi:10.1007/978-3-031-22966-4_7.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Multi-key homomorphic encryption from TFHE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 446–472. Springer, Cham, December 2019. doi:10.1007/978-3-030-34621-8_16.

[CDKS19] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 395–412. ACM Press, November 2019. doi:10.1145/3319535.3363207.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020. doi:10.1007/s00145-019-09319-x.

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016. doi:10.1112/S1461157016000371.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Berlin, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_24.

[Dv21] Léo Ducas and Wessel P. J. van Woerden. NTRU fatigue: How stretched is overstretched? In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 3–32. Springer, Cham, December 2021. doi:10.1007/978-3-030-92068-5_1.

[GPV23] Antonio Guimarães, Hilder V. L. Pereira, and Barry Van Leeuwen. Amortized bootstrapping revisited: Simpler, asymptotically-faster, implemented. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VI*, volume 14443 of *LNCS*, pages 3–35. Springer, Singapore, December 2023. doi:10.1007/978-981-99-8736-8_1.

[KKL+23] Taechan Kim, Hyesun Kwak, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 726–740, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3576915.3623176.

[KL23] Jiseung Kim and Changmin Lee. A polynomial time algorithm for breaking NTRU encryption with multiple keys. *DCC*, 91(8):2779–2789, 2023. doi:10.1007/s10623-023-01233-5.

[KLSW21] Hyesun Kwak, Dongwon Lee, Yongsoo Song, and Sameer Wagh. A unified framework of homomorphic encryption for multiple parties with non-interactive setup. Cryptology ePrint Archive, Paper 2021/1412, 2021. https://eprint.iacr.org/2021/1412. URL: https://eprint.iacr.org/2021/1412.

[KMS22] Hyesun Kwak, Seonhong Min, and Yongsoo Song. Towards practical multi-key TFHE: Parallelizable, key-compatible, quasi-linear complexity. Cryptology ePrint Archive, Report 2022/1460, 2022. URL: https://eprint.iacr.org/2022/1460.

[KMS24] Hyesun Kwak, Seonhong Min, and Yongsoo Song. Towards practical multi-key TFHE: Parallelizable, key-compatible, quasi-linear complexity. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14604 of *LNCS*, pages 354–385. Springer, Cham, April 2024. doi:10.1007/978-3-031-57728-4_12.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 1219–1234, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2213977.2214086.

[LP19] Hyang-Sook Lee and Jeongeun Park. On the security of multikey homomorphic encryption. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 236–251. Springer, Cham, December 2019. doi:10.1007/978-3-030-35199-1_12.

[MS18] Daniele Micciancio and Jessica Sorrell. Ring packing and amortized FHEW bootstrapping. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 100:1–100:14. Schloss Dagstuhl, July 2018. doi:10.4230/LIPIcs.ICALP.2018.100.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_26.

[Par21] Jeongeun Park. Homomorphic encryption for multiple users with less communications. *IEEE Access*, 9:135915–135926, 2021. doi:10.1109/ACCESS.2021.3117029.

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 217–238. Springer, Berlin, Heidelberg, October / November 2016. doi:10.1007/978-3-662-53644-5_9.

[XTW+23] Kexin Xu, Benjamin Hong Meng Tan, Li-Ping Wang, Khin Mi Mi Aung, and Huaxiong Wang. Multi-key fully homomorphic encryption from ntru and (r)lwe with faster bootstrapping. *Theoretical Computer Science*, 968:114026, 2023. URL: https://www.sciencedirect.com/science/article/pii/S0304397523003390, doi:10.1016/j.tcs.2023.114026.