Faster Secret Keys for (T)FHE

Loris Bergerat[*1], Ilaria Chillotti[†1], Damien Ligier[‡1], Jean-Baptiste Orfila[§1], Adeline Roux-Langlois[¶2], and Samuel Tap[‖1]

[1]Zama, Paris, France - https://zama.ai/
[2]Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC
14000 Caen, France

Abstract

GLWE secret keys come with some associated public information, like their size or the distribution probability of their coefficients. Those information have an impact on the FHE algorithms, their computational cost, their noise growth, and the overall security level.

In this paper, we identify two limitations with (T)FHE: there is no fine-grained control over the size of a GLWE secret key, and there is a minimal noise variance which leads to an unnecessary increment of the level of security with large GLWE secret keys.

We introduce two (non exclusive) new types of secret keys for GLWE-based cryptosystems, that are designed to overcome the aforementioned limitations. We explain why these are as secure as the traditional ones, and detail all the improvements that they brought to the FHE algorithms. We provide many comparisons with state-of-the-art TFHE techniques, and benchmarks showing computational speed-ups between 1.3 and 2.4 while keeping the same level of security and failure probability. Furthermore, the size of the public material (i.e., key switching and bootstrapping keys) is also reduced by factors from 1.5 and 2.7.

[*] loris.bergerat@zama.ai
[†] ilaria.chillotti@zama.ai
[‡] damien.ligier@zama.ai
[§] jb.orfila@zama.ai
[¶] adeline.roux-langlois@cnrs.fr
[‖] samuel.tap@zama.ai

Contents

1 Introduction

Fully homomorphic encryption (FHE) is a technology allowing to perform computations over encrypted data. After the first solution proposed in 2009 by Gentry [Gen09], the field has received a huge interest and has experienced a drastic improvement in the following decade. Gentry introduced a vital technique called bootstrapping, able to reduce the noise inside ciphertexts. Indeed, FHE ciphertexts contain noise for security reasons, and every time an homomorphic operation is performed, the noise level grows. If not controlled, too much noise will eventually compromise the message. Nowadays, practical FHE schemes are all inspired by Gentry's solution and are based on the Learning With Errors (LWE) problem [Reg05], its Ring variants (RLWE) [SSTX09, LPR10] and the General approach (GLWE) [LS15, BGV12]. One of these schemes is called TFHE [CGGI20], and will be the focus of this paper. TFHE differs from other FHE schemes because it proposed a very efficient bootstrapping technique that reduces the noise and, at the same time, is able to homomorphically evaluate a function, represented as a look-up table (LUT).

TFHE, as many other (R)LWE based schemes, works with a cyclotomic ring $\mathfrak{R}_{q,N} = \mathbb{Z}_q\left[X\right] / \left\langle X^N + 1 \right\rangle$, with N a power of 2. The parameter N has a huge impact on the precision of the messages that can be bootstrapped. Roughly speaking, in order to bootstrap an additional bit of message, we need to double the size of N, which implies doubling the size of the secret key and more than double the cost of the operation. Apart from being used to encrypt and decrypt messages, secret keys are used in bootstrapping and key switching (an operation used to homomorphically switch from a secret key to another) procedures.

In this paper we propose new ways to generate secret keys allowing to reduce the size of the public keys, and the cost of some homomorphic algorithms.

Impact of parameter choices. As mentioned above, the choice of parameters has a huge impact on performance, and it has to guarantee the desired security as well as the desired correctness for a given message precision. Several parameters are brought into play to estimate the level of security: the ciphertext modulo q, the sizes of the secret keys (either n in LWE keys or kN in GLWE keys), the distributions of the secret keys and the variances of the noise in fresh encryptions.

In TFHE implementations, q is often chosen equal to 2^{32} or 2^{64}, in order to be able to work with 32 or 64 bit integers, respectively, since they are native types in the majority of machines used nowadays. The polynomial size N often has to be a power of 2. In order to keep the same security level when increasing n, we can reduce the variance σ^2 of the noise, but the value of q imposes a lower bound on this value, meaning that starting from a certain point, we cannot reduce the variance anymore, otherwise we lose security. Notice that a small increase of the value of n allows for a small decrease of the value of σ^2. But when working with polynomials, moving to a bigger power of 2 for N will lead to a large increase of the size of the secret key, from kN to $k \cdot 2N$, and so a large decrease of σ^2 when allowed. For the same security reason as above, at some point we reach a limit where we cannot reduce the variance σ^2 of the noise anymore. The consequence is that to avoid having no security at all, we end up with a security level way higher than desired. This is of course not a problem in terms of security, but maybe there could be a third option that could lead to keep the desired level of security and improve the efficiency of the homomorphic computation.

We can now sum up a couple of limitations for TFHE:

1. there is *no fine-grained control over the size of a GLWE secret key*, it is of the form kN with N a power of two;

2. there is *a minimal value for the variance σ^2 of the noise* for a given security level and a given ciphertext modulus q. So when one increases either n (or kN), they eventually reach a plateau in terms of noise variance. We write n_{plateau} the first value of this plateau, and we evaluated it's value to be 2443 for 128 bits of security and $q = 2^{64}$.

Our contribution. In this paper, we investigate how we could get rid of those limitations by modifying the way secret keys are generated.

A key switch is an FHE operator that can be costly. It allows to convert a ciphertext encrypted under a secret key $\boldsymbol{s}_1$ to another secret key $\boldsymbol{s}_2$, and consists in evaluating a linear combination between public ciphertext values and the key switching key, which is composed of individual encryptions of the elements of $\boldsymbol{s}_1$ under the secret key $\boldsymbol{s}_2$. Roughly speaking, reducing the size of the first secret key reduces the amount of operations in this linear combination, and so improves the key switching in terms of computational cost, noise growth and size of the key switching key. Furthermore, if $\boldsymbol{s}_1$ publicly shares some elements with $\boldsymbol{s}_2$, this would improve the key switching operation as well since some operations in the linear combinations would become unnecessary. This phenomenon resemble to what happens during the relinearization technique (e.g. [Bra12, FV12]), where only a part of the mask of the ciphertext is relinearized.

Following all these observations, we propose two new kinds of secret keys: *partial secret keys* and *shared randomness secret keys*. The first kind consists in allowing a GLWE secret key traditionally containing kN random elements, to contain less random elements, the rest being zeros. The second kind, the shared randomness secret keys, consists in reusing the randomness from a bigger key inside a smaller key, instead of generating it independently as done traditionally.

We provide details justifying the security of our new types of secret keys. In order to exploit them, we adapted TFHE's algorithms, which lead to variants offering smaller noise growth, smaller amount of computation and smaller needed public material. We also share benchmarks comparing the running time needed with and without the new secret keys. Our simulations, done with the optimization tool proposed in [BBB$^+$22], show that the use of these new secret keys improve the state of the art for all precisions. These results are confirmed by practical experiments, showing a speed up between 1.3 and 2.4 times, while keeping the same level of security and failure probability. Furthermore, the use of these new key types allows to reduce the size of the public material (i.e., key switching and bootstrapping keys) by a factor between 1.5 and 2.7.

Related Works. To the best of our knowledge, there are no mentions of such GLWE secret keys or something similar in the prior art. The closest work we can mention is by Lee and Yoon [LY23] where the server can publicly transform a bootstrapping key encrypted under a traditional secret key to an extended version encrypted under a secret key containing zeros between each secret coefficients. This extended bootstrapping key allows the authors to bootstrap messages with bigger precision, but it does not improve the noise growth. All this contribution only involves traditional GLWE secret keys for encryption.

Concurrent Works. Lee *et al.* [LMSS23] introduced new types of secret keys. They call it *block binary keys*, which are different from our contribution but concurrently exploit the advantage of having nested secret keys, re-using their randomness, as we also introduce in this paper as *shared randomness secret keys*.

Paper Organization. In Section 2 we recall mathematics and FHE notations and security notions. In Section 3 we introduce the concept of *partial GLWE secret keys* and provide intuitions of security as well as all the FHE algorithms that benefit from them either in terms of noise growth or in terms of computation. Section 4 is composed in the same manner, but it introduces the concept of *shared randomness secret keys* this time. In Section 5 we detail what can be exploited when combining both partial and shared randomness secret keys, and we give a few applications for them. In Section 6, we illustrate how useful those new types of secret keys are by comparing the techniques we introduced in this paper and the state-of-the-art. Finally, in Section 7, we conclude and speak about potential future work.

2 Preliminaries

Notations. Let q be a positive integer and let N be a power of 2. In this paper we mostly work with $\mathbb{Z}_q$, which refers to the ring $\mathbb{Z}/q\mathbb{Z}$, and with the ring $\mathfrak{R}_{q,N} = \mathbb{Z}_q[X] / \langle X^N + 1 \rangle$. For error distributions, we note χ for a generic distribution and $\mathcal{N}_{\sigma^2}$ for a *Gaussian distribution* with a mean set to zero and a standard deviation set to σ. We note by $\mathcal{U}(S)$ a uniform distribution in the generic set S and by $\mathcal{D}(S)$ a generic probability distribution in the generic set S. We use the symbol $||$ for vector concatenation. We note with upper cases (e.g. M, A, S, B, E) polynomials and with lower cases (e.g., m, a, s, b, e) scalars. We use bold characters to note vectors of polynomials (e.g., $\boldsymbol{A}, \boldsymbol{S}$) and vectors of coefficients (e.g. $\boldsymbol{a}, \boldsymbol{s}$).

Definition 1 (General Learning With Errors (GLWE) Decision Problem) *Let $\boldsymbol{S} = (S_0, \cdots, S_{k-1}) \in \mathfrak{R}_{q,N}^k$ be a secret, where $S_i = \sum_{j=0}^{N-1} s_{i,j} X^j$ is sampled from a given distribution $\mathcal{D}(\mathfrak{R}_{q,N})$ for all $0 \leq i < k$, and let χ be an error distribution. We define $(\boldsymbol{A}, B = \sum_{i=0}^{k} A_i \cdot S_i + E) \in \mathfrak{R}_{q,N}^{k+1}$ to be a sample from the general learning with errors ($\mathrm{GLWE}_{N,k,\chi}$) distribution, such that $\boldsymbol{A} = (A_0, \ldots, A_{k-1}) \hookleftarrow \mathcal{U}(\mathfrak{R}_{q,N})^k$, meaning that all the coefficients of A_i are sampled uniformly from $\mathbb{Z}_q$, and the error (noise) polynomial $E \in \mathfrak{R}_{q,N}$ is such that all the coefficients are sampled from χ.*

The decisional $\mathrm{GLWE}_{N,k,\chi}$ problem [LS15, BGV12] consists in distinguishing m independent samples from $\mathcal{U}(\mathfrak{R}_{q,N})^{k+1}$ from the same amount of samples from $\mathrm{GLWE}_{N,k,\chi}$, where $\boldsymbol{S} \in \mathfrak{R}_{q,N}^k$ follows a given distribution $\mathcal{D}$.

In general, the secret key distribution $\mathcal{D}(\mathfrak{R}_{q,N})$ is such that the polynomial coefficients are usually either sampled from a uniform binary distribution, a uniform ternary distribution or a Gaussian distribution ([BJRLW23, ACPS09]).

Starting from the GLWE problem, we can define GLWE ciphertexts.

Definition 2 (GLWE Ciphertexts) *A GLWE ciphertext of a plaintext $M \in \mathfrak{R}_{q,N}$ under the secret key $\boldsymbol{S} \in \mathfrak{R}_{q,N}^k$ is defined as follows:*

$$\mathsf{CT} = \left(\boldsymbol{A}, B = \sum_{i=0}^{k-1} A_i \cdot S_i + M + E \right) \in \mathrm{GLWE}_{\boldsymbol{S}}(M) \subseteq \mathfrak{R}_{q,N}^{k+1} \tag{1}$$

such that $\boldsymbol{A} = (A_0, \ldots, A_{k-1}) \hookleftarrow \mathcal{U}(\mathfrak{R}_{q,N})^k$ and $E \in \mathfrak{R}_{q,N}$ is such that all its coefficients are sampled from a gaussian distribution $\mathcal{N}_{\sigma^2}$.

Remark 1 (LWE and RLWE) *We recall that, when $N = 1$, the GLWE problem (resp. ciphertext) becomes the LWE problem (resp. ciphertext): $\mathrm{GLWE}_{1,k,\chi} = \mathrm{LWE}_{n=k,\chi}$. In this case we consider the parameter $n = k$ to be the size of the LWE secret key and we note the ciphertext, the message and the secret with a lower case: $\mathsf{ct} \in \mathrm{LWE}_{\boldsymbol{s}}(m)$. When $k = 1$, the GLWE problem (resp. ciphertext) becomes the RLWE problem (resp. ciphertext): $\mathrm{GLWE}_{N,1,\chi} = \mathrm{RLWE}_{N,\chi}$. In this case, since we are still working with polynomials, we keep upper cases for ciphertexts, messages and secret key: $\mathsf{CT} \in \mathrm{RLWE}_{\boldsymbol{S}}(M)$.*

Definition 3 (Flatten Representation of a GLWE Secret Key) *A GLWE secret key* $\boldsymbol{S} = \left(S_0 = \sum_{j=0}^{N-1} s_{0,j}X^j, \cdots, S_{k-1} = \sum_{j=0}^{N-1} s_{k-1,j}X^j\right) \in \mathfrak{R}_{q,N}^k$ *can be* flattened *into an LWE secret key* $\bar{\boldsymbol{s}} = (\bar{s}_0, \cdots, \bar{s}_{kN-1}) \in \mathbb{Z}^{kN}$ *in the following manner:* $\bar{s}_{iN+j} := s_{i,j}$*, for* $0 \leq i < k$ *and* $0 \leq j < N$*.*

In TFHE, apart from LWE, RLWE and GLWE ciphertexts, there are two additional constructions that are used, that are called GLev [CLOT21] and GGSW [GSW13]. A GLev ciphertext is a collection of GLWE ciphertexts, while a GGSW is a collection of GLev ciphertexts. In the same way as GLWE can be specialized as LWE and as RLWE ciphertext, GLev and GGSW can be specialized as Lev and GSW respectively, and as RLev and RGSW. We give the general definition below.

Definition 4 (GLev Ciphertexts [CLOT21]) *Given a decomposition base* $\beta \in \mathbb{N}^*$ *and a decomposition level* $\ell \in \mathbb{N}^*$*, a GLev ciphertext of a plaintext* $M \in \mathfrak{R}_{q,N}$ *under a GLWE secret key* $\boldsymbol{S} \in \mathfrak{R}_{q,N}^k$ *is defined as follows:*

$$\overline{\mathsf{CT}} = (\mathsf{CT}_0, \ldots, \mathsf{CT}_{\ell-1}) \in \mathsf{GLEV}_{\boldsymbol{S}}^{(\beta,\ell)}(M) \subseteq \mathfrak{R}_{q,N}^{\ell\times(k+1)} \tag{2}$$

such that $\mathsf{CT}_j \in \mathrm{GLWE}_{\boldsymbol{S}}\left(\frac{q}{\beta^{j+1}}M\right) \subseteq \mathfrak{R}_{q,N}^{k+1}$ *for* $0 \leq j < \ell$*.*

Definition 5 (GGSW Ciphertexts [GSW13, CLOT21]) *Given a decomposition base* $\beta \in \mathbb{N}^*$ *and a decomposition level* $\ell \in \mathbb{N}^*$*, a GGSW ciphertext of a plaintext* $M \in \mathfrak{R}_{q,N}$ *under a GLWE secret key* $\boldsymbol{S} \in \mathfrak{R}_{q,N}^k$ *is defined as follows:*

$$\overline{\overline{\mathsf{CT}}} = \left(\overline{\mathsf{CT}}_0, \ldots, \overline{\mathsf{CT}}_k\right) \in \mathsf{GGSW}_{\boldsymbol{S}}^{(\beta,\ell)}(M) \subseteq \mathfrak{R}_{q,N}^{(k+1)\times\ell\times(k+1)} \tag{3}$$

such that $\overline{\mathsf{CT}}_i \in \mathsf{GLEV}_{\boldsymbol{S}}^{(\beta,\ell)}(-S_i \cdot M) \subseteq \mathfrak{R}_{q,N}^{\ell\times(k+1)}$ *for* $0 \leq i \leq k$*.*

Programmable Bootstrapping (PBS). We call Programmable Bootstrapping [CGGI20, CJL$^+$20, CJP21], or PBS, any FHE operator that enables to reset the noise in a ciphertext to a fixed level (when certain conditions are fulfilled) and to evaluate, at the same time, a lookup-table homomorphically on the encrypted message. Such an operator takes as input an LWE ciphertext encrypting a message m, a bootstrapping key BSK (i.e., a list of GGSW ciphertexts encrypting the elements of the secret key used to encrypt the message m), an encryption of a lookup table L, and outputs an LWE ciphertext with a fixed level of noise encrypting the message $L[m]$ when the process is successful. There is a failure probability that can be estimated and that we note p_{fail}.

Remark 2 (FFT Error) *Inside TFHE PBS, polynomial multiplications are performed with an FFT. While very efficient, it introduces a noise due to the casting of the bootstrapping key (64-bit integers) into floating points (double with 53-bits mantissa) and the accumulation of the error along the computation in the Fourier* *domain. We use the corrective formula from [BBB⁺22] to model this noise. It consists into adding to the noise of an external product the following formula*

$$\mathsf{FftError}_{k,N,\beta,\ell} = 2^{\omega_1} \cdot \ell \cdot \beta^2 \cdot N^2 \cdot (k+1)$$

with $\omega_1 \approx 22 - 2.6$*, the* GLWE *dimension* k*, the polynomial size* N *and the decomposition parameters* (β, ℓ)*.*

Secret Keys With Fixed Hamming Weight (FHW). A fixed-Hamming-weight (FHW) binary (resp. ternary) GLWE secret key of hamming weight $h \in \mathbb{N}$ is a GLWE secret key such that its polynomial coefficients are in $\{0,1\}$ (resp. $\{-1,0,1\}$) and contains exactly h non-zero coefficients. We note these two distributions $\mathcal{FHW}(h, \{0,1\})$ and $\mathcal{FHW}(h, \{-1,0,1\})$ respectively. Such keys come with public knowledge: the dimension k, the polynomial ring $\mathfrak{R}_{q,N}$ (including the polynomial degree N), the distribution (binary or ternary), the hamming weight h. This type of secret keys has already been used in FHE schemes, such as CKKS [CKKS17], because it offers a lower value for the worst case noise growth.

Table 1 summarizes public knowledge for different secret key types used in FHE.

Key Type	Size	Ring	Distribution	Hamming Weight
Uniform Binary	k	$\mathfrak{R}_{q,N}$	$\mathcal{U}(\{0,1\})$	unknown
Uniform Ternary	k	$\mathfrak{R}_{q,N}$	$\mathcal{U}(\{-1,0,1\})$	unknown
Gaussian	k	$\mathfrak{R}_{q,N}$	$\mathcal{N}_{\mu}\sigma^2$	unknown
Small Uniform	k	$\mathfrak{R}_{q,N}$	$\mathcal{U}(\mathbb{Z}_\alpha)$	unknown
Uniform	k	$\mathfrak{R}_{q,N}$	$\mathcal{U}(\mathbb{Z}_q)$	unknown
FHW Binary	k	$\mathfrak{R}_{q,N}$	$\mathcal{FHW}(h, \{0,1\})$	h
FHW Ternary	k	$\mathfrak{R}_{q,N}$	$\mathcal{FHW}(h, \{-1,0,1\})$	h

Table 1: Comparison between secret key types in terms of public knowledge.

2.1 Setting the parameters

Attacks on LWE. We quickly recall the known attacks against LWE which are important to choose parameters. These attacks and the associated references are the one used in the lattice estimator [APS15]. This tool is used[1] to find out the smallest noise variance σ^2 guarantying the desired level of security λ.

The first well known kind of attacks is the so called LWE primal attacks. This attack was first formulated in [ADPS16] and studied and verified in [AGVW17, DSDGR20, PV21]. It consists of using lattice reduction to solve an instance of uSVP (unique Shortest Vector Problem) generated from LWE samples. The most common way to perform this reduction, is to use the BKZ algorithm [SE94] to reduce a lattice basis by using an SVP (Shortest Vector Problem) oracle. So, based on this attack, the security of an LWE instance is based on the cost of lattice reduction for

[1] https://github.com/zama-ai/concrete/tree/main/tools/parameter-curves

solving uSVP. In the paper [ADPS16], the authors propose to analyze the hardness of RLWE as an LWE problem. All the research on this attack tend to find the best cost of solving uSVP in order to find the closest model of security for LWE and by extension for RLWE.

The second type of attack is the LWE dual attacks. This attack is explained in [MR09] and upgraded with the dual hybrid attacks in [Alb17]. It consist of solving SIS (Short Integer Solution problem) in the dual lattice of the lattice formed by LWE samples. As for the first type of attacks, the security of an LWE instance is based on the cost of solving the problem SIS.

The third well known kind of attacks is the coded-BKW attacks, which are based on the algorithm BKW (Blum, Kalai and Wasserman [BKW03]). This attack is explained in [GJS15, KF15]. BKW algorithm is a recursive dimension reduction for LWE instances. In [GJS15], authors use these attacks on RLWE. To do that, the RLWE problem is seen as a sub problem of LWE.

Attacks on RLWE/GLWE. In the last decade, some attacks (as exemple [CDW17, PMHS19, BRL20, BLNRL23]) tried to take advantage of the structure of the RLWE and GLWE to solve the id-SVP (ideal-Shortest Vector Problem). Nonetheless, none of this attacks is as efficient as the LWE attacks presented before. It means that when one wants to efficiently break GLWE, they actually use LWE attacks so the security of GLWE $\in \mathfrak{R}_{N,q}^{k+1}$ is estimated from the security of LWE $\in \mathbb{Z}_q^{kN+1}$.

Other Attacks. Some other attacks are not based on classical problem reduction but on the leakage of some fraction of the coordinates of the NTT transform of the RLWE secret. It's the case of the article [DSGKS18] which propose a more direct attacks against RLWE.

3 Partial GLWE Secret Key

A GLWE secret key usually contains kN random elements. Our first observation is that we can allow this secret key to only contain a number ϕ of random elements and the rest of them will be set to zero. We first formally define this notion of partial secret key, then explain why this should not have an impact on the hardness of the underlying GLWE problem. After discussing the hardness of such keys, we list the different advantages and improvements which they offer.

A partial GLWE secret key is composed of two parts, the first one contains secret random elements (sampled from a distribution $\mathfrak{D}$) and the second part is filled with *known zeros*. As a simple example we could define the following partial GLWE secret key:

$$\boldsymbol{S} = (S_0, S_1) \in \mathfrak{R}_{q,N}^2 \text{ with } S_0 = \sum_{j=0}^{N-1} s_{0,j} X^j \text{ and } S_1 = \sum_{j=0}^{N/2-1} s_{1,j} X^j$$

where $s_{0,0}, \cdots, s_{0,N-1}$ and $s_{1,0}, \cdots, s_{1,\frac{N}{2}-1}$ are sampled from $\mathfrak{D}$, and the other coefficients are publicly known to be set to zero.

Definition 6 (GLWE Partial Secret Key) *A Partial GLWE secret key is a vector $\boldsymbol{S}^{[\phi]} \in \mathfrak{R}_{q,N}^k$ associated with its filling amount ϕ such that $0 \leq \phi \leq kN$. Indeed, this key will have ϕ random coefficients sampled from a distribution $\mathfrak{D}$ and $kN - \phi$ known zeros. Both the locations of the random elements and the known zeros are public. By convention, we fill the coefficients starting at coefficient $s_{0,0}$, then $s_{0,1}$ and so on, and when the first polynomial is entirely filled, we fill the second polynomial starting at $s_{1,0}$ and so on, until we fill ϕ coefficients, up to $s_{k-1,N-1}$.*

When we write $\mathsf{Var}\left(\boldsymbol{S}^{[\phi]}\right)$ (resp. $\mathbb{E}\left(\boldsymbol{S}^{[\phi]}\right)$), we refer to the variance (resp. the expectation) of $\mathfrak{D}$ (either uniform binary distribution, uniform ternary distribution, Gaussian distribution, small uniform distribution). When $\mathfrak{D}$ is a uniform binary distribution, $\mathsf{Var}\left(\boldsymbol{S}^{[\phi]}\right) = 1/4$ and $\mathbb{E}\left(\boldsymbol{S}^{[\phi]}\right) = 1/2$.

We now define the flatten representation of a partial GLWE secret key.

Definition 7 (Flatten Representation of a Partial GLWE Secret Key) *A partial GLWE secret key $\boldsymbol{S}^{[\phi]} = \left(S_0 = \sum_{j=0}^{N-1} s_{0,j} X^j, \cdots, S_{k-1} = \sum_{j=0}^{N-1} s_{k-1,j} X^j\right) \in \mathfrak{R}_{q,N}^k$ (Definition 6) can be viewed as a flatten LWE secret key $\bar{\boldsymbol{s}} = (\bar{s}_0, \cdots, \bar{s}_{\phi-1}) \in \mathbb{Z}^\phi$ in the following manner: $\bar{s}_{iN+j} := s_{i,j}$, for $0 \leq j < N$ and $0 \leq i < k$ with $iN + j < \phi$. This flatten representation contains only the ϕ unknown coefficients.*

Before checking the security in detail, this type of keys seem to be a secure solution, taking into account the plateau limitation (Limitation 2). They also seem to offer a nice in-between solution.

3.1 Hardness of Partial GLWE

The GLWE partial secret key problem $\boldsymbol{S}^{[\phi]} \in \mathfrak{R}_{q,N}^{k}$, as defined in Definition 6, seems to be at least as hard as a GLWE problem in a ring of dimension ϕ. First, we present the GLWE alternate partial secret key, a key where the secret elements are separated by $2^\nu - 1$ known zeros. We prove the security of a such secret key distribution by proving that the $GLWE$ problem in $\mathfrak{R}_{q,N}^{k+1}$ is equivalent to the $GLWE$ problem in $\mathfrak{R}_{q,2^\nu N}^{k+1}$ instantiated with alternate partial GLWE secret keys.

Next, we will use this result to generalized the idea to the partial GLWE secret keys.

Definition 8 (Alternate Partial GLWE Secret Key) *An alternate partial* GLWE *(*P-GLWE$_{2^\nu N,k,\chi}$*) secret, is a* GLWE *secret where the key alternates between one unknown element and $2^\nu - 1$ known elements. This key is composed of N random coefficients sampled from a distribution $\mathfrak{D}$ and $(2^\nu - 1)N$ known zero coefficients. As for the partial* GLWE *secret key (Definition 6), both the locations of the random elements and the known zeros are public. For example, we have $S = \sum_{k=0}^{N-1} s_k \cdot X^{k \cdot 2^\nu}$, with $s_i \hookleftarrow \mathcal{U}(\{0,1\})$, is the binary version of partial secret key in $\mathfrak{R}_{q,2^\nu N}$.*

We show in Theorem 1 that the alternate partial GLWE problem defined in Definition 8 on the ring $\mathfrak{R}_{q,2^\nu N}$ is at least as hard as the GLWE problem on the ring $\mathfrak{R}_{q,N}$.

Theorem 1 (Hardness of P-GLWE**)** *For any $\nu \in \mathbb{Z}$, the* P-GLWE$_{2^\nu N,k,\chi}$ *sample in $\mathfrak{R}_{q,2^\nu N}^{k+1}$ is as least as hard than 2^ν $GLWE_{N,k,\chi}$ samples in $\mathfrak{R}_{q,N}^{k+1}$.*

Proof 1 (Theorem 1) *The idea of this proof is to pack 2^ν* GLWE$_{N,k,\chi}$ *samples in one* P-GLWE$_{2^\nu N,k,\chi}$ *sample. To do so, we differentiate the 2^ν samples from* GLWE$_{N,k,\chi}$ *in $\mathfrak{R}_{N,q}^{k+1}$, by noting them* GLWE$_{\boldsymbol{S}(X)}^{w}$ *with $w \in [\![0, 2^\nu[\![$. Observe that all of them are encrypted under the same secret key $\boldsymbol{S} = (S_0, \ldots, S_{k-1}) \in \mathfrak{R}_{q,N}^{k}$, with $S_i = \sum_{j=0}^{N-1} s_{i,j} X^j$.*

Each one of the k polynomials composing the GLWE$_{\boldsymbol{S}(X)}^{w}$ *sample is noted with an exponent w: $A_i^w = \sum_{j=0}^{N-1} a_{i,j}^w X^j$, with $i \in [\![0, k[\![$. Starting from these 2^ν samples, we define a new sample from* P-GLWE$_{2^\nu N,k,\chi}$*. First, for each sample* GLWE$_{\boldsymbol{S}(X)}^{w} \in \mathfrak{R}_{q,N}^{k+1}$*, we need to evaluate each polynomial in X^ν:*

$$\mathfrak{R}_{q,N} \longrightarrow \mathfrak{R}_{q,2^\nu N},$$
$$A_i^w(X) = \sum_{j=0}^{N-1} a_{i,j}^w \cdot X^j \longmapsto A_i^w(X^{2^\nu}) = \sum_{j=0}^{N-1} a_{i,j}^w \cdot X^{j \cdot 2^\nu}.$$

So, for each sample GLWE$_{\boldsymbol{S}(X)}^{w}$ *in $\mathfrak{R}_{q,N}^{k+1}$, we obtain:*

$$\mathrm{GLWE}_{\boldsymbol{S}(X)}^{w} \longrightarrow \widetilde{\mathrm{GLWE}}_{\boldsymbol{S}(X^{2^\nu})}^{w},$$
$$\left(A_0^w(X), \ldots, A_{k-1}^w(X), B^w(X)\right) \longmapsto \left(A_0^w(X^{2^\nu}), \ldots, A_{k-1}^w(X^{2^\nu}), B^w(X^{2^\nu})\right).$$

We notice that for each polynomial, each coefficient is separated from the other by $2^\nu - 1$ *zeros. Following the previous definition of* P-GLWE *(Definition 8), the secret key is in the desired shape. But the* $A_i^w(X^{2^\nu})$ *polynomials are not uniform anymore, only the coefficients of degree multiple of* 2^ν *are. So we can't already define* $\widetilde{\mathrm{GLWE}}^{w}_{\boldsymbol{S}(X^{2^\nu})}$ *as a sample of* $\text{P-GLWE}_{2^\nu N,k,\chi}$*. Now for each* $\widetilde{\mathrm{GLWE}}^{w}_{\boldsymbol{S}(X^{2^\nu})}$ *we rotate all the* A_i^w *and the* B^w *polynomials by* X^w*:*

$$\left(A_0^w(X^{2^\nu}) \cdot X^w, \ldots, A_{k-1}^w(X^{2^\nu}) \cdot X^w, B^w(X^{2^\nu}) \cdot X^w\right).$$

We now sum all of them together to obtain the expected sample from $\text{P-GLWE}_{2^\nu N,k,\chi} \in \mathfrak{R}_{q,2^\nu N}^{k+1}$*:*

$$\sum_{w=0}^{2^\nu-1} \left(A_0^w(X^{2^\nu})X^w, \ldots, A_{k-1}^w(X^{2^\nu})X^w, B^w(X^{2^\nu})X^w\right) = (A_0, \ldots, A_{k-1}, B) \in \text{P-GLWE}_{2^\nu N,k,\chi}$$

with:

$$\begin{aligned}
S_i &= \sum_{j=0}^{N-1} s_{i,j} \cdot X^{j \cdot 2^\nu} = \sum_{j=0}^{2^\nu N-1} \tilde{s}_{i,j} \cdot X^j \textit{ for } i \in [\![0, k[\![\\
A_i &= \sum_{w=0}^{2^\nu-1} A_i^w(X^{2^\nu})X^w = \sum_{w=0}^{2^\nu-1}\sum_{j=0}^{N-1} a_{i,j}^w X^{j\cdot 2^\nu + w} = \sum_{j=0}^{2^\nu N-1} \tilde{a}_{i,j} X^j \textit{ for } i \in [\![0, k[\![\\
B &= \sum_{w=0}^{2^\nu-1} B^w(X^{2^\nu})X^w = \sum_{w=0}^{2^\nu-1}\sum_{j=0}^{N-1} b_j^w X^{j\cdot 2^\nu + w} = \sum_{j=0}^{2^\nu N-1} \tilde{b}_j X^j
\end{aligned}$$

Lets focus on how b_j^w *evolve all along the reduction:*

$$\begin{aligned}
b_j^w &= \sum_{i=0}^{k-1}\left(\sum_{\tau=0}^{j} a_{i,\tau}^w \cdot s_{i,j-\tau} - \sum_{\tau=j+1}^{N-1} a_{i,\tau}^w \cdot s_{i,N+j-\tau}\right) + e_j^w \\
&= \sum_{i=0}^{k-1}\left(\sum_{\tau=0}^{j} \tilde{a}_{i,\tau 2^\nu + w} \cdot \tilde{s}_{i,(j-\tau)2^\nu} - \sum_{\tau=j+1}^{N-1} \tilde{a}_{i,\tau 2^\nu + w} \cdot \tilde{s}_{i,(N+j-\tau)2^\nu}\right) + \tilde{e}_{j2^\nu + w} \\
&= \sum_{i=0}^{k-1}\left(\sum_{\tau=0}^{j2^\nu + w} \tilde{a}_{i,\tau} \cdot \tilde{s}_{i,j2^\nu + w - \tau} - \sum_{\tau=j2^\nu+w+1}^{2^\nu N-1} \tilde{a}_{i,\tau} \cdot \tilde{s}_{i,2^\nu(N+j)+w-\tau}\right) + \tilde{e}_{j2^\nu + w} \\
&= \tilde{b}_{j2^\nu + w}
\end{aligned}$$

Each coefficient is correctly decrypted and each $\tilde{b}_{j2^\nu+w}$ *is equal to* b_j^w*. Moreover, the polynomials* A_i *of the new* $\text{P-GLWE}_{2^\nu N,k,\chi}$ *sample follows the same distribution as the polynomials* A_i^w*, we can make the same remark for the polynomial* B*. To conclude, we have packed several* $\text{GLWE}_{N,k,\chi}$ *ciphertext in one* $\text{P-GLWE}_{2^\nu N,k,\chi}$ *ciphertext by increasing the dimension of this new ciphertext without changing the noise distribution* χ*.* □

This reduction proves that this specific case of partial key is at least as secure as 2^ν GLWE samples. In this case, we showed that increasing the size of the polynomials without changing the noise and slightly changing the key doesn't affect the security (it only increases the number of samples).

Remark 3 (Security of Partial Secret Key) *The reduction presented in Theorem 1 proves that the partial alternate secret keys (Definition 8) problem in $\mathfrak{R}^k_{q,2^\nu N}$ is at least as hard as a* GLWE *problem in $\mathfrak{R}^k_{q,N}$. So adding some zeros in the secret key at specific places doesn't impact the security.*

Now, if we take two GLWE *samples such that the first one is encrypted under an alternate partial key and the second one is encrypted under a secret partial key which have the same amount of unknown coefficients, this two samples should to be indistinguishable. With* LWE *samples, increasing the size of the secret key increases the security, so to keep the same level of security, we can reduce the noise. Regarding this observation and the previous proof, increasing the number of known coefficients in a partial secret key allows us to reduce the noise and keep the same level of security.*

To sum up, to guarantee a given level of security for RLWE samples encrypted under a partial secret key with ϕ random elements, we use the noise parameter given for LWE samples of dimension $n = \phi$ with the same level of security.

Impact of Partial Key on the Noise Distribution. Regarding the security of the partial secret key and the different attacks presented in Section 2.1, we can use the lattice estimator to find out the smallest noise variance σ^2 for an LWE $\in \mathbb{Z}_q^{\phi+1}$ guarantying the desired level of security λ. By using this same σ^2 for GLWE $\in \mathfrak{R}^{k+1}_{q,N}$ with partial secret key $S^{[\phi]}$ we obtain the same level of security λ.

3.2 Advantages of Partial GLWE Secret Keys

Partial GLWE secret keys enable, in many contexts, to have a smaller computational cost for certain algorithms and/or to have a smaller noise growth. This will lead to faster parameter sets (for a given failure probability and security level) after optimization. For more details refer to Section 6.

3.2.1 Advantage with Sample Extraction

Algorithms 6 and 7 in Supplementary Material C explain how to compute a sample extraction (i.e., transforming one of the GLWE ciphertext coefficient into an LWE ciphertext) in the context of a partial GLWE secret key. They are generalizations of the same algorithm used for "traditional" secret keys. Indeed, a traditional secret key is captured when $\phi = k \times N$. We prove the correctness of those algorithms in Supplementary Material A.

Remark 4 (Noise and Cost of Sample Extraction) *A sample extraction, whether it includes a partial secret key or not, does not add any noise to the plaintext. The cost of the sample extraction is also roughly the same and it is negligible.*

3.2.2 Noise Advantage with GLWE Key Switch

A GLWE-to-GLWE key switching with $N \neq 1$, as described in Algorithm 4 in Supplementary Material B, takes as input a GLWE ciphertext $\mathsf{CT}_{\mathsf{in}} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}$ encrypting the plaintext $P \in \mathfrak{R}_{q,N}$ under the secret key $\boldsymbol{S}^{[\phi_{\mathsf{in}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}}$, and outputs $\mathsf{CT}_{\mathsf{out}} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$ encrypting the plaintext $P + E_{\mathsf{KS}} \in \mathfrak{R}_{q,N}$ under the secret key $\boldsymbol{S}^{[\phi_{\mathsf{out}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}}$. The added noise during this procedure E_{KS}, is composed of a rounding error plus a linear combination of the noise from the key switching key ciphertexts. The larger ϕ_{in} is, the larger the amount of rounding error is.

Theorem 2 (Noise of GLWE Key Switch) *After performing a key switching (Algorithm 4), taking as input a* GLWE *ciphertext* $\mathsf{CT}_{\mathsf{in}} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}$ *under the secret key* $\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}}$ *and a key switching key with noise variance* σ_{KSK}^2*, and outputting a* GLWE *ciphertext* $\mathsf{CT}_{\mathsf{out}} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$ *under the secret key* $\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}}$*, the variance of the noise of each coefficient of the output can be estimated by the following formula:*

$$\begin{aligned}\mathsf{Var}\left(\mathsf{CT}_{\mathsf{out}}\right) = \sigma_{\mathsf{in}}^2 + \phi_{\mathsf{in}}\left(\frac{q^2-\beta^{2\ell}}{12\beta^{2\ell}}\right)\left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right) \\ + \frac{\phi_{\mathsf{in}}}{4}\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \ell k_{\mathsf{in}} N \sigma_{\mathsf{ksk}}^2 \frac{\beta^2+2}{12}\end{aligned} \tag{4}$$

where, β and ℓ are the decomposition base and level respectively.

The proof of Theorem 2 is a noise analysis of the result of the operation as done in [CLOT21]. Note that when $\phi_{\mathsf{in}} = k_{\mathsf{in}} \cdot N$ we end up with the same formula given in the mentioned paper. It computes the first step of the decryption, extracts the error and analyzes its variance. We report the detailed proof in Supplementary Material A.

Remark 5 (Cost of GLWE Key Switch) *We recall that the cost of a GLWE-to-GLWE key switching, which is the same whether it involves partial secret keys or not, is:*

$$\begin{aligned}\mathscr{C}\left(\mathsf{FftLweKeySwitch}\right) = k_{\mathsf{in}}\ell \cdot \mathscr{C}\left(\mathbf{FFT}_N\right) + \left(k_{\mathsf{out}}+1\right) \cdot \mathscr{C}\left(\mathbf{iFFT}_N\right) \\ + N k_{\mathsf{in}}\ell \cdot \left(k_{\mathsf{out}}+1\right) \cdot \mathscr{C}\left(\times_{\mathbb{C}}\right) \\ + N \cdot \left(k_{\mathsf{in}}\ell - 1\right) \cdot \left(k_{\mathsf{out}}+1\right) \cdot \mathscr{C}\left(+_{\mathbb{C}}\right)\end{aligned} \tag{5}$$

where $+_{\mathbb{C}}$ and $\times_{\mathbb{C}}$ represent a double-complex addition and multiplication (in the FFT domain) respectively, and $\mathbf{FFT}_N$ (resp. $\mathbf{iFFT}_N$) the Fast Fourier Transform (resp. inverse FFT).

3.2.3 Noise Advantage with Secret Product GLWE Key Switch

A GLWE-to-GLWE key switch also computing a product with a secret polynomial as described in Algorithm 5, follows the exact same definition than above, except that the output ciphertext encrypts $Q \cdot P + E_{\mathsf{KS}}$ with $Q \in \mathfrak{R}_{q,N}$ the secret polynomial hidden in the key switching key. The added noise E_{KS} also depends on the input secret key $\boldsymbol{S}^{[\phi_{\mathsf{in}}]}$ and its filling amount ϕ_{in}. Indeed, this term is the product between the rounding term (dependent on ϕ_{in}) and the polynomial Q.

Theorem 3 (Noise of Secret-Product GLWE Key Switch) *After performing a Secret-Product key switching (Algorithm 5), taking in input a* GLWE *ciphertext* $\mathsf{CT}_{\mathsf{in}} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}$ *under the secret key* $\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}}$ *and a key switching key with noise variance* σ_{KSK}^2 *encrypting a secret message* M_2*, and outputting a* GLWE *ciphertext* $\mathsf{CT}_{\mathsf{out}} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$ *under the secret key* $\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}}$*, the variance of the noise of each coefficient of the output can be estimated by the following formula:*

$$\begin{aligned}\mathsf{Var}\left(\mathsf{CT}_{\mathsf{out}}\right) &= \ell(k_{\mathsf{in}}+1)N\sigma_{\mathsf{KSK}}^2\frac{\beta^2+2}{12}\\ &+ ||M_2||_2^2 \cdot \left(\sigma_{\mathsf{in}}^2 + \left(\frac{q^2-\beta^{2\ell}}{12\beta^{2\ell}}\right)\left(1+\phi_{\mathsf{in}}\left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)+\mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)\right)+\frac{\phi_{\mathsf{in}}}{4}\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)\end{aligned}$$

The proof of this theorem is very similar to the proof of Theorem 2, so we include it in Supplementary Material A.

3.2.4 Noise Advantage with External Product

A GLWE external product is a special case of a secret-product GLWE-to-GLWE key switch where the input secret key and the output secret key are the same. It is pretty easy to compute the noise this procedure will add. The cost to compute a GLWE external product whether it includes a partial secret key or not, is the same.

Theorem 4 (Noise of GLWE External Product) *The external product algorithm is the same as the algorithm of secret-product* GLWE *key switch (Algorithm 5). The only difference is that the external product uses the same key* $\boldsymbol{S}^{[\phi]} \in \mathfrak{R}_{q,N}^{k}$ *as input and as output, and the key switching key is now seen as a* GGSW *ciphertext of message* M_2 *encrypted with noise variance* σ_2^2*. For each coefficient of the output* $\mathsf{CT}_{\mathsf{out}}$*, the noise can be estimated by the following formula:*

$$\begin{aligned}\mathsf{Var}\left(\mathsf{CT}_{\mathsf{out}}\right) &= \ell(k+1)N\sigma_2^2\frac{\beta^2+2}{12}\\ &+ ||M_2||_2^2 \cdot \left(\sigma_{\mathsf{in}}^2 + \left(\frac{q^2-\beta^{2\ell}}{12\beta^{2\ell}}\right)\left(1+\phi\left(\mathsf{Var}\left(\boldsymbol{S}^{[\phi]}\right)+\mathbb{E}^2\left(\boldsymbol{S}^{[\phi]}\right)\right)\right)+\frac{\phi}{4}\mathsf{Var}\left(\boldsymbol{S}^{[\phi]}\right)\right)\end{aligned}$$

Proof 2 (Theorem 4) *This proof is the same than the proof of Theorem 3. Here we have* $k = k_{\mathsf{in}} = k_{\mathsf{out}}$ *and* $\boldsymbol{S}^{[\phi]} = \boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} = \boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}$. □

Noise Advantage with TFHE's PBS. Using a partial GLWE secret key to encrypt a bootstrapping key for TFHE's programmable bootstrapping enables two convenient features: on one hand to have a smaller output LWE ciphertext with less than $k \cdot N + 1$ coefficients, and on the other hand it offers a smaller noise growth in each external product (see Proof in Supplementary Material 11. External product is the main operation used in the CMuxes of the blind rotation), as explained above. The direct consequence of having smaller output ciphertexts is the fact that we can perform smaller LWE-to-LWE key switchings before the next PBS. Furthermore, when $k \cdot N$ is big enough to reach the noise plateau (as explained in Limitation 2), partial secret keys enable to avoid adding unnecessary noise to the bootstrapping.

3.3 LWE-to-LWE Key Switch

Partial GLWE secret key can be used to design a new LWE-to-LWE key switching that is FFT-based. The idea is an adaptation of the work done by Chen *et al.* [CDKS20] but now exploits the use of partial GLWE secret keys. First one casts the input LWE ciphertext into a GLWE ciphertext (Algorithm 10) so we can apply to it a GLWE-to-GLWE key switching (Algorithm 4) to go to a partial GLWE secret key, and finally compute a sample extraction (Algorithm 6). Indeed, the GLWE-to-GLWE key switch can exploit the speed-up coming from the FFT. Details about this new LWE-to-LWE key switch are provided in Algorithm 1.

```
Algorithm 1: $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{FftLweKeySwitch}(\mathsf{ct}_{\mathsf{in}}, \mathsf{KSK})$

Context:
  $n_{\mathsf{in}} \leq k_{\mathsf{in}} \cdot N, \quad n_{\mathsf{out}} \leq k_{\mathsf{out}} \cdot N$
  $\boldsymbol{s}_{\mathsf{in}} = (s_0, \cdots, s_{n_{\mathsf{in}}-1}) \in \mathbb{Z}_q^{n_{\mathsf{in}}}$ : the input LWE secret key
  $\boldsymbol{s}_{\mathsf{out}} = (s'_0, \cdots, s'_{n_{\mathsf{out}}-1}) \in \mathbb{Z}_q^{n_{\mathsf{out}}}$ : the output LWE secret key
  $\ell \in \mathbb{N}$ : the number of levels in the decomposition
  $\beta \in \mathbb{N}$ : the base in the decomposition
  $\boldsymbol{S}_{\mathsf{in}}^{[n_{\mathsf{in}}]} = (S_{\mathsf{in},0}, \cdots, S_{\mathsf{in},k_{\mathsf{in}}-1}) \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}}$ : a partial secret
      key (Definition 6) such that its flattened version is $\boldsymbol{s}_{\mathsf{in}}$
  $\boldsymbol{S}_{\mathsf{out}}^{[n_{\mathsf{out}}]} = (S_{\mathsf{out},0}, \cdots, S_{\mathsf{out},k_{\mathsf{out}}-1}) \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}}$ : a partial secret
      key (Definition 6) such that its flattened version is $\boldsymbol{s}_{\mathsf{out}}$
Input:
  $\mathsf{ct}_{\mathsf{in}} \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{in}}}(p) \subseteq \mathbb{Z}_q^{n_{\mathsf{in}}+1}$, with $p \in \mathbb{Z}_q$
  $\mathsf{KSK} = \{\mathsf{KSK}_i\}_{0 \leq i < k_{\mathsf{in}}}$, with $\mathsf{KSK}_i \in \mathsf{GLEV}^{(\beta,\ell)}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}(S_{\mathsf{in},i})$ : a key switching key
Output: $\mathsf{ct}_{\mathsf{out}} \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{out}}}(p) \subseteq \mathbb{Z}_q^{n_{\mathsf{out}}+1}$

/* Inverse of a constant sample extraction (Algorithm 10)       */
Set $\mathsf{CT} \leftarrow \mathsf{ConstantSampleExtraction}^{-1}(\mathsf{ct}_{\mathsf{in}}, k_{\mathsf{in}}, N) \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}$
/* GLWE-to-GLWE key switch based on the FFT (Algorithm 4)       */
Set $\mathsf{CT}' \leftarrow \mathsf{GlweKeySwitch}(\mathsf{CT}, \mathsf{KSK}) \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$
/* Constant sample extraction (Algorithm 6)                     */
Set $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{ConstantSampleExtract}(\mathsf{CT}') \in \mathbb{Z}_q^{n_{\mathsf{out}}+1}$
return $\mathsf{ct}_{\mathsf{out}}$
```

Remark 6 (Inverse Constant Sample Extraction) *Algorithm 10, presented in Supplementary Material C, trivially casts an LWE ciphertext of size $n+1$ into a*

GLWE ciphertext of size $k+1$ and with polynomials of size N. We obviously need that $n \leq kN$. If $n = kN$, the output is a GLWE ciphertext under a traditional secret key, otherwise it is a GLWE ciphertext under a partial GLWE secret key. Note that the constant term of the output GLWE plaintext is exactly the plaintext of the input LWE ciphertext, however the rest of the coefficients of the output GLWE ciphertext are filled with uniformly random values.

We have the property that for all $p \in \mathbb{Z}_q$, for all $\boldsymbol{s} \in \mathbb{Z}_q^n$, for all $\mathsf{ct} \in \mathsf{LWE}_{\boldsymbol{s}}(p) \subseteq \mathbb{Z}_q^{n+1}$ and for all $(k, N) \in \mathbb{N}^2$ such that $n \leq kN$:

$$\mathsf{ct} = \mathsf{ConstantSampleExtract}\left(\mathsf{ConstantSampleExtract}^{-1}\left(\mathsf{ct}, k, N\right)\right).$$

Theorem 5 (Noise & Cost of FFT-Based LWE Key Switch) *We consider the new LWE-to-LWE key switch as detailed in Algorithm 1. The cost of such key switching is the same as the cost of a GLWE-to-GLWE key switch as introduced in Remark 5 i.e, $\mathscr{C}\left(\mathsf{FftLweKeySwitch}\right) = \mathscr{C}\left(\mathsf{GlweKeySwitch}\right)$.*

The output noise can be expressed from the noise formula of the GLWE-to-GLWE key switch (Theorem 2). To sum up, the output noise is:

$$\mathsf{Var}\left(\mathsf{FftLweKeySwitch}\right) = \mathsf{FftError}_{k_{\mathsf{max}},N,\beta,\ell} + \mathsf{Var}\left(\mathsf{GlweKeySwitch}\right)$$

with $\phi_{\mathsf{in}} = n_{\mathsf{in}}$ and $\phi_{\mathsf{out}} = n_{\mathsf{out}}$, $k_{\mathsf{max}} = \max\left(k_{\mathsf{in}}, k_{\mathsf{out}}\right)$ and $\mathsf{FftError}_{k_{\mathsf{max}},N,\beta,\ell}$ being the error added by the FFT conversions.

Proof 3 (Theorem 5) *The cost is quite straight forward, since we can neglect the complexity of the sample extraction and its inverse. The estimation of the variance of the error is as well immediate. We use the corrective formula introduced in Remark 2 to estimate an upper bound on the FFT error. Indeed, it is easy to see that the FFT-based LWE key switch with k_{in} and k_{out}, is a special case of an external product with $k = \max\left(k_{\mathsf{in}}, k_{\mathsf{out}}\right)$ where some of the ciphertexts composing the* GGSW *are trivial encryptions of 0 or 1 (no noise, all mask elements set to zero and the clear plaintext in the b/B part).* □

Practical Improvement. The use of partial secret keys brings a practical non-negligible improvement to homomorphic computations. Figure 1, presents a comparison between our techniques and the state of the art [CJP21]. More details on the experiments are reported in Section 6.1.

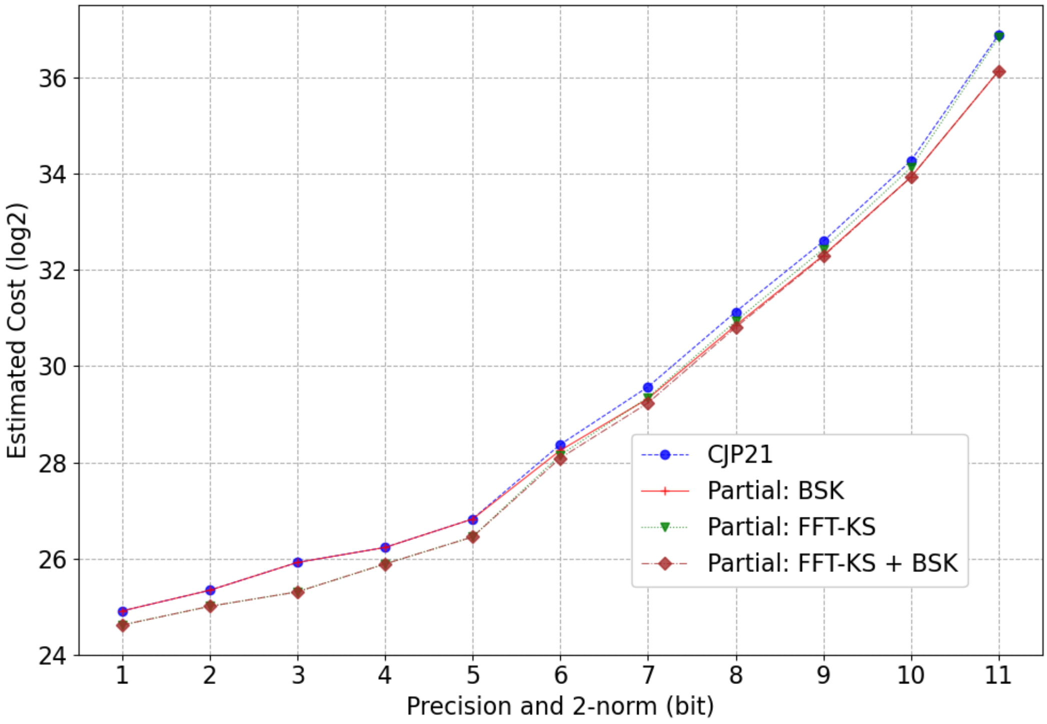

Figure 1: Comparison in terms of estimated computation, between traditional CJP, our baseline, and three variants of CJP based on partial secret keys. Details can be found in Section 6.1 and exact plotted values can be found in Tables 2, 3 and 4.

4 Shared Randomness Secret Keys

To use FHE schemes, one needs to generate several secret keys of different sizes. Our second observation is that instead of sampling those keys independently, we can generate a list of α nested GLWE keys with the same level of security λ. It means that it is publicly known that all the secret coefficients of a smaller key will be included in a larger secret key.

As a simple example we can consider three integers $1 < n_0 < n_1 < n_2$ and a secret key $\boldsymbol{s}^{(2)} \in \mathbb{Z}_q^{n_2}$ generated in the traditional manner (either sampled from an uniform binary/ternary, or a small Gaussian). Let's write it as a concatenation of 3 vectors: $\boldsymbol{s}^{(2)} = \boldsymbol{r}^{(0)} || \boldsymbol{r}^{(1)} || \boldsymbol{r}^{(2)}$. We can now build two smaller secret keys out of $\boldsymbol{s}^{(2)}$ such that for all pair of keys, the smaller one will be included in the bigger one, in its first coefficients:

$$\boldsymbol{s}^{(0)} = \boldsymbol{r}^{(0)} \in \mathbb{Z}_q^{n_0} \quad \text{and} \quad \boldsymbol{s}^{(1)} = \boldsymbol{r}^{(0)} || \boldsymbol{r}^{(1)} \in \mathbb{Z}_q^{n_1}.$$

This kind of secret keys are useful for key switching and for bootstrapping procedures. Note that each of those secret keys will use a different variance for the noise added during encryption: the smaller the secret key, the bigger the variance, so they can all guarantee the same level of security λ.

Definition 9 (GLWE Shared Randomness Secret Keys) *Two GLWE secret keys $\boldsymbol{S} \in \mathfrak{R}_{q,N}^{k}$ and $\boldsymbol{S}' \in \mathfrak{R}_{q,N'}^{k'}$, with $kN \leq k'N'$, are said to share randomness if we have that for all $0 \leq i < kN$, $\bar{s}_i = \bar{s}'_i$, where the $\bar{s}_i$ and the $\bar{s}'_i$ respectively come from the flatten view (Definition 3) of $\boldsymbol{S}$ and $\boldsymbol{S}'$. We note by $\boldsymbol{S} \prec \boldsymbol{S}'$ this relationship between secret keys.*

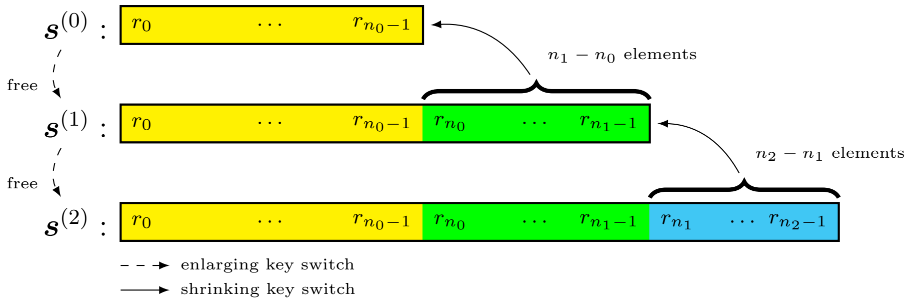

Figure 2: Illustration of simplified key switch procedures between three shared randomness LWE secret keys.

4.1 Hardness of Shared Randomness Secret Keys

Let us consider different samples of GLWE with shared randomness. By taking independently the samples under the same secret key, all the samples are secure and have the same level of security. We now study the level of security of several samples of GLWE considered together with shared secret keys.

First, we present the decisional LWE problem with shared randomness and prove that, under certain conditions, this problem can be reduced to a LWE problem. Next we show that the new operations offered by the shared randomness secret key can not impact the security. Indeed, with this new secret keys using shared randomness, it becomes possible to combine two ciphertexts encrypted under different secret keys.

Definition 10 (Shared Randomness Secret Key Decisional Problem) *Let $n_1 > n_0$. Given a secret key $\boldsymbol{s}^{(0)} \in \mathbb{Z}_q^{n_0}$ following a given distribution $\mathfrak{D}$, a secret $\boldsymbol{r} \in \mathbb{Z}_q^{n_1-n_0}$ following the same distribution $\mathfrak{D}$, and two errors distribution χ_0 and χ_1, we define the LWE with shared randomness secret samples – and we note* $\text{sh-LWE}_{n_0,\chi_0,n_1,\chi_1}$ *– the pairs $\left((\boldsymbol{a_0}, b_0 = \left\langle \boldsymbol{a_0}, \boldsymbol{s}^{(0)} \right\rangle + e_0), (\boldsymbol{a_1}, b_1 = \left\langle \boldsymbol{a_1}, \boldsymbol{s}^{(1)} \right\rangle + e_1)\right) \in \mathbb{Z}_q^{n_0+1} \times \mathbb{Z}_q^{n_1+1}$, where $\boldsymbol{s}^{(1)} = \boldsymbol{s}^{(0)}||\boldsymbol{r}$, $\boldsymbol{a_0} \hookleftarrow \mathcal{U}\left(\mathbb{Z}_q\right)^{n_0}$, $\boldsymbol{a_1} \hookleftarrow \mathcal{U}\left(\mathbb{Z}_q\right)^{n_1}$, $e_0 \hookleftarrow \chi_0$ and $e_1 \hookleftarrow \chi_1$.*

The decision $\text{sh-LWE}_{n_0,\chi_0,n_1,\chi_1}$ *problem consist of distinguishing m independent samples from $\mathcal{U}\left(\mathbb{Z}_q^{n_0+1} \times \mathbb{Z}_q^{n_1+1}\right)$ from m independent samples $((\boldsymbol{a_0}, b_0), (\boldsymbol{a_1}, b_1)) \in$* $\mathrm{LWE}_{n0,\chi_0} \times \mathrm{LWE}_{n1,\chi_1} \subseteq \mathbb{Z}_q^{n_0+1} \times \mathbb{Z}_q^{n_1+1}$ *as defined above.*

Theorem 6 (Hardness of sh-LWE) *If we have three random distributions χ_0, χ_1 and χ' such that, if we sample $e_1 \hookleftarrow \chi_1$ and $e' \hookleftarrow \chi'$, $e_1 + e'$ follows the distribution χ_0. Then* $\text{sh-LWE}_{n_0,\chi_0,n_1,\chi_1}$ *with m samples is at least as hard than* $\mathrm{LWE}_{n_0,\chi_1}$ *with $2m$ samples.*

Remark 7 (Error Distribution χ) *In GLWE-based FHE schemes, χ usually follows a discrete normal distribution. The condition for Theorem 6 is then always verified. In the following, the goal is to use a noise variance σ_0 for χ_0 and a noise variance σ_1 for χ_1 such that $n_0 < n_1$ and $\sigma_0 > \sigma_1$.*

Proof 4 (Theorem 6) *We define an instance of* $\mathrm{LWE}_{n_0,\chi_1}$ *where the samples are encrypted under a secret key $\boldsymbol{s}^{(0)} \in \mathbb{Z}_q^{n_0}$ which follows a given distribution $\mathfrak{D}$, and where for the given distribution χ_1 it exists a distribution χ' such that, for any $e_1 \hookleftarrow \chi_1$ and for any $e' \hookleftarrow \chi'$, we have that $e_1 + e'$ follows a distribution χ_0.*

We now prove that solving the problem $\text{sh-LWE}_{n_0,\chi_0,n_1,\chi_1}$ *is at least as hard than solving the problem* $\mathsf{LWE}_{n_0,\chi_1}$*. To do so, we consider an oracle that can solve the decision* $\text{sh-LWE}_{n_0,\chi_0,n_1,\chi_1}$ *problem and show that a such oracle can solve the decisional* $\mathsf{LWE}_{n_0,\chi_1}$ *instance.*

Observe that, starting from an $\mathsf{LWE}_{n_0,\chi_1}$ *sample, we can easily create either an* $\mathsf{LWE}_{n_0,\chi_0}$ *sample or an* $\mathsf{LWE}_{n_1,\chi_1}$ *sample. To create an* $\mathrm{LWE}_{n_0,\chi_0}$ *sample $(\boldsymbol{a}_0, b_0 = \left\langle \boldsymbol{a}_0, \boldsymbol{s}^{(0)} \right\rangle + e_0) \in \mathbb{Z}_q^{n_0+1}$, with e_0 coming from a distribution χ_0, from an* $\mathrm{LWE}_{n_0,\chi_1}$ *sample $(\boldsymbol{a}_1, b_1 = \left\langle \boldsymbol{a}_1, \boldsymbol{s}^{(0)} \right\rangle + e_1) \in \mathbb{Z}_q^{n_0+1}$, with e_1 coming from a distribution χ_1, we only need to take $\boldsymbol{a}_0 = \boldsymbol{a}_1$ and modify the noise. Following the condition above, it is sufficient to sample $e' \hookleftarrow \chi'$ and then take $b_0 = b_1 + e'$, which makes the new noise in b_0 equal to $e_1 + e'$, which follows the distribution χ_0 as we wanted.*

To create an $\mathrm{LWE}_{n_1,\chi_1}$ *sample $(\boldsymbol{a}_1, b_1 = \left\langle \boldsymbol{a}_1, \boldsymbol{s}^{(1)} \right\rangle + e_1) \in \mathbb{Z}_q^{n_1+1}$ from a* $\mathrm{LWE}_{n_0,\chi_1}$ *sample $(\boldsymbol{a}_0, b_0 = \left\langle \boldsymbol{a}_0, \boldsymbol{s}^{(0)} \right\rangle + e_1) \in \mathbb{Z}_q^{n_0+1}$, we start by generating an random key $r \in \mathbb{Z}_q^{n_1-n_0}$, which follows the same distribution $\mathfrak{D}$ than $s^{(0)}$, as well as a new vector $\boldsymbol{a}' \in \mathfrak{U}\left(\mathbb{Z}_q^{n_1-n_0}\right)$. Then, we take $\boldsymbol{a}_1 = \boldsymbol{a}_0 || \boldsymbol{a}'$ and $b_1 = b_0 + \langle \boldsymbol{a}', \boldsymbol{r}' \rangle$, which give us an* $\mathrm{LWE}_{n_1,\chi_1}$ *sample as expected.*

Following what just described, we observe that given $2m$ $\mathrm{LWE}_{n_0,\chi_1}$ *samples, we can generate m* $\mathrm{LWE}_{n_0,\chi_0}$ *samples and m* $\mathrm{LWE}_{n_1,\chi_1}$ *samples. Now we can provide all the valid samples of* $\mathrm{LWE}_{n_0,\chi_0} \times \mathrm{LWE}_{n_1,\chi_1}$ *to the oracle. Otherwise, when the decisional* $\mathsf{LWE}_{n_0,\chi_1}$ *problem send uniform samples in $\mathbb{Z}_q^{n_0}$, the two transformations proposed before also return uniform samples in $\mathbb{Z}_q^{n_0}$ or in $\mathbb{Z}_q^{n_1}$. As the oracle can solve the decision* $\text{sh-LWE}_{n_0,\chi_0,n_1,\chi_1}$ *problem, we can solve the decision* $\mathsf{LWE}_{n_0,\chi_1}$ *problem.* □

Remark 8 (Security With More Than Two Shared Keys.) *The Proof 4, can easily be adapted to more than only two shared keys.*

Operations Under Shared Randomness Any known homomorphic operation (that we know) that makes two or more ciphertexts interact (encrypted under the same key or different keys) will have as a result a ciphertext with a level of security at least as high as the input with the lowest security level. In light of the common existing attacks, the level of security of a set of GLWE samples encrypted under shared randomness secret keys is then lower bounded by the level of security of the GLWE having the smallest level of security.

As for the partial secret keys, this new type of keys may lead to new unknown attacks and the level of security could be impacted. But at the current state of the art, no attacks seem to have an impact on shared randomness secret key. On the other hand, if one of the key sets is compromised, the other key sets will be impacted consequently.

4.2 Advantages of Shared Randomness Secret Keys

Using shared randomness secret keys enables to speed up homomorphic computations and reduce the amount of noise added by these operations. It is particularly useful for LWE-to-LWE key switch procedures.

4.2.1 Cost & Noise Advantage with LWE-to-LWE Key Switch

Shared randomness secret keys enable to key switch more efficiently and add less noise during the procedure. Figure 2 illustrates key switching processes between three LWE shared randomness secret keys. A key switch from a key to a bigger key is represented with dotted arrows and is called *Enlarging Key Switch.* A key switch from a key to a smaller key is represented with solid arrows and is called *Shrinking Key Switch.*

Enlarging Key Switch. When we consider a ciphertext $\mathsf{ct}_{\mathsf{in}} = (a_0, \cdots, a_{n_1-1}, b) \in \mathsf{LWE}_{\boldsymbol{s}^{(1)}}(m) \subseteq \mathbb{Z}_q^{n_1+1}$ under the secret key $\boldsymbol{s}^{(1)} \in \mathbb{Z}_q^{n_1}$ and we want to key switch it to the secret key $\boldsymbol{s}^{(2)} \in \mathbb{Z}_q^{n_2}$, where $\boldsymbol{s}^{(1)} \prec \boldsymbol{s}^{(2)}$, the algorithm translates into simply adding zeros at the end of the ciphertext:

$$\mathsf{ct}_{\mathsf{out}} := (a_0, \cdots, a_{n_1-1}, 0, \cdots, 0, b) \in \mathsf{LWE}_{\boldsymbol{s}^{(2)}}(m) \subseteq \mathbb{Z}_q^{n_2+1}$$

Algorithm 8, reported in Supplementary Material C, describes this procedure in detail. In this paper we only use this algorithm with LWE ciphertexts, but it can trivially be extended to GLWE ciphertexts as well.

To sum up, with shared randomness secret keys, the enlarging key switchings are basically free and they do not require the use of a public key. They also add no noise, instead of adding a linear combination of freshly encrypted ciphertexts under $\boldsymbol{s}^{(2)}$.

Theorem 7 (Cost & Noise of Enlarging Key Switching) *When working with shared randomness secret keys, the cost of an enlarging key switching (Algorithm 8) is reduced to zero, and the noise in output is the same as the one in input (no noise is added).*

The proof of this theorem is trivial.

Shrinking Key Switch. When we consider a ciphertext $\mathsf{ct}_{\mathsf{in}} = (a_0, \cdots, a_{n_2-1}, b) \in \mathbb{Z}_q^{n_2+1}$ under the secret key $\boldsymbol{s}^{(2)} \in \mathbb{Z}_q^{n_2}$ and we want to key switch it to the secret key $\boldsymbol{s}^{(1)} \in \mathbb{Z}_q^{n_1}$, where $\boldsymbol{s}^{(1)} \prec \boldsymbol{s}^{(2)}$ and $\boldsymbol{s}^{(2)} = \boldsymbol{s}^{(1)} || \boldsymbol{r}^{(2)}$, the algorithm is simplified because of the shared randomness:

1. the parts $(a_0, \cdots, a_{n_1-1})$ and b do not need to be processed but simply reorganized into a temporary ciphertext: $\mathsf{ct} = (a_0, \cdots, a_{n_1-1}, b) \in \mathbb{Z}_q^{n_1+1}$,
2. the part $(a_{n_1}, \cdots, a_{n_2-1})$ has to be key switched, which can be viewed somehow as a traditional key switching algorithm: i.e., key switching the ciphertext $(a_{n_1}, \cdots, a_{n_2-1}, 0) \in \mathbb{Z}_q^{n_2-n_1+1}$ with a key switching key going from the secret key $\boldsymbol{r}^{(2)}$ to $\boldsymbol{s}^{(1)}$, and at the end, add it to ct and return the result.

Algorithm 9, reported in Supplementary Material C, describes this procedure in detail. In this paper we only use this algorithm with LWE ciphertexts, but it can be also trivially extended to GLWE ciphertexts.

To sum up, with shared randomness secret keys, the shrinking key switching requires smaller key switching keys: indeed their size is proportional to $n_2 - n_1$ instead of n_2. As a consequence, the computation is faster, equivalent to key switch a ciphertext of size $n_2 - n_1 + 1$ instead of $n_2 + 1$. Finally, the noise in output is also smaller because the algorithm involves a smaller linear combination of freshly encrypted ciphertexts under $\boldsymbol{s}^{(1)}$.

Theorem 8 (Cost & Noise of Shrinking Key Switching) *We consider two shared randomness secret keys $\boldsymbol{s}^{(0)} \prec \boldsymbol{s}^{(1)}$ with $\boldsymbol{s}^{(0)} \in \mathbb{Z}_q^{n_0}$, $\boldsymbol{s}^{(1)} \in \mathbb{Z}_q^{n_1}$ and $1 < n_0 < n_1$. Let $\beta \in \mathbb{N}^*$ and $\ell \in \mathbb{N}^*$ be the decomposition base and level used in key switching. The cost of shrinking key switching (Algorithm 9) is $\ell(n_1 - n_0)(n_0 + 1)$ integer multiplications and $(\ell(n_1 - n_0) - 1)(n_0 + 1)$ integer additions.*

The added noise during such a shrinking key switching is:

$$\mathsf{Var}(\mathsf{ShrinkingKeySwitch}) = (n_1 - n_0)\left(\frac{q^2 - \beta^{2\ell}}{12\beta^{2\ell}}\right)\left(\mathsf{Var}\left(\boldsymbol{s}_{\mathsf{in}}\right) + \mathbb{E}^2\left(\boldsymbol{s}_{\mathsf{in}}\right)\right) + \frac{(n_1 - n_0)}{4}\mathsf{Var}\left(\boldsymbol{s}_{\mathsf{in}}\right) + \ell \cdot (n_1 - n_0) \cdot \frac{\beta^2 + 2}{12}\sigma_{\mathsf{KSK}}^2$$

The details of this proof can be found in Supplementary Material 12.

4.2.2 Stair Key Switch

In Section 4.2.1, we saw that when one uses different secret keys within an FHE use case, it is convenient to have shared randomness secret keys. However, this concept can be used locally inside a key switch procedure to explore a cost/noise trade-off.

For simplicity, let's consider an FHE use case where there are only two LWE secret keys, and only a key switch from the big one to the small one. We start by setting the two secret keys as shared randomness. The idea here is to add one or many shared randomness secret keys, only during the key switch procedure.

For example, let's assume a fixed decomposition base β, a fixed number of levels ℓ and let $\boldsymbol{s}^{(2)}$ be our big secret key and $\boldsymbol{s}^{(0)}$ be our small (as defined in Section 4.2.1). To key switch from $\boldsymbol{s}^{(2)}$ to $\boldsymbol{s}^{(0)}$, we will this time add one intermediate shared randomness secret key $\boldsymbol{s}^{(1)}$ and compute first a key switch from $\boldsymbol{s}^{(2)}$ to $\boldsymbol{s}^{(1)}$ and then a key switch from $\boldsymbol{s}^{(1)}$ to $\boldsymbol{s}^{(0)}$. This algorithm will be more costly, because its first part will be a linear combination of $(n_2 - n_1)$ ciphertexts of size $n_1 + 1$, and its second part a linear combination of $(n_1 - n_0)$ ciphertexts of smaller size $n_0 + 1$, instead of having a single linear combination of $n_2 - n_0$ ciphertexts of size $n_0 + 1$: so the total number of ciphertexts in the linear combination and in the key switching key has not changed ($n_2 - n_1 + n_1 - n_0 = n_2 - n_0$ as in the key switching from $\boldsymbol{s}^{(2)}$ to $\boldsymbol{s}^{(0)}$), but the linear combinations are slightly more costly and the ciphertexts composing the key switching keys slightly larger. On the other hand, this algorithm adds less noise: indeed its first part has ciphertexts with less noise inside because they are encrypted under a bigger secret key.

Here is the trade-off we want to study. The extreme is to go from $\boldsymbol{s}^{(\mathsf{nb})}$ to $\boldsymbol{s}^{(0)}$ by key switching one element of the key in each key switching, meaning that we will have a total number of $\mathsf{nb} = n_{\mathsf{nb}} - n_0$ shrinking key switching (Algorithm 9) to perform. So nb corresponds to the steps in the stair. This means considering a total number of shared keys equal to $\mathsf{nb} + 1$, including secret keys $\boldsymbol{s}^{(\mathsf{nb})}$ and $\boldsymbol{s}^{(0)}$ which are at the extremes of the stair. We call the added keys between $\boldsymbol{s}^{(\mathsf{nb})}$ and $\boldsymbol{s}^{(0)}$ intermediate secret keys, so we have a total of $\mathsf{nb} - 1$ intermediate secret keys. In practice, this means that we start with coefficient $a_{n_{\mathsf{nb}}-1}$ and key switch it to the secret key with $n_{\mathsf{nb}} - 1$ elements, add it to the rest, and do the same with the next last element, and so on until we reach the desired secret key, one coefficient at a time. The other extreme is when we key switch directly from $\boldsymbol{s}^{(1)}$ and $\boldsymbol{s}^{(0)}$ without intermediary key switchings, so $\mathsf{nb} = 1$.

Algorithm 2 gives details about this procedure. It is important to point out that there are now nb couples of decomposition parameters $(\beta_\alpha, \ell_\alpha)$ for $0 \leq \alpha \leq \mathsf{nb} - 1$, one for each step of the stairs. Note that we could also allow to have more than one such couple per step as well.

Theorem 9 (Cost & Noise of Stair Shrinking Key Switching) *We consider the stair key switch as detailed in Algorithm 2. The cost of such a stair shrinking key switch is $\sum_{\alpha=0}^{\mathsf{nb}-1} \ell_\alpha \left(n_{\alpha+1} - n_\alpha\right)\left(n_\alpha + 1\right)$ integer multiplications and $\sum_{\alpha=0}^{\mathsf{nb}-1} \left(\ell_\alpha \left(n_{\alpha+1} - n_\alpha\right) - 1\right)\left(n_\alpha + 1\right)$ integer additions.*

The added noise during stair shrinking key switching is:

$$\begin{aligned}\mathsf{Var}(\mathsf{StairShrinkKS}) = &\sum_{\alpha=0}^{\mathsf{nb}-1} (n_{\alpha+1} - n_\alpha)\left(\frac{q^2 - \beta_\alpha^{2\ell_\alpha}}{12\beta_\alpha^{2\ell_\alpha}}\right)\left(\mathsf{Var}\left(\boldsymbol{s}^{(\alpha+1)}\right) + \mathbb{E}^2\left(\boldsymbol{s}^{(\alpha+1)}\right)\right) \\ &+ \frac{(n_{\alpha+1} - n_\alpha)}{4}\mathsf{Var}\left(\boldsymbol{s}^{(\alpha+1)}\right) + \ell_\alpha \cdot (n_{\alpha+1} - n_\alpha) \cdot \frac{\beta_\alpha^2 + 2}{12}\sigma^2_{\mathsf{KSK}_\alpha}\end{aligned}$$

Proof 5 (Theorem 9) *The cost and noise of the stair shirnking key switching can be trivially deduced from the Theorem 8. Indeed, at step α of the loop in Algorithm 2, the cost of the shrinking key switching is $\ell_\alpha \left(n_{\alpha+1} - n_\alpha\right)\left(n_\alpha + 1\right)$ integer multiplications and $\left(\ell_\alpha \left(n_{\alpha+1} - n_\alpha\right) - 1\right)\left(n_\alpha + 1\right)$ integer additions.*

```
Algorithm 2: $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{StairKeySwitch}\left(\mathsf{ct}_{\mathsf{in}}, \{\mathsf{KSK}_\alpha\}_{0\le\alpha\le\mathsf{nb}-1}\right)$

Context: $\begin{cases} \mathsf{nb} \in \mathbb{N} : \text{ the number of steps in the algorithm} \\ n_0 < n_1 < \cdots < n_{\mathsf{nb}} \\ \boldsymbol{s}^{(\mathsf{nb})} \in \mathbb{Z}_q^{n_{\mathsf{nb}}} : \text{the input secret key} \\ \boldsymbol{s}^{(0)} \in \mathbb{Z}_q^{n_0} : \text{the output secret key} \\ \boldsymbol{s}^{(\alpha)} \in \mathbb{Z}_q^{n_\alpha}, \forall 1 \le \alpha \le \mathsf{nb}-1 : \text{ intermediate secret keys} \\ \boldsymbol{s}^{(0)} \prec \boldsymbol{s}^{(1)} \prec \cdots \prec \boldsymbol{s}^{(\mathsf{nb})} : \text{ shared randomness secret keys (Definition 9)} \end{cases}$

Input: $\begin{cases} \mathsf{ct}_{\mathsf{in}} \in \mathsf{LWE}_{\boldsymbol{s}^{(\mathsf{nb})}}(p) \subseteq \mathbb{Z}_q^{n_{\mathsf{nb}}+1}, \text{ with } p \in \mathbb{Z}_q \\ \{\mathsf{KSK}_\alpha\}_{0\le\alpha\le\mathsf{nb}-1} : \text{ intermediate key switching key as in Algorithm 9} \\ \qquad\qquad \text{where } \mathsf{KSK}_\alpha \text{ switches from } \boldsymbol{s}^{(\alpha+1)} \text{ to } \boldsymbol{s}^{(\alpha)} \end{cases}$

Output: $\mathsf{ct}_{\mathsf{out}} \in \mathsf{LWE}_{\boldsymbol{s}^{(0)}}(p) \subseteq \mathbb{Z}_q^{n_0+1}$

/* Set the counter to go from $\mathsf{nb}-1$ to 0 */
Set $\alpha := \mathsf{nb} - 1$
/* Set the initial ciphertext */
Set $\mathsf{ct} := \mathsf{ct}_{\mathsf{in}}$
while $\alpha >= 0$ do
    /* Call to Algorithm 9 */
    Update $\mathsf{ct} \leftarrow \mathsf{ShrinkingKeySwitch}(\mathsf{ct}, \mathsf{KSK}_\alpha) \in \mathsf{LWE}_{\boldsymbol{s}^{(\alpha)}}(p) \subseteq \mathbb{Z}_q^{n_\alpha+1}$
    $\alpha := \alpha - 1$
return $\mathsf{ct}_{\mathsf{out}} := \mathsf{ct}$
```

The variance of the noise added at the step α is:

$$\mathsf{Var}(\mathsf{ShrinkKS}_\alpha) = (n_{\alpha+1} - n_\alpha)\left(\frac{q^2 - \beta_\alpha^{2\ell_\alpha}}{12\beta_\alpha^{2\ell_\alpha}}\right)\left(\mathsf{Var}\left(\boldsymbol{s}^{(\alpha+1)}\right) + \mathbb{E}^2\left(\boldsymbol{s}^{(\alpha+1)}\right)\right) + \frac{(n_{\alpha+1} - n_\alpha)}{4}\mathsf{Var}\left(\boldsymbol{s}^{(\alpha+1)}\right) + \ell_\alpha \cdot (n_{\alpha+1} - n_\alpha) \cdot \frac{\beta_\alpha^2 + 2}{12}\sigma^2_{\mathsf{KSK}_\alpha}$$

To obtain the total cost of the algorithm and the total variance of the noise added, it is sufficient to iterate from $\alpha = 0, \dots, \mathsf{nb} - 1$.

□

Remark 9 (Stairs in the Blind Rotation.) *A similar process could be introduced in the blind rotation algorithm. The idea would be, during the blind rotation, to progressively use GLWE partial secret keys (Definition 6) with a smaller filling amount ϕ which will reduce the output noise of the blind rotate. As with the stair shrinking key switch, we could use different bases and levels for the external products offering potentially some overall speed-up.*

Practical Improvement. The use of shared secret keys brings a practical non-negligible improvement to homomorphic computations, has it happened with partial secret keys. Figure 3, presents a comparison between our techniques and the state of the art [CJP21]. More details on the experiments are reported in Section 6.2.

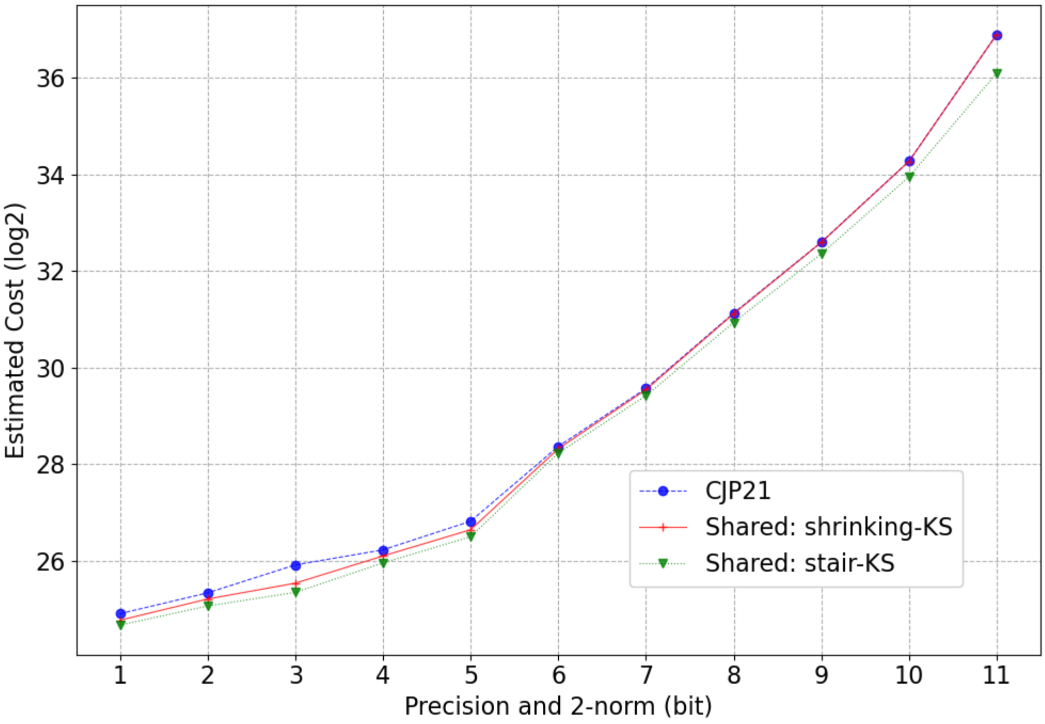

Figure 3: Comparison in terms of estimated computation, between traditional CJP, our baseline, and two variants of CJP based on shared randomness secret keys. Details can be found in Section 6.2 and exact plotted values can be found in Tables 2, 3 and 4.

5 Combining Both Techniques & Their Applications

In this section, we start by providing details on FHE algorithms that benefit from having secret keys that are both partial and shared randomness. Later we describe some nice applications coming from using such types of secret keys.

5.1 Combining Both Techniques

Shared randomness partial GLWE secret keys are a list of partial GLWE secret keys (Section 3) with some public knowledge about shared coefficients (in the exact same way as in Section 4). This type of keys is a combination of shared randomness secret keys and partial secret keys, offering the advantages from both sides.

It is possible to design a faster shrinking key switch (Algorithm 1) which uses partial secret keys (Definition 6). This means that for this faster algorithm, we use both partial secret keys and shared randomness secret keys. Details about this new procedure is given in Algorithm 3.

```
Algorithm 3: $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{FftShrinkingKeySwitch}(\mathsf{ct}_{\mathsf{in}}, \mathsf{KSK})$
Context: $\begin{cases} n_{\mathsf{out}} < n_{\mathsf{in}},\ \ n_{\mathsf{in}} - n_{\mathsf{out}} \leq k_{\mathsf{KSK,in}} \cdot N_{\mathsf{KSK}} \text{ and } n_{\mathsf{out}} \leq k_{\mathsf{KSK,out}} \cdot N_{\mathsf{KSK}} \\ \boldsymbol{s}_{\mathsf{out}} \prec \boldsymbol{s}_{\mathsf{in}} : \text{ shared randomness secret keys (Definition 9)} \\ \boldsymbol{s}_{\mathsf{out}} \in \mathbb{Z}_q^{n_{\mathsf{out}}} : \text{the output LWE secret key} \\ \boldsymbol{s} = (s_{n_{\mathsf{out}}}, \cdots, s_{n_{\mathsf{in}}-1}) \in \mathbb{Z}_q^{n_{\mathsf{in}}-n_{\mathsf{out}}} \\ \boldsymbol{s}_{\mathsf{in}} = \boldsymbol{s}_{\mathsf{out}} || \boldsymbol{s} \in \mathbb{Z}_q^{n_{\mathsf{in}}} : \text{the input LWE secret key} \end{cases}$
Input: $\begin{cases} \mathsf{ct}_{\mathsf{in}} = (a_0, \cdots, a_{n_{\mathsf{in}}-1}, b) \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{in}}}(p) \subseteq \mathbb{Z}_q^{n_{\mathsf{in}}+1}, \text{ where } p \in \mathbb{Z}_q \\ \mathsf{KSK} : \text{ the key switching key suited for Algorithm 1} \end{cases}$
Output: $\mathsf{ct}_{\mathsf{out}} \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{out}}}(p)$
/* Split the input LWE ciphertext into two parts: one related to $\boldsymbol{s}_{\mathsf{out}}$, and
   the rest                                                                       */
Set $\mathsf{ct}_0 := (a_0, \cdots, a_{n_{\mathsf{out}}-1}, b) \in \mathbb{Z}_q^{n_{\mathsf{out}}+1}$
Set $\mathsf{ct}_1 := (a_{n_{\mathsf{out}}}, \cdots, a_{n_{\mathsf{in}}-1}, 0) \in \mathbb{Z}_q^{n_{\mathsf{in}}-n_{\mathsf{out}}+1}$
/* Call Algorithm 1                                                               */
Set $\mathsf{ct}_1' \leftarrow \mathsf{FftLweKeySwitch}(\mathsf{ct}_1, \mathsf{KSK}) \in \mathbb{Z}_q^{n_{\mathsf{out}}+1}$
return $\mathsf{ct}_{\mathsf{out}} = \mathsf{ct}_0 + \mathsf{ct}_1'$
```

Theorem 10 (Noise & Cost of the FFT-Based Shrinking Key Switch) *We consider the FFT-based LWE shrinking key switching as detailed in Algorithm 3. The cost of such a key switching can be expressed expressed with the cost of a GLWE-to-GLWE key switch (Remark 5) since we neglect the cost of a sample extraction and its inverse. The cost is then* $\mathscr{C}(\mathsf{FftShrinkingKeySwitch}) = \mathscr{C}(\mathsf{GlweKeySwitch})$*. Note that the* k_{in} *is smaller thanks to the shared randomness feature of the secret keys, which leads to a faster procedure.*

The added noise can be expressed from the noise formula of the GLWE-to-GLWE key switch (Theorem 2). To sum up, the noise added is $\mathsf{Var}(\mathsf{FftShrinkingKeySwitch}) = \mathsf{FftError}_{k_{\mathsf{max}},N,\beta,\ell} + \mathsf{Var}(\mathsf{GlweKeySwitch})$ *with* $\phi_{\mathsf{in}} = n_{\mathsf{out}} - n_{\mathsf{in}}$ *and* $k_{\mathsf{max}} = \max(k_{\mathsf{in}}, k_{\mathsf{out}})$*.*

Proof 6 (Theorem 10) *The estimation of the variance of the error is immediate. For the FFT error, we refer to Remark 2 and Proof 3.*

Theorem 11 *(Noise of GLWE Key Switching With Partial & Shared Randomness Keys) After performing a key switching (Algorithm 11) from* $\mathsf{CT}_{\mathsf{in}} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}$ *under the secret key* $\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}}$*, to* $\mathsf{CT}_{\mathsf{out}} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$ *under the secret key* $\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}}$*, where the key are shared and partial, i.e.,* $\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \prec \boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}$*, each coefficient of the output has some added noise variance that can be estimated from the following variance:*

$$\begin{aligned}\mathsf{Var}(\mathsf{GlweKeySwitch}') = (\phi_{\mathsf{in}} - \phi_{\mathsf{out}}) \left(\frac{q^2 - \beta^{2\ell}}{12\beta^{2\ell}}\right) \left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right) \\ + \frac{\phi_{\mathsf{in}} - \phi_{\mathsf{out}}}{4} \mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \ell(k_{\mathsf{in}} - k_{\mathsf{out}}) N \sigma_{\mathsf{ksk}}^2 \frac{\beta^2 + 2}{12}\end{aligned}$$

The proof of this theorem can be found in Supplementary Material 13.

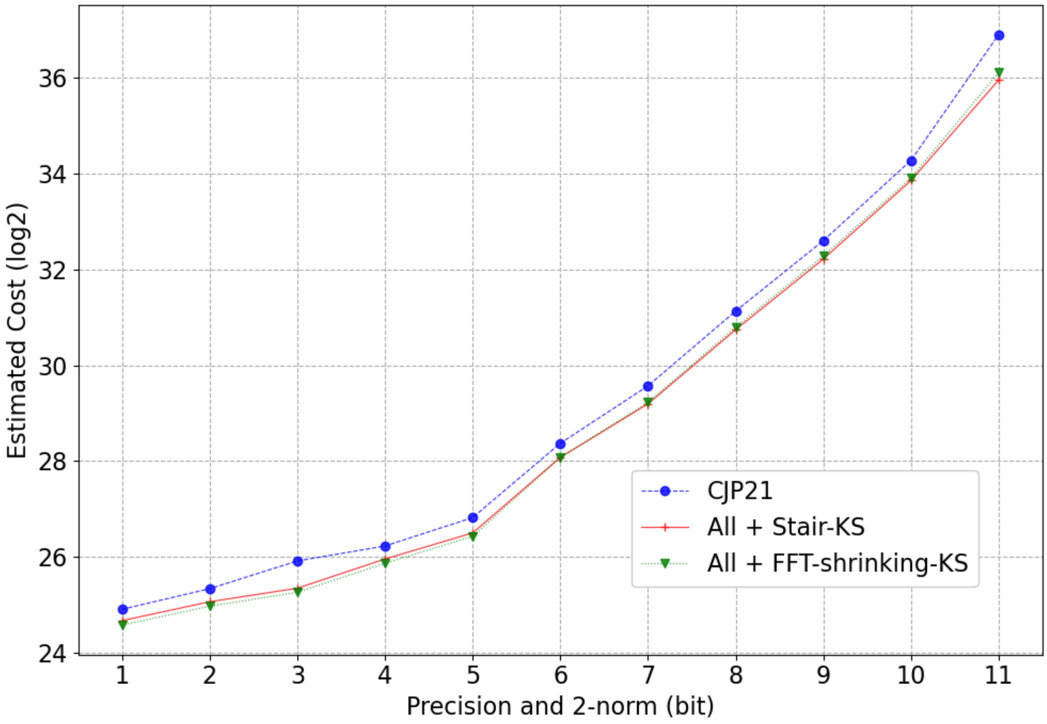

Figure 4: Comparison in terms of estimated computation, between traditional CJP, our baseline, and two variants of CJP based on both partial secret keys and shared randomness secret keys. Details can be found in Section 6.3 and exact plotted values can be found in Tables 2, 3 and 4.

5.2 Some Higher Level Applications

Through Sections 3.2, 4.2 and 5.1, we detailed the many advantages of using partial and/or shared randomness secret keys in FHE algorithms. Now we start giving higher level advantages.

Key Switching Key Compression. When one is deploying an FHE instance, if they use for all their secret keys the shared randomness property, it is then possible to reduce the amount of public material for key switching keys. Indeed, they only need to generate all the shrinking key switching keys (Algorithm 9), from the biggest key to the smallest. All of these shrinking key switching keys are way smaller than the sum of all the traditional key switching keys that would be needed. Note that it is possible to provide more levels in some of the key switching keys, and only use the one that are needed at a moment for a given noise constraint.

Bootstrapping Key Compression. In the same manner, with shared randomness secret keys, it is possible to reduce the amount of public material for bootstrapping keys. When the polynomial size N is shared, since a bootstrapping key is a list of GGSW ciphertexts, each one encrypting a secret key coefficient of the

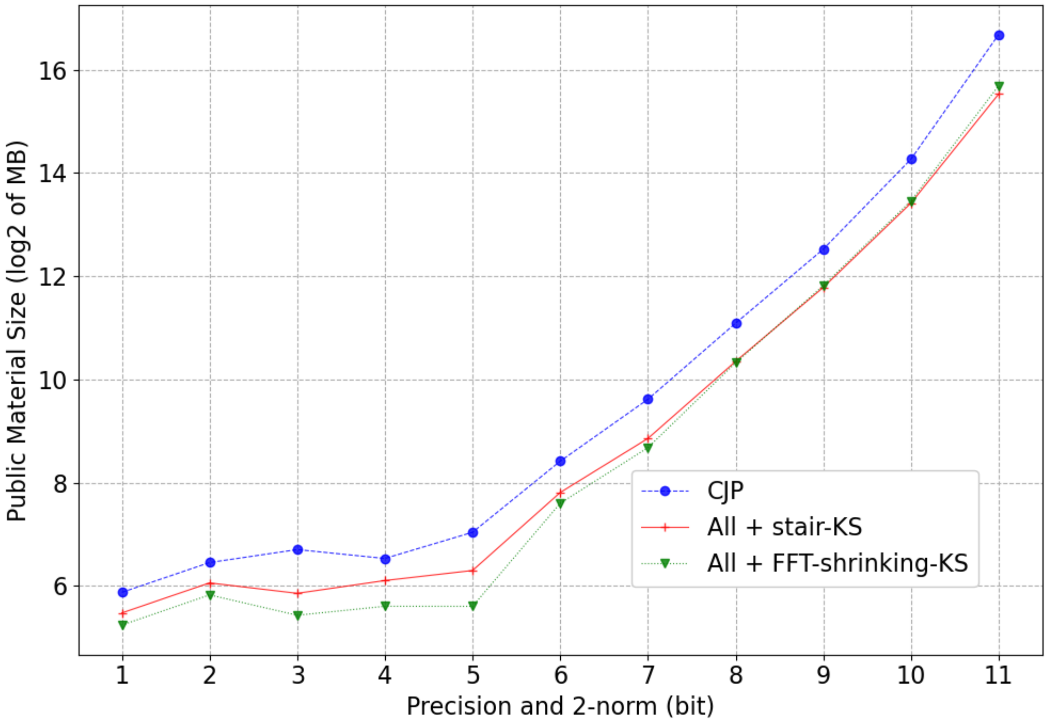

Figure 5: Comparison in terms of size of public material, between traditional CJP, our baseline, and two variants of CJP based on both partial secret keys and shared randomness secret keys. Details can be found in Section 6.3 and exact plotted values can be found in Tables 2, 3 and 4.

input LWE secret key, it is possible to only provide all the GGSW ciphertexts for the longer LWE secret key of the instance. Then, when it is needed to bootstrap an LWE ciphertext with a smaller dimension, one will only use the first part of the bootstrapping key. In the same manner, additional levels can be added, and only used when needed.

Easier Parameter Set Conversion. In [BBB$^+$22], the authors consider use-cases where there are a couple of parameter sets coexisting, and it is necessary to move from one parameter set to the other. Using shared (and partial) secret keys will help converting in a faster manner ciphertexts between the two (or more) parameter sets and will add less noise during the process. Without the shared randomness property, this would require lots of key switching keys and some more computation.

Multikey Compatibility. Both the partial and shared randomness properties are preserved by the MK-FHE approaches (such as [KKL$^+$22, KMS22]) and by threshold-FHE approaches. Indeed, summing two partial secret keys results in another partial secret key, and summing two pairs of shared randomness secret keys together results in a new pair of shared randomness secret keys. Those new secret keys could improve performance of MK-FHE and threshold-FHE, which are in general less efficient that the ones of (single key) FHE, as well as reducing the size of the public material.

Other FHE Schemes. Partial and shared randomness secret keys could be used in other FHE schemes such as FHEW [DM15] or NTRU-based schemes (such as [BIP$^+$22]). This types of keys could also be used in either BFV [Bra12, FV12] or CKKS [CKKS17] if it is needed to have bigger polynomials while keeping the same modulo q, for instance.

Combined With Fixed-Hamming Weight. Both partial and shared randomness secret keys could be instantiated with a fixed Hamming weight if needed.

LWE Encryption Public Key With GLWE Material. If one wants to take advantage of the FFT to encrypt fresh LWE ciphertexts with a secret key $\boldsymbol{s} \in \mathbb{Z}_q^n$, and/or shrink the size of ciphertexts, with partial GLWE secret key, it is possible to provide a GLWE encryption public key for a partial GLWE secret key $\boldsymbol{S}^{[\phi=n]} \in \mathfrak{R}_{q,N}^k$ such that its flatten version is actually $\boldsymbol{s}$. In this case, one could encrypt with a GLWE encryption procedure and sample extract right after to obtain the desired LWE encryption.

6 Parameters & Benchmarks

In this section, we describe how we generated FHE parameters for all our experiments. We use the procedure introduced in [BBB$^+$22] to compare the different approaches. To demonstrate the impact of partial and/or shared randomness secret keys, we will use the Atomic Pattern (AP) called CJP in [BBB$^+$22] (the name coming from the paper [CJP21]).

Below, we explain how we optimized parameters for the different experiments and we show the different improvement (both in computational time and size of public material) brought by each of the new procedures introduced in this paper.

In real life applications, there are additions and multiplications by integers (i.e. a dot product) between two consecutive bootstrappings. Formally, given a list of ciphertexts $\{\mathsf{ct}_i\}_{i\in[\![1,\alpha]\!]} \in (\mathsf{LWE}_{\boldsymbol{s}_{\mathsf{in}}})^\alpha$ (with independent noise values) and a list of integers $\{\omega_i\}_{i\in[\![1,\alpha]\!]} \in \mathbb{Z}^\alpha$, we can compute $\sum_{i=1}^{\alpha} \omega_i \cdot \mathsf{ct}_i$. In that case we have $\nu^2 = \sum_{i=1}^{\alpha} \omega_i^2$. This value ν is used to fully describe the noise growth during a dot product and follows the formalization of [BBB$^+$22]. In this paper, we will set $\nu = 2^p$ with p the precision of the message.

For every experiments below, we set probability of failure $p_{\mathsf{fail}} \leq 2^{-13.9}$. Note that with the FHE parameter generation process used in this paper, is is possible to set it to any other probability.

6.1 Partial GLWE Secret Key

We conducted three experiments with partial GLWE secret keys (Definition 6), and we plotted the results predicted with an optimizer in Figure 1. On the X axis, there is precision p and on the Y axis there is the 2-log of the estimated cost by the optimizer.

Our baseline is CJP, the blue dashed curve with the • symbol.

The first experiment is the CJP atomic pattern where we allow the GLWE secret key to be partial, i.e. it has a filling amount ϕ. On the figure, it is the red solid curve with the + symbol. During optimization, we set ϕ to the minimum between $k \cdot N$ and the value n_{plateau} explained in limitation 2. As expected, we can see an improvement mostly with larger precisions, starting at $p = 6$ where the plateau is actually reached.

The second experiment is the CJP atomic pattern where the traditional LWE-to-LWE key switch is replaced with the FFT-base LWE key switch that we introduced in this paper in Algorithm 1. On the figure, it is the greed dotted curve with the ▼ symbol. During the optimization, we had to introduce new FHE parameters for this particular key switch: an input GLWE dimension k_{in}, an output GLWE dimension k_{out} and a polynomial size N_{KS}. We can see a non-negligible improvement for all precisions when using this key switch, but it is more visible with the smaller precision, between 1 and 6.

The third and last experiment is the combination of the two first ones: we allow the GLWE secret key to be partial (when the plateau is reached) and we use the

FFT-base LWE key switch (Algorithm 1). On the figure, it is the brown dashed curve with the ◆ symbol. As expected the curve is below the two others because it is indeed exploiting the best of the two improvements. We can see a non-negligible improvement for all precisions.

Note that there is no way to build an LWE-to-LWE key switch based on the FFT without partial secret keys, so we cannot compare with this here.

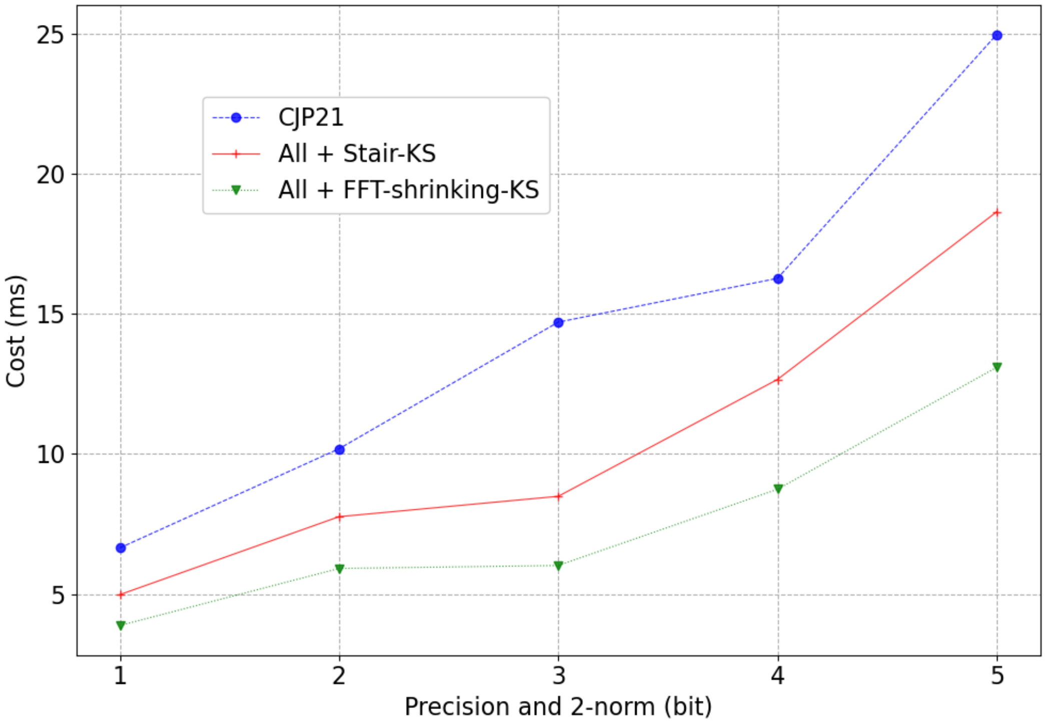

Figure 6: Comparison in terms of time of computation, between traditional CJP, our baseline, and two variants of CJP based on both partial secret keys and shared randomness secret keys. Details can be found in Section 6.3 and exact plotted values can be found in Tables 2, 3 and 4.

6.2 Shared Randomness Secret Keys

We conducted two experiments with shared randomness secret keys (Definition 9), and we plotted the results predicted with an optimizer in Figure 3. This figure follows the same logic as the previous one. Our baseline is still CJP, the blue dashed curve with the • symbol.

The first experiment is the CJP atomic pattern where we allow the secret keys to share their randomness, so we can use the shrinking LWE key switch detailed in Algorithm 9. On the figure, it is the red solid curve with the + symbol. We can see a non-negligible improvement with smaller precisions, until $p = 6$.

The second and last experiment is the CJP atomic pattern where we allow the secret keys to share their randomness, so we can use the 2-step stair LWE key switch detailed in Algorithm 2. On the figure, it is the green dotted curve with the ▼ symbol. We can see a non-negligible improvement at all precisions when we use such a key switch.

Note that if one tries to trivially have a 2-step stair key switch without any shared randomness, the computational cost is basically the same as in CJP.

6.3 Combining Both

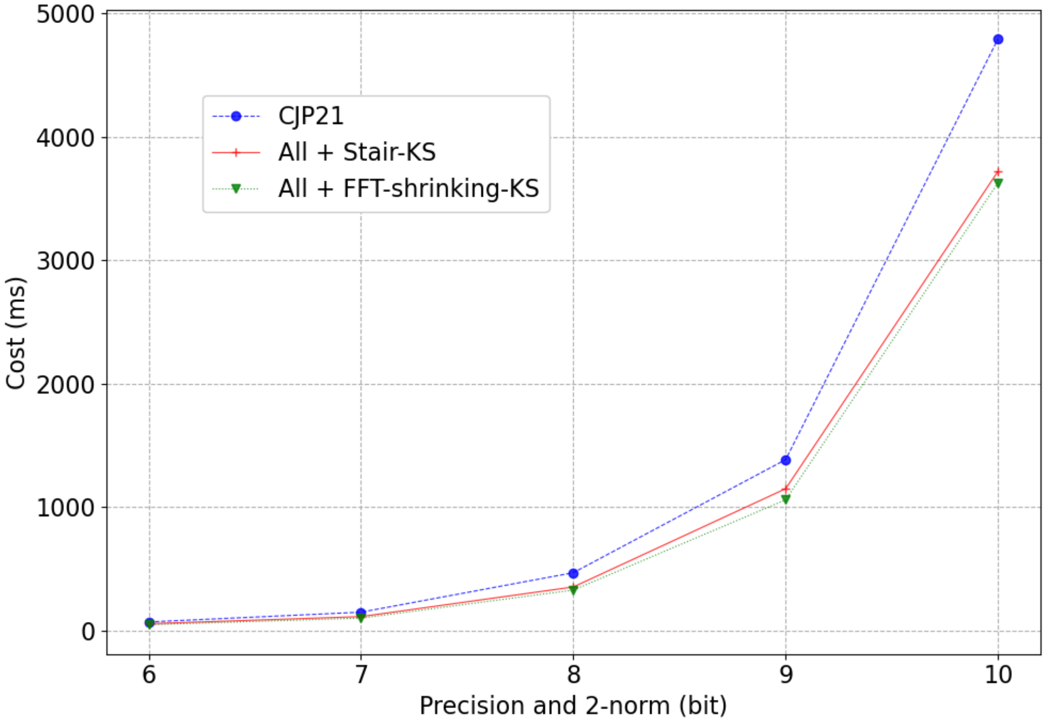

Figure 7: Comparison in terms of time of computation, between traditional CJP, our baseline, and two variants of CJP based on both partial secret keys and shared randomness secret keys. Details can be found in Section 6.3 and exact plotted values can be found in Tables 2, 3 and 4.

We conducted two experiments with both partial (Definition 6) and shared randomness secret keys (Definition 9). First we plotted the predicted computational cost in Figure 4 obtained with an optimizer. This figure follows the same logic as the previous ones. Our baseline is again CJP, the blue dashed curve with the • symbol.

The first experiment is the CJP atomic pattern where we allow the secret keys to be partial and share their randomness. We use the 2-step stair LWE key switch detailed in Algorithm 2 and we allow the GLWE secret key to be partial (when the plateau is reached). On the figure, it is the red solid curve with the + symbol. We can see a non-negligible improvement at all precisions with such a strategy.

The second and last experiment is also the CJP atomic pattern where we allow the secret keys to be partial and share their randomness. We allow the GLWE secret key to be partial (when the plateau is reached), we use the FFT-base LWE key switch (Algorithm 3) since our secret keys also share randomness. On the figure, it is the green dotted curve with the ▼ symbol. We can see a similar improvement at all precisions with such a strategy.

For those experiments, we also plotted the size of the public material needed in Figure 5 to demonstrate their benefit in this matter. The legends corresponding to the experiments are the same as the ones described above for Figure 4. Both the stair key switch curve and the FFT shrinking key switch curve are below our baseline. They actually follow pretty much the curves of the predictions plotted in Figure 4.

For those experiments, we finally plotted the timings we obtained with our benchmarks in Figures 6 and 7 to validate our predictions. The legends corresponding to the experiments are the same as described above for Figure 4, except for the Y axis, which is not logarithmic anymore, so one can easily read the timings. Both the stair key switch curve and the FFT shrinking key switch curve are below our baseline as predicted, and we even have better results with the FFT shrinking key switch than expected. Note that at precision $p = 3$ we have a 2.4 speed-up comparing to the baseline (Figure 6).

All the experiments have been done on AWS with a m6i.metal instance Intel Xeon 8375C (Ice Lake) at 3.5 GHz, with 128 vCPUs and 512.0 GiB of memory using the TFHE-rs library[2].

In Supplementary Material D.2, in Tables 2, 3 and 4 we provide all the parameter sets used to estimate the cost in Figure 4, their benchmarks and their size of public materials.

[2] https://github.com/zama-ai/tfhe-rs

7 Conclusion

To sum up, the traditional way of generating GLWE secret keys leads to unnecessary computation, larger noise growth and bigger public material. In this paper we introduced two (as secure) new ways to generate GLWE secret keys: partial and/or shared randomness. The benefits are indeed non negligible as demonstrated in practical experiments covering a wide range of message precisions. In this paper, we also described several applications exploiting such secret keys.

As future work, it will be interesting to investigate how to optimize the (shrinking) stair key switch in terms of number of steps, their size and decomposition parameters.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 595–618. Springer, 2009.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange-a new hope. In *USENIX security symposium*, volume 2016, 2016.

[AGVW17] Martin R Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to lwe. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 297–322. Springer, 2017.

[Alb17] Martin R Albrecht. On dual lattice attacks against small-secret lwe and parameter choices in helib and seal. In *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part II*, pages 103–129. Springer, 2017.

[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.

[BBB$^+$22] Loris Bergerat, Anas Boudi, Quentin Bourgerie, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Parameter optimization & larger precision for (t)fhe. 2022. https://eprint.iacr.org/2022/704.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

[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, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 188–215. Springer, 2022.

[BJRLW23] Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen. On the hardness of module learning with errors with short distributions. *Journal of Cryptology*, 36(1):1, 2023.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[BLNRL23] Olivier Bernard, Andrea Lesavourey, Tuong-Huy Nguyen, and Adeline Roux-Langlois. Log-s-unit lattices using explicit stickelberger generators to solve approx ideal-svp. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part III*, pages 677–708. Springer, 2023.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. *IACR Cryptology ePrint Archive*, 2012.

[BRL20] Olivier Bernard and Adeline Roux-Langlois. Twisted-phs: Using the product formula to solve approx-svp in ideal lattices. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 349–380. Springer, 2020.

[CDKS20] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) LWE ciphertexts. *IACR Cryptol. ePrint Arch.*, 2020.

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-svp. In *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part I*, pages 324–348. Springer, 2017.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *J. Cryptol.*, 2020.

[CJL$^+$20] Ilaria Chillotti, Marc Joye, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Concrete: Concrete operates on ciphertexts rapidly by extending TfhE. In *WAHC 2020*, 2020.

[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In *CSCML 2021*. Springer, 2021.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT 2017*, 2017.

[CLOT21] Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. In *ASIACRYPT 2021*. Springer, 2021.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT 2015*, 2015.

[DSDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. Lwe with side information: attacks and concrete security estimation. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II*, pages 329–358. Springer, 2020.

[DSGKS18] Dana Dachman-Soled, Huijing Gong, Mukul Kulkarni, and Aria Shahverdi. Partial key exposure in ring-lwe-based cryptosystems: Attacks and resilience. Cryptology ePrint Archive, Paper 2018/1068, 2018. https://eprint.iacr.org/2018/1068.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC 2009*, 2009.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-bkw: Solving lwe using lattice codes. In *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 23–42. Springer, 2015.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013*. Springer, 2013.

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved bkw algorithm for lwe with applications to cryptography and lattices. In *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35*, pages 43–62. Springer, 2015.

[KKL$^+$22] Taechan Kim, Hyesun Kwak, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition. *IACR Cryptol. ePrint Arch.*, page 347, 2022.

[KMS22] Hyesun Kwak, Seonhong Min, and Yongsoo Song. Towards practical multi-key TFHE: parallelizable, key-compatible, quasi-linear complexity. *IACR Cryptol. ePrint Arch.*, page 1460, 2022.

[LMSS23] Changmin Lee, Seonhong Min, Jinyeong Seo, and Yongsoo Song. Faster tfhe bootstrapping with block binary keys. In *ACM ASIACCS 2023*, 2023.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*. Springer, 2010.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[LY23] Kang Hoon Lee and Ji Won Yoon. Discretization error reduction for high precision torus fully homomorphic encryption. In *Public-Key Cryptography–PKC 2023: 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7–10, 2023, Proceedings, Part II*, pages 33–62. Springer, 2023.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. *Post-quantum cryptography*, pages 147–191, 2009.

[PMHS19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-svp in ideal lattices with pre-processing. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part II 38*, pages 685–716. Springer, 2019.

[PV21] Eamonn W Postlethwaite and Fernando Virdia. On the success probability of solving unique svp via bkz. In *Public-Key Cryptography–PKC 2021: 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10–13, 2021, Proceedings, Part I*, pages 68–98. Springer, 2021.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*. ACM, 2005.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66:181–199, 1994.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT 2009*. Springer, 2009.

Supplementary Material

A Proofs

In this section we provide some useful proofs.

Proof 7 (Correctness for Constant Sample Extraction) *As in algorithm 6, we consider a GLWE ciphertext* $\mathsf{CT}_{\mathsf{in}} := (A_0, \cdots, A_{k-1}, B) \in \mathsf{GLWE}_{\boldsymbol{S}^{[\phi]}}(P) \subseteq \mathfrak{R}_{q,N}^{k+1}$ *where* $P = \sum_{i=0}^{N-1} p_i X^i \in \mathfrak{R}_{q,N}$ *and for all* $0 \leq i \leq k-1$ *we have* $A_i = \sum_{j=0}^{N-1} a_{i,j} X^j$ *and* $B = \sum_{j=0}^{N-1} b_j X^j$*. The GLWE secret key is noted* $\boldsymbol{S}^{[\phi]} = (S_0, \cdots, S_{k-1}) \in \mathfrak{R}_{q,N}^k$ *and follows Definition 6. By definition of GLWE ciphertexts, it means that it exists an error polynomial* $E = \sum_{i=0}^{N-1} e_i X^i \in \mathfrak{R}_{q,N}$ *such that* $B - \sum_{i=0}^{k-1} A_i \cdot S_i = P + E$*.*

Following Algorithm 6, the constant sample extraction outputs the following LWE ciphertext: $\mathsf{ct}_{\mathsf{out}} = (a_{\mathsf{out},0}, \cdots, a_{\mathsf{out},\phi-1}, b_{\mathsf{out}}) \in \mathsf{LWE}_{\bar{\boldsymbol{s}}}(p_0) \subseteq \mathbb{Z}_q^{\phi+1}$ *encrypted under the LWE secret key* $\bar{\boldsymbol{s}} = (\bar{s}_0, \cdots, \bar{s}_{\phi-1}) \in \mathbb{Z}_q^\phi$ *obtained as defined in Definition 7.*

First we define two index functions, the first one is $\iota : i \mapsto \left(\left\lfloor \frac{i}{N} \right\rfloor, i \bmod N\right)$ *and the second one is* $\tilde{\iota} : i \mapsto \left(\left\lfloor \frac{i}{N} \right\rfloor, (N-i) \bmod N\right)$*. We also need to define a last function* $\gamma : i \mapsto 1 - ((i \bmod N) == 0)$

$$\begin{aligned} b_{\mathsf{out}} - \sum_{i=0}^{\phi-1} a_{\mathsf{out},i} \cdot \bar{s}_i &= b_0 - \sum_{i=0}^{\phi-1} a_{\mathsf{out},i} \cdot \bar{s}_i - \underbrace{\sum_{i=\phi}^{kN-1} (-1)^{\gamma(i)} \cdot a_{\tilde{\iota}(i)} \cdot s_{\iota(i)}}_{\text{null since all } s_{\iota(i)}=0 \text{ because it is a partial key)}} \\ &= b_0 - \sum_{i=0}^{\phi-1} \underbrace{(-1)^{\gamma(i)} \cdot a_{\tilde{\iota}(i)} \cdot s_{\iota(i)}}_{\text{lines 2 and 3 in Algorithm 6}} - \sum_{i=\phi}^{kN-1} (-1)^{\gamma(i)} \cdot a_{\tilde{\iota}(i)} \cdot s_{\iota(i)} \\ &= b_0 - \sum_{i=0}^{kN-1} (-1)^{\gamma(i)} \cdot a_{\tilde{\iota}(i)} \cdot s_{\iota(i)} = b_0 - \sum_{i=0}^{k-1} \sum_{j=0}^{N-1} a_{i,N-j} \cdot s_{i,j} \\ &= b_0 - \sum_{i=0}^{k-1} \sum_{j=0}^{N-1} a_{i,(N-j) \bmod N} X^{(N-j) \bmod N} \cdot s_{i,j} X^j \end{aligned} \tag{6}$$

This quantity is what we have on the constant term of the polynomial resulting from the decryption of $\mathsf{CT}_{\mathsf{in}}$*:*

$$\begin{aligned} B - \sum_{i=0}^{k-1} A_i \cdot S_i &= B - \sum_{i=0}^{k-1} \sum_{j=0}^{N-1} \sum_{j'=0}^{N-1} a_{i,j} X^j \cdot s_{i,j'} X^{j'} \\ &= \underbrace{X^0 \cdot \left(b_0 - \sum_{i=0}^{k-1} \sum_{j=0}^{N-1} a_{i,(N-j) \bmod N} X^{(N-j) \bmod N} \cdot s_{i,j} X^j \right)}_{\text{constant coefficient with the same quantity}} \\ &\quad \underbrace{+ X^1 \cdot (b_1 - \ldots) \quad + \quad \cdots \quad + X^{N-1} \cdot (b_{N-1} - \ldots)}_{\text{non-constant coefficients}} \end{aligned} \tag{7}$$

□

Proof 8 (Correctness for Sample Extraction) *We follow the context and inputs of Algorithm 7. It is trivial to show that the α-th coefficient of the decryption of $\mathsf{CT}_{\mathsf{in}}$ is equal to what is in the constant coefficient of $X^{-\alpha} \cdot \mathsf{CT}_{\mathsf{in}}$.*

□

Proof 9 (Theorem 2) *The inputs of a* GLWE*-to-*GLWE *key switching (Algorithm 4) are:*

- *The input GLWE ciphertext: $\mathsf{CT}_{\mathsf{in}} = (\boldsymbol{A}_{\mathsf{in}}, B_{\mathsf{in}}) \in \mathrm{GLWE}_{\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}}(\Delta \cdot M) \subseteq \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}$, where $B_{\mathsf{in}} = \sum_{i=0}^{k_{\mathsf{in}}-1} A_{\mathsf{in},i} \cdot S_{\mathsf{in},i} + \Delta \cdot M + E_{\mathsf{in}}$, $A_{\mathsf{in},i} = \sum_{j=0}^{N-1} a_{i,j} \cdot X^j \hookleftarrow \mathcal{U}(\mathfrak{R}_{q,N})$ for all $i \in [\![0, k[\![$ and $E_{\mathsf{in}} = \sum_{j=0}^{N-1} e_j \cdot X^j$, and $e_j \hookleftarrow \mathcal{N}_{\sigma_{\mathsf{in}}^2}$ for all $j \in [\![0, N-1[\![$.*
- *The key switch key: $\mathsf{KSK} = (\mathsf{KSK}_0, \ldots, \mathsf{KSK}_{k_{\mathsf{in}}-1})$, where $\mathsf{KSK}_i \in \mathsf{GLEV}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}^{(\beta,\ell)}(S_{\mathsf{in},i}) = \left(\mathrm{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta} S_{\mathsf{in},i}\right), \cdots, \mathrm{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta^\ell} S_{\mathsf{in},i}\right)\right)$ for all $0 \leq i < k_{\mathsf{in}}$. We note by $\mathsf{KSK}_{i,j} = (\boldsymbol{A_{i,j}}, B_{i,j}) \in \mathrm{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta^{j+1}} S_{\mathsf{in},i}\right)$, for all $0 \leq i < k_{\mathsf{in}}$ and for all $0 \leq j < \ell$, where $B_{i,j} = \sum_{\tau=0}^{k_{\mathsf{out}}-1} A_{i,j,\tau} \cdot S_{\mathsf{out},\tau}^{[\phi_{\mathsf{out}}]} + \frac{q}{\beta^{j+1}} S_{\mathsf{in},i} + E_{\mathsf{ksk},i,j}$, and $E_{\mathsf{ksk},i,j} = \sum_{\tau=0}^{N-1} e_{\mathsf{ksk},i,j,\tau} \cdot X^\tau$ and $e_{\mathsf{ksk},i,j,\tau} \hookleftarrow \mathcal{N}_{\sigma_{\mathsf{ksk}}^2}$.*

The output of this algorithm is: $\mathsf{CT}_{\mathsf{out}} = (\boldsymbol{A}_{\mathsf{out}}, B_{\mathsf{out}}) \in \mathrm{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}(\Delta \cdot M) \subseteq \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$. By definition, in the decomposition described in Supplementary Material B, we have that $\mathbf{Dec}^{(\beta,\ell)}(A_{\mathsf{in},i}) = \left(\widetilde{A}_{\mathsf{in},i,0}, \cdots, \widetilde{A}_{\mathsf{in},i,\ell-1}\right)$ such that $\widetilde{A}_{\mathsf{in},i} = \sum_{j=0}^{\ell-1} \frac{q}{\beta^{j+1}} \widetilde{A}_{\mathsf{in},i,j}$, for all $0 \leq i < k_{\mathsf{in}}$.
Let define $\bar{A}_{\mathsf{in},i} = A_{\mathsf{in},i} - \widetilde{A}_{\mathsf{in},i}$, $|\bar{a}_{i,\tau}| = |a_{i,\tau} - \widetilde{a}_{i,\tau}| < \frac{q}{2\beta^\ell}$, $\bar{a}_{i,\tau} \in \left[\!\!\left[\frac{-q}{2\beta^\ell}, \frac{q}{2\beta^\ell}\right[\!\!\right[$ for all $0 \leq \tau < N$. So we have that their expectations and variances are respectively $\mathbb{E}(\bar{a}_{i,\tau}) = -\frac{1}{2}$, $\mathsf{Var}(\bar{a}_{i,\tau}) = \frac{q^2}{12\beta^{2\ell}} - \frac{1}{12}$, $\mathbb{E}(\widetilde{a}_{i,\tau}) = -\frac{1}{2}$ and $\mathsf{Var}(\widetilde{a}_{i,\tau}) = \frac{\beta^2-1}{12}$.
Now, we can decrypt:

$$\begin{aligned}
B_{\mathsf{out}} - \left\langle \boldsymbol{A}_{\mathsf{out}}, \boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \right\rangle &= \left\langle (\boldsymbol{A}_{\mathsf{out}}, B_{\mathsf{out}}), \left(-\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}, 1\right) \right\rangle \\
&= \left\langle (\boldsymbol{0}, B_{\mathsf{in}}) - \sum_{i=0}^{k_{\mathsf{in}}-1} \mathbf{Dec}^{(\beta,\ell)}(A_{\mathsf{in},i}) \cdot \mathsf{KSK}_i, \left(-\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}, 1\right) \right\rangle \\
&= B_{\mathsf{in}} - \sum_{i=0}^{k_{\mathsf{in}}-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},i,j} \left\langle \mathsf{KSK}_{i,j}, \left(-\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}, 1\right) \right\rangle \\
&= B_{\mathsf{in}} - \sum_{i=0}^{k_{\mathsf{in}}-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},i,j} \left(\frac{q}{\beta^{j+1}} S_{\mathsf{in},i} + E_{\mathsf{ksk},i,j} \right) \\
&= \underbrace{B_{\mathsf{in}} - \sum_{i=0}^{k_{\mathsf{in}}-1} \widetilde{A}_{\mathsf{in},i} S_{\mathsf{in},i}}_{(I)} - \underbrace{\sum_{i=0}^{k_{\mathsf{in}}-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},i,j} \cdot E_{\mathsf{ksk},i,j}}_{(II)}
\end{aligned}$$

Now let's focus on the w^{th} *coefficient of part* (I)*:*

$$\begin{aligned}
&b_{\mathsf{in},w} - \sum_{i=0}^{k_{\mathsf{in}}-1}\left(\sum_{\tau=0}^{w}\widetilde{a}_{\mathsf{in},i,w-\tau}\cdot s_{\mathsf{in},i,\tau} - \sum_{\tau=w+1}^{N-1}\widetilde{a}_{\mathsf{in},i,N+w-\tau}\cdot s_{\mathsf{in},i,\tau}\right)\\
&= b_{\mathsf{in},w} - \sum_{i=0}^{k_{\mathsf{in}}-1}\left(\sum_{\tau=0}^{w}\left(a_{\mathsf{in},i,w-\tau} - \bar{a}_{\mathsf{in},i,w-\tau}\right)\cdot s_{\mathsf{in},i,\tau} - \sum_{\tau=w+1}^{N-1}\left(a_{\mathsf{in},i,N+w-\tau} - \bar{a}_{\mathsf{in},i,N+w-\tau}\right)\cdot s_{\mathsf{in},i,\tau}\right)\\
&= \Delta m_w + e_w + \sum_{i=0}^{k_{\mathsf{in}}-1}\left(\sum_{\tau=0}^{w}\bar{a}_{\mathsf{in},i,w-\tau}\cdot s_{\mathsf{in},i,\tau} - \sum_{\tau=w+1}^{N-1}\bar{a}_{\mathsf{in},i,N+w-\tau}\cdot s_{\mathsf{in},i,\tau}\right)
\end{aligned}$$

Now let's focus on the w^{th} *coefficient of part* (II)*:*

$$\sum_{i=0}^{k_{\mathsf{in}}-1}\sum_{j=0}^{\ell-1}\left(\sum_{\tau=0}^{w}\widetilde{a}_{\mathsf{in},i,j,w-\tau}\cdot e_{\mathsf{ksk},i,j,\tau} - \sum_{\tau=w+1}^{N-1}\widetilde{a}_{\mathsf{in},i,j,N+w-\tau}\cdot e_{\mathsf{ksk},i,j,\tau}\right)$$

We can now isolate the output error for the w^{th} *coefficient and remove the message coefficient. We obtain that the output error is:*

$$\begin{aligned}
e'_w = e_w + &\underbrace{\sum_{i=0}^{k_{\mathsf{in}}-1}\left(\sum_{\tau=0}^{w}\bar{a}_{\mathsf{in},i,w-\tau}\cdot s_{\mathsf{in},i,\tau} - \sum_{\tau=w+1}^{N-1}\bar{a}_{\mathsf{in},i,N+w-\tau}\cdot s_{\mathsf{in},i,\tau}\right)}_{(*)}\\
&+ \sum_{i=0}^{k_{\mathsf{in}}-1}\sum_{j=0}^{\ell-1}\left(\sum_{\tau=0}^{w}\widetilde{a}_{\mathsf{in},i,j,w-\tau}\cdot e_{\mathsf{ksk},i,j,\tau} - \sum_{\tau=w+1}^{N-1}\widetilde{a}_{\mathsf{in},i,j,N+w-\tau}\cdot e_{\mathsf{ksk},i,j,\tau}\right)
\end{aligned}$$

Observe that in the term $(*)$ *there are* $k_{\mathsf{in}}N - \phi_{\mathsf{in}}$ *terms of type* $\bar{a}_{\mathsf{in},i,\cdot}\cdot s_{\mathsf{in},i,\cdot}$ *that are equal to* 0*. So we have:*

$$\begin{aligned}
\mathsf{Var}(e'_w) &= \mathsf{Var}(e_w) + \phi_{\mathsf{in}}\cdot\mathsf{Var}\left(\bar{a}_{\mathsf{in},i,\cdot}\cdot s_{\mathsf{in},i,\cdot}\right) + k_{\mathsf{in}}\cdot\ell\cdot N\cdot\mathsf{Var}\left(\widetilde{a}_{\mathsf{in},i,j,\cdot}\cdot e_{\mathsf{ksk},i,j,\cdot}\right)\\
&= \sigma_{\mathsf{in}}^2 + \phi_{\mathsf{in}}\left(\mathsf{Var}\left(\bar{a}_{\mathsf{in},i,\cdot}\right)\mathsf{Var}\left(s_{\mathsf{in},i,\cdot}\right) + \mathsf{Var}\left(\bar{a}_{\mathsf{in},i,\cdot}\right)\mathbb{E}^2\left(s_{\mathsf{in},i,\cdot}\right) + \mathbb{E}^2(\bar{a}_{\mathsf{in},i,\cdot})\mathsf{Var}\left(s_{\mathsf{in},i,\cdot}\right)\right)\\
&\quad + \ell k_{\mathsf{in}}N\left(\mathsf{Var}\left(\widetilde{a}_{\mathsf{in},i,j,\cdot}\right)\mathsf{Var}\left(e_{\mathsf{ksk},i,j,\cdot}\right) + \mathbb{E}^2\left(\widetilde{a}_{\mathsf{in},i,j,\cdot}\right)\mathsf{Var}\left(e_{\mathsf{ksk},i,j,\cdot}\right)\right.\\
&\quad \left.+ \mathsf{Var}\left(\widetilde{a}_{\mathsf{in},i,j,\cdot}\right)\mathbb{E}^2\left(e_{\mathsf{ksk},i,j,\cdot}\right)\right)\\
&= \sigma_{\mathsf{in}}^2 + \phi_{\mathsf{in}}\left(\frac{q^2-\beta^{2\ell}}{12\beta^{2\ell}}\right)\left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)\\
&\quad + \frac{\phi_{\mathsf{in}}}{4}\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \ell k_{\mathsf{in}}N\frac{\beta^2+2}{12}\sigma_{ksk}^2.
\end{aligned}$$

□

Proof 10 (Theorem 3) *This proof is similar to the proof proposed for the key switch with partial key (proof 9).*
The inputs of secret-product GLWE key switch (Algorithm 5) are:

- *The input GLWE ciphertext:* $\mathsf{CT}_{\mathsf{in}} = (\boldsymbol{A}_{\mathsf{in}}, B_{\mathsf{in}}) \in \mathrm{GLWE}_{\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}}(\Delta\cdot M_1) \subseteq \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}$*, where* $B_{\mathsf{in}} = \sum_{i=0}^{k_{\mathsf{in}}-1} A_{\mathsf{in},i}\cdot S_{\mathsf{in},i}^{[\phi_{\mathsf{in}}]} + \Delta\cdot M_1 + E_{\mathsf{in}}$*,* $A_{\mathsf{in},i} = \sum_{j=0}^{k-1} a_{i,j}\cdot X^j \hookleftarrow \mathcal{U}\left(\mathfrak{R}_{q,N}\right)$ *for all* $i \in \llbracket 0, k\llbracket$ *and* $E_{\mathsf{in}} = \sum_{j=0}^{k-1} e_j\cdot X^j$*, and* $e_j \hookleftarrow \mathcal{N}_{\sigma_{\mathsf{in}}^2}$ *for all* $j \in \llbracket 0, N-1\llbracket$*.*

- *The secret product key switch key :* $\mathsf{KSK} = (\mathsf{KSK}_0, \ldots, \mathsf{KSK}_{k_{\mathsf{in}}})$*, where* $\mathsf{KSK}_i \in \mathsf{GLEV}^{(\beta,\ell)}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(-M_2 S^{[\phi_{\mathsf{in}}]}_{\mathsf{in},i}\right) = \left(\mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(-\frac{qM_2}{\beta} S^{[\phi_{\mathsf{in}}]}_{\mathsf{in},i}\right), \cdots, \mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(-\frac{qM_2}{\beta^{\ell}} S^{[\phi_{\mathsf{in}}]}_{\mathsf{in},i}\right)\right)$ *for all* $0 \leq i \leq k_{\mathsf{in}}$ *(for this proof, we define* $S_{k_{\mathsf{in}}} = -1$*). We note by* $\mathsf{KSK}_{i,j} = (\boldsymbol{A_{i,j}}, B_{i,j}) \in \mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(-\frac{qM_2}{\beta^{j+1}} S^{[\phi_{\mathsf{in}}]}_{\mathsf{in},i}\right)$*, for all* $0 \leq i < k_{\mathsf{in}}$ *and for all* $0 \leq j < \ell$*, where* $B_{i,j} = \sum_{\tau=0}^{k_{\mathsf{out}}-1} A_{i,j,\tau} \cdot S^{[\phi_{\mathsf{out}}]}_{\mathsf{out},\tau} + \frac{qM_2}{\beta^{j+1}} S^{[\phi_{\mathsf{in}}]}_{\mathsf{in},i} + E_{\mathsf{ksk},i,j}$*, and* $E_{\mathsf{ksk},i,j} = \sum_{\tau=0}^{N-1} e_{\mathsf{ksk},i,j,\tau} \cdot X^{\tau}$ *and* $e_{\mathsf{ksk},i,j,m} \hookleftarrow \mathcal{N}_{\sigma^2_{\mathsf{ksk}}}$*.*

In output we obtain: $\mathsf{CT}_{\mathsf{out}} = (\boldsymbol{A}_{\mathsf{out}}, B_{\mathsf{out}}) \in \mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}(\Delta \cdot M_1 \cdot M_2) \subseteq \mathfrak{R}^{k_{\mathsf{out}}+1}_{q,N}$*.*
By definition, for any random polynomial A_i*, we have* $A_i = \sum_{j=0}^{N-1} a_{i,j} \cdot X^j$ *where* $a_{i,j} \sim \mathbb{U}(\mathbb{Z}_q)$*,(* $a_{i,j} \in \llbracket \frac{-q}{2}, \frac{q}{2} \rrbracket$*).*
By definition, for the decomposition (describe in Annex B), we have $\mathbf{Dec}^{(\beta,\ell)}(A_i) = \left(\tilde{A}_{i,0}, \cdots, \tilde{A}_{i,\ell-1}\right)$ *such that* $\tilde{A}_i = \sum_{j=0}^{\ell-1} \frac{q}{\beta^{j+1}} \tilde{A}_{i,j}$*.*
Let define $\bar{A}_i = |A_i - \tilde{A}_i|$*,* $\bar{a}_{i,j} = |a_{i,j} - \tilde{a_{i,j}}| < \frac{q}{2\beta^{\ell}}$*;* $\bar{a}_{i,j} \in \left\llbracket \frac{-q}{2\beta^{\ell}}, \frac{q}{2\beta^{\ell}} \right\llbracket$*. Finally we obtain:* $\mathbb{E}(\bar{a}_i) = -\frac{1}{2}$*;* $\mathsf{Var}(\bar{a}_i) = \frac{q^2}{12\beta^{2\ell}} - \frac{1}{12}$*;* $\mathbb{E}(\tilde{a}_{i,j}) = -\frac{1}{2}$*;* $\mathsf{Var}(\tilde{a}_i) = \frac{\beta^2-1}{12}$*.*
As B_i *is seen as an uniform polynomial, we obtain the same results for the variance and the expectation for* $\tilde{B}_i$ *(resp.* $\bar{B}_i$*) than* $\tilde{A}_i$ *(Resp.* $\bar{A}_i$*). In the next calculations,* $\tilde{B}_{\mathsf{in},j} \cdot E_j$ *will be write as* $-\tilde{A}_{\mathsf{in},k_{\mathsf{in}},j} \cdot E_{k_{\mathsf{in}},j}$
Now, we can compute the decryption:

$$\begin{aligned}
B_{\mathsf{out}} - \left\langle \boldsymbol{A}_{\mathsf{out}}, \boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}} \right\rangle =& \left\langle (\boldsymbol{A}_{\mathsf{out}}, B_{\mathsf{out}}); \left(-\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}, 1\right) \right\rangle \\
=& \left\langle \mathbf{Dec}^{(\beta,\ell)}(B_{\mathsf{in}}) \cdot \mathsf{KSK}_{k_{\mathsf{in}}} + \sum_{i=0}^{k_{\mathsf{in}}-1} \mathbf{Dec}^{(\beta,\ell)}(A_{\mathsf{in},i}) \cdot \mathsf{KSK}_i; \left(-\boldsymbol{S}^{[\phi_{\mathsf{out}}]}, 1\right) \right\rangle \\
=& M_2 \left(\tilde{B}_{\mathsf{in},i} - \sum_{i=0}^{k_{\mathsf{in}}-1} \tilde{A}_{\mathsf{in},i} \cdot S_{\mathsf{in},i}\right) - \sum_{i=0}^{k_{\mathsf{in}}} \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},i,j} \cdot E_{i,j} \\
=& M_2 \left(\Delta M_1 + E_{\mathsf{in}} + \bar{B}_{\mathsf{in},i} - \sum_{i=0}^{k_{\mathsf{in}}-1} \bar{A}_{\mathsf{in},i} \cdot S_{\mathsf{in},i}\right) - \sum_{i=0}^{k_{\mathsf{in}}} \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},i,j} \cdot E_{i,j} \\
=& \Delta M_2 \cdot M_1 + M_2 \left(E_{\mathsf{in}} + \bar{B}_{\mathsf{in},i} - \sum_{i=0}^{k_{\mathsf{in}}-1} \bar{A}_{\mathsf{in},i} \cdot S_{\mathsf{in},i}\right) - \sum_{i=0}^{k_{\mathsf{in}}} \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},i,j} \cdot E_{i,j}
\end{aligned}$$

By following the same idea as the proof for the key switch (proof 9), we can isolate the noise and compute his variance. We obtain:

$$\begin{aligned}
&\mathsf{Var}\left(M_2 \left(E + \bar{B}_{\mathsf{in},i} - \sum_{i=0}^{k_{\mathsf{in}}-1} \bar{A}_{\mathsf{in},i} \cdot S_{\mathsf{in},i}\right) - \sum_{i=0}^{k_{\mathsf{in}}} \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},i,j} \cdot E_{i,j}\right) \\
&= ||M_2||_2^2 \cdot \mathsf{Var}\left(E + \bar{B}_{\mathsf{in},i} - \sum_{i=0}^{k_{\mathsf{in}}-1} \bar{A}_{\mathsf{in},i} \cdot S_{\mathsf{in},i}\right) + \mathsf{Var}\left(\sum_{i=0}^{k_{\mathsf{in}}} \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},i,j} \cdot E_{i,j}\right)
\end{aligned}$$

where

$$\begin{aligned}
&\mathsf{Var}\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)\\
&=\sigma_{\mathsf{in}}^2+\left(\frac{q^2}{12\beta^{2\ell}}-\frac{1}{12}\right)+\phi_{\mathsf{in}}\left(\frac{q^2}{12\beta^{2\ell}}-\frac{1}{12}\right)\left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)+\mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)+\frac{\phi_{\mathsf{in}}}{4}\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\\
&=\sigma_{\mathsf{in}}^2+\left(\frac{q^2-\beta^{2\ell}}{12\beta^{2\ell}}\right)\left(1+\phi_{\mathsf{in}}\left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)+\mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)\right)+\frac{\phi_{\mathsf{in}}}{4}\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)
\end{aligned}$$

and

$$\mathsf{Var}\left(\sum_{i=0}^{k_{\mathsf{in}}}\sum_{j=0}^{\ell-1}\tilde{A}_{\mathsf{in},i,j}\cdot E_{i,j}\right)=\ell(k_{\mathsf{in}}+1)N\sigma_{\mathsf{KSK}}^2\frac{\beta^2+2}{12}$$

□

Proof 11 (Noise Eternal Product in Bootstrapping) *The Theorem 2 gave us the following noise for an external product:*

$$\begin{aligned}
&\mathsf{Var}\left(M_2\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)-\sum_{i=0}^{k_{\mathsf{in}}}\sum_{j=0}^{\ell-1}\tilde{A}_{\mathsf{in},i,j}\cdot E_{i,j}\right)\\
&=||M_2||_2^2\cdot\mathsf{Var}\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)+\mathsf{Var}\left(\sum_{i=0}^{k_{\mathsf{in}}}\sum_{j=0}^{\ell-1}\tilde{A}_{\mathsf{in},i,j}\cdot E_{i,j}\right)
\end{aligned}$$

where

$$\begin{aligned}
&\mathsf{Var}\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)\\
&=\sigma_{\mathsf{in}}^2+\left(\frac{q^2}{12\beta^{2\ell}}-\frac{1}{12}\right)+\phi_{\mathsf{in}}\left(\frac{q^2}{12\beta^{2\ell}}-\frac{1}{12}\right)\left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)+\mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)+\frac{\phi_{\mathsf{in}}}{4}\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\\
&=\sigma_{\mathsf{in}}^2+\left(\frac{q^2-\beta^{2\ell}}{12\beta^{2\ell}}\right)\left(1+\phi_{\mathsf{in}}\left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)+\mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)\right)+\frac{\phi_{\mathsf{in}}}{4}\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)
\end{aligned}$$

and

$$\mathsf{Var}\left(\sum_{i=0}^{k_{\mathsf{in}}}\sum_{j=0}^{\ell-1}\tilde{A}_{\mathsf{in},i,j}\cdot E_{i,j}\right)=\ell(k_{\mathsf{in}}+1)N\sigma_{\mathsf{in}}^2\frac{\beta^2+2}{12}$$

In TFHE, we use binary key to perform the bootstrap. So we have $M\in\{0,1\}$ which represent a bit of the binary key. $\mathsf{Var}(M_2)=\frac{1}{4}$ *and* $\mathbb{E}(M_2)=\frac{1}{2}$. *Let focus on the part with the message:*

$$\begin{aligned}
&\mathsf{Var}\left(M_2\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)\right)\\
&\quad=\left(\mathsf{Var}(M_2)+\mathbb{E}^2(M_2)\right)\mathsf{Var}\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)\\
&\qquad+\mathsf{Var}\left(M_2\right)\mathbb{E}^2\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)\\
&\quad=\frac{1}{2}\mathsf{Var}\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)+\frac{1}{4}\mathbb{E}^2\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)
\end{aligned}$$

We have:

$$\begin{aligned}
\mathbb{E}^2\left(E+\bar{B}_{\mathsf{in},i}-\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)&=\left(\mathbb{E}(E)+\mathbb{E}(\bar{B}_{\mathsf{in},i})-\mathbb{E}\left(\sum_{i=0}^{k_{\mathsf{in}}-1}\bar{A}_{\mathsf{in},i}\cdot S_{\mathsf{in},i}\right)\right)^2\\
&=\left(0-\frac{1}{2}-\phi_{\mathsf{in}}\mathbb{E}\left(\bar{a}_{\mathsf{in},i}\right)\mathbb{E}\left(s_{\mathsf{in},i}\right)\right)^2\\
&=\frac{1}{4}\left(-1+\phi_{\mathsf{in}}\mathbb{E}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)^2
\end{aligned}$$

Finally, for each coefficient after one external product in the bootstrapping, we obtain the following formula for the noise variance:

$$\begin{aligned}
\frac{\sigma_{\mathsf{in}}^2}{2}+\left(\frac{q^2-\beta^{2\ell}}{24\beta^{2\ell}}\right)\left(1+\phi_{\mathsf{in}}\left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)+\mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right)\right)+\frac{\phi_{\mathsf{in}}}{8}\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\\
+\frac{1}{16}\left(-1+\phi_{\mathsf{in}}\mathbb{E}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)^2\right)+\ell(k_{\mathsf{in}}+1)N\sigma_{\mathsf{in}}^2\frac{\beta^2+2}{12}
\end{aligned}$$

□

Proof 12 (Theorem 8) *The proof of this theorem follows the same footprint as the other key switching proofs presented in this paper (e.g., Theorem 2). We generalize the proof of this theorem to the GLWE case: the LWE result presented in the theorem follows by taking $k_0 = n_0$, $k_1 = n_1$ and $N = 1$.*

We consider two shared randomness GLWE secret keys $\boldsymbol{S}^{(0)} \prec \boldsymbol{S}^{(1)}$ with $\boldsymbol{S}^{(0)} = \left(S_0^{(0)},\ldots,S_{k_0-1}^{(0)}\right) \in \mathfrak{R}_{q,N}^{k_0}$, $\boldsymbol{S}^{(1)} = \left(S_0^{(1)},\ldots,S_{k_1-1}^{(1)}\right) \in \mathfrak{R}_{q,N}^{k_1}$, $1 < k_0 < k_1$ and $S_i^{(1)} = S_i^{(0)}$ for all $0 \leq i < k_0$. We take in input:

- *A GLWE ciphertext: $\mathsf{CT}_{\mathsf{in}} = (\boldsymbol{A}_{\mathsf{in}}, B_{\mathsf{in}}) \in \mathrm{GLWE}_{\boldsymbol{S}^{(1)}}(\Delta M) \subseteq \mathfrak{R}_{q,N}^{k_1+1}$, where $B_{\mathsf{in}} = \sum_{i=0}^{k_1-1} A_{\mathsf{in},i}\cdot S_i^{(1)} + \Delta M + E_{\mathsf{in}}$, $A_{\mathsf{in},i} = \sum_{j=0}^{N-1} a_{i,j}\cdot X^j \hookleftarrow \mathfrak{U}\left(\mathfrak{R}_{q,N}\right)$ for all $i \in [\![0,k[\![$ and $E_{\mathsf{in}} = \sum_{j=0}^{N-1} e_j \cdot X^j$, and $e_j \hookleftarrow \mathfrak{N}_{\sigma_1^2}$ for all $j \in [\![0,N-1[\![$.*

- *The key switching key: $\mathsf{KSK} = (\mathsf{KSK}_0,\ldots,\mathsf{KSK}_{k_1-k_0-1})$, where $\mathsf{KSK}_i \in \mathsf{GLEV}_{\boldsymbol{S}^{(0)}}^{(\beta,\ell)}\left(S_{k_0+i}^{(1)}\right) = \left(\mathrm{GLWE}_{\boldsymbol{S}^{(0)}}\left(\frac{q}{\beta}S_{k_0+i}^{(1)}\right),\cdots,\mathrm{GLWE}_{\boldsymbol{S}^{(0)}}\left(\frac{q}{\beta^\ell}S_{k_0+i}^{(1)}\right)\right)$ for all*

$0 \leq i < k_1$. We note by $\mathsf{KSK}_{i,j} = (\boldsymbol{A_{i,j}}, B_{i,j}) \in \mathrm{GLWE}_{\boldsymbol{S}^{(0)}}\left(\frac{q}{\beta^{j+1}} S^{(1)}_{k_0+i}\right)$, for all $0 \leq i < k_1$ and for all $0 \leq j < \ell$, where $B_{i,j} = \sum_{\tau=0}^{k_0-1} A_{i,j,\tau} \cdot \boldsymbol{S}^{(0)}_{\tau} + \frac{q}{\beta^{j+1}} S^{(1)}_{k_0+i} + E_{\mathsf{ksk},i,j}$, and $E_{\mathsf{ksk},i,j} = \sum_{\tau=0}^{N-1} e_{\mathsf{ksk},i,j,\tau} \cdot X^{\tau}$ and $e_{\mathsf{ksk},i,j,\tau} \hookleftarrow \mathcal{N}_{\sigma^2_{\mathsf{ksk}}}$.

The output of this algorithm is computed as:

$$\left(A_{\mathsf{in},0}, \ldots, A_{\mathsf{in},k_0-1}, B_{\mathsf{in}}\right) - \sum_{i=0}^{k_1-k_0-1} \mathbf{Dec}^{(\beta,\ell)}\left(A_{\mathsf{in},k_0+i}\right) \cdot \mathsf{KSK}_i \in \mathrm{GLWE}_{\boldsymbol{S}^{(0)}}\left(\Delta M\right) \subseteq \mathfrak{R}_{q,N}^{k_0+1}$$

By definition, in the decomposition described in Supplementary Material B, we have that $\mathbf{Dec}^{(\beta,\ell)}\left(A_{\mathsf{in},i}\right) = \left(\widetilde{A}_{\mathsf{in},i,0}, \cdots, \widetilde{A}_{\mathsf{in},i,\ell-1}\right)$ such that $\widetilde{A}_{\mathsf{in},i} = \sum_{j=0}^{\ell-1} \frac{q}{\beta^{j+1}} \widetilde{A}_{\mathsf{in},i,j}$, for all $k_0 \leq i < k_1$.
Let define $\bar{A}_{\mathsf{in},i} = A_{\mathsf{in},i} - \widetilde{A}_{\mathsf{in},i}$, $|\bar{a}_{i,\tau}| = |a_{i,\tau} - \widetilde{a}_{i,\tau}| < \frac{q}{2\beta^{\ell}}$, $\bar{a}_{i,\tau} \in \left[\!\left[\frac{-q}{2\beta^{\ell}}, \frac{q}{2\beta^{\ell}} \right[\!\right[$ for all $0 \leq \tau < N$. So we have that their expectations and variances are respectively $\mathbb{E}\left(\bar{a}_{i,\tau}\right) = -\frac{1}{2}$, $\mathsf{Var}\left(\bar{a}_{i,\tau}\right) = \frac{q^2}{12\beta^{2\ell}} - \frac{1}{12}$, $\mathbb{E}\left(\widetilde{a}_{i,\tau}\right) = -\frac{1}{2}$ and $\mathsf{Var}\left(\widetilde{a}_{i,\tau}\right) = \frac{\beta^2-1}{12}$.
Now, we can decrypt:

$$\begin{aligned}
&\left\langle \left(A_{\mathsf{in},0}, \ldots, A_{\mathsf{in},k_0-1}, B_{\mathsf{in}}\right) - \sum_{i=0}^{k_1-k_0-1} \mathbf{Dec}^{(\beta,\ell)}\left(A_{\mathsf{in},k_0+i}\right) \cdot \mathsf{KSK}_i, \left(-\boldsymbol{S}^{(0)}, 1\right) \right\rangle \\
&= B_{\mathsf{in}} - \sum_{i=0}^{k_0-1} A_{\mathsf{in},i} \cdot S^{(0)}_i - \sum_{i=0}^{k_1-k_0-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},k_0+i,j} \cdot \left\langle \mathsf{KSK}_{i,j}, \left(-\boldsymbol{S}^{(0)}, 1\right) \right\rangle \\
&= B_{\mathsf{in}} - \sum_{i=0}^{k_0-1} A_{\mathsf{in},i} \cdot S^{(0)}_i - \sum_{i=0}^{k_1-k_0-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},k_0+i,j} \cdot \left(\frac{q}{\beta^{j+1}} S^{(1)}_{k_0+i} + E_{\mathsf{ksk},i,j} \right) \\
&= B_{\mathsf{in}} - \sum_{i=0}^{k_0-1} A_{\mathsf{in},i} \cdot S^{(0)}_i - \sum_{i=0}^{k_1-k_0-1} \widetilde{A}_{\mathsf{in},k_0+i} \cdot S^{(1)}_{k_0+i} - \sum_{i=0}^{k_1-k_0-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},k_0+i,j} \cdot E_{\mathsf{ksk},i,j} \\
&= B_{\mathsf{in}} - \sum_{i=0}^{k_0-1} A_{\mathsf{in},i} \cdot S^{(0)}_i - \sum_{i=0}^{k_1-k_0-1} \left(A_{\mathsf{in},k_0+i} - \bar{A}_{\mathsf{in},k_0+i}\right) \cdot S^{(1)}_{k_0+i} - \sum_{i=0}^{k_1-k_0-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},k_0+i,j} \cdot E_{\mathsf{ksk},i,j}
\end{aligned}$$

Since $S^{(0)}_i = S^{(1)}_i$ for all $0 \leq i < k_0$, the equation becomes:

$$\begin{aligned}
&= B_{\mathsf{in}} - \sum_{i=0}^{k_0-1} A_{\mathsf{in},i} \cdot S^{(1)}_i - \sum_{i=0}^{k_1-k_0-1} \left(A_{\mathsf{in},k_0+i} - \bar{A}_{\mathsf{in},k_0+i}\right) \cdot S^{(1)}_{k_0+i} - \sum_{i=0}^{k_1-k_0-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},k_0+i,j} \cdot E_{\mathsf{ksk},i,j} \\
&= \Delta M + E_{\mathsf{in}} + \underbrace{\sum_{i=0}^{k_1-k_0-1} \bar{A}_{\mathsf{in},k_0+i} \cdot S^{(1)}_{k_0+i}}_{(I)} - \underbrace{\sum_{i=0}^{k_1-k_0-1} \sum_{j=0}^{\ell-1} \widetilde{A}_{\mathsf{in},k_0+i,j} \cdot E_{\mathsf{ksk},i,j}}_{(II)}
\end{aligned}$$

The w^{th} coefficient of part (I) is equal to:

$$\sum_{i=k_0}^{k_1-1} \left(\sum_{\tau=0}^{w} \bar{a}_{\mathsf{in},i,w-\tau} \cdot s^{(1)}_{i,\tau} - \sum_{\tau=w+1}^{N-1} \bar{a}_{\mathsf{in},i,N+w-\tau} \cdot s^{(1)}_{i,\tau} \right)$$

The w^{th} coefficient of part (II) is equal to:

$$\sum_{i=0}^{k_1-k_0-1}\sum_{j=0}^{\ell-1}\left(\sum_{\tau=0}^{w}\widetilde{a}_{\mathsf{in},k_0+i,j,w-\tau}\cdot e_{\mathsf{ksk},i,j,\tau}-\sum_{\tau=w+1}^{N-1}\widetilde{a}_{\mathsf{in},k_0+i,j,N+w-\tau}\cdot e_{\mathsf{ksk},i,j,\tau}\right)$$

We can now isolate the output error for the w^{th} coefficient and remove the message coefficient. We obtain that the output error is:

$$\begin{aligned}e'_w = e_{\mathsf{in},w} &+ \sum_{i=k_0}^{k_1-1}\left(\sum_{\tau=0}^{w}\bar{a}_{\mathsf{in},i,w-\tau}\cdot s^{(1)}_{i,\tau}-\sum_{\tau=w+1}^{N-1}\bar{a}_{\mathsf{in},i,N+w-\tau}\cdot s^{(1)}_{i,\tau}\right)\\ &-\sum_{i=0}^{k_1-k_0-1}\sum_{j=0}^{\ell-1}\left(\sum_{\tau=0}^{w}\widetilde{a}_{\mathsf{in},k_0+i,j,w-\tau}\cdot e_{\mathsf{ksk},i,j,\tau}-\sum_{\tau=w+1}^{N-1}\widetilde{a}_{\mathsf{in},k_0+i,j,N+w-\tau}\cdot e_{\mathsf{ksk},i,j,\tau}\right)\end{aligned}$$

So the variance is:

$$\begin{aligned}\mathsf{Var}(e'_w) &= \mathsf{Var}(e_{\mathsf{in},w}) + (k_1-k_0)N\mathsf{Var}\left(\bar{a}_{\mathsf{in},i,\cdot}.s^{(1)}_{i,\cdot}\right) + (k_1-k_0)\ell N\mathsf{Var}\left(\widetilde{a}_{\mathsf{in},i,j,\cdot}\cdot e_{\mathsf{ksk},i,j,\cdot}\right)\\ &= \sigma^2_{\mathsf{in}} + (k_1-k_0)N\left(\frac{q^2-\beta^{2\ell}}{12\beta^{2\ell}}\right)\left(\mathsf{Var}\left(\boldsymbol{S}^{(1)}\right)+\mathbb{E}^2\left(\boldsymbol{S}^{(1)}\right)\right)\\ &+\frac{(k_1-k_0)N}{4}\mathsf{Var}\left(\boldsymbol{S}^{(1)}\right) + (k_1-k_0)\ell N\frac{\beta^2+2}{12}\sigma^2_{\mathsf{KSK}}.\end{aligned}$$

□

Proof 13 (Theorem 11) *Lets consider two shared and partial secret keys such that $\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}} \prec \boldsymbol{S}^{[\phi_{\mathsf{in}}]}_{\mathsf{in}}$. We have $\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}} = (S_{\mathsf{out},0},\cdots,S_{\mathsf{out},k_{\mathsf{out}}-1})$, where $S_{\mathsf{out},k_{\mathsf{out}}-1} = \sum_{i=0}^{\phi_{\mathsf{out}}-(k_{\mathsf{out}}-1)N-1} s_{\mathsf{out},k_{\mathsf{out}}-1,i}X^i$ we call $S_{\mathsf{out},k_{\mathsf{out}}-1} : \underline{S}$.*

We have $\boldsymbol{S}^{[\phi_{\mathsf{in}}]}_{\mathsf{in}} = (S_{\mathsf{in},0},\cdots,S_{\mathsf{in},k_{\mathsf{in}}-1})$ such that for all $j\in[\![0,k_{\mathsf{out}}-1[\![$, $S_{\mathsf{out},j} = S_{\mathsf{in},j}$ and $S_{\mathsf{in},k_{\mathsf{out}}-1} = \underline{S}+\bar{S}$ where $\bar{S} = \sum_{j=\phi_{\mathsf{out}}-(k_{\mathsf{out}}-1)N}^{N-1} s_{\mathsf{in},k_{\mathsf{out}-1},j}X^j$.

The inputs of a GLWE key switching with partial & shared randomness keys (Algorithm 11) are:

- *The input GLWE ciphertext:* $\mathsf{CT}_{\mathsf{in}} = (\boldsymbol{A}_{\mathsf{in}},B_{\mathsf{in}}) \in \mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{in}}]}_{\mathsf{in}}}(\Delta\cdot M) \subseteq \mathfrak{R}^{k_{\mathsf{in}}+1}_{q,N}$, *where* $B_{\mathsf{in}} = \sum_{i=0}^{k_{\mathsf{in}}-1}A_{\mathsf{in},i}\cdot S_{\mathsf{in},i}+\Delta\cdot M+E_{\mathsf{in}}$, $A_{\mathsf{in},i} = \sum_{j=0}^{k-1}a_{i,j}\cdot X^j \hookleftarrow \mathcal{U}(\mathfrak{R}_{q,N})$ *for all* $i\in[\![0,k[\![$ *and* $E_{\mathsf{in}} = \sum_{j=0}^{k-1}e_j\cdot X^j$, *and* $e_j \hookleftarrow \mathcal{N}_{\sigma^2_{\mathsf{in}}}$ *for all* $j\in[\![0,N-1[\![$.

- *The key switch key:* $\mathsf{KSK} = (\mathsf{KSK}_{k_{\mathsf{out}-1}},\mathsf{KSK}_{k_{\mathsf{out}}}\cdots,\mathsf{KSK}_{k_{\mathsf{in}}-1})$, *where* $\mathsf{KSK}_i \in \mathsf{GLEV}^{(\beta,\ell)}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}(S_{\mathsf{in},i}) = \left(\mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(\frac{q}{\beta}S_{\mathsf{in},i}\right),\cdots,\mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(\frac{q}{\beta^\ell}S_{\mathsf{in},i}\right)\right)$ *for all* $k_{\mathsf{out}} \leq i < k_{\mathsf{in}}$, *and* $\mathsf{KSK}_{k_{\mathsf{out}}-1} \in \mathsf{GLEV}^{(\beta,\ell)}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}(\bar{S}) = \left(\mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(\frac{q}{\beta}\bar{S}\right),\cdots,\mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(\frac{q}{\beta^\ell}\bar{S}\right)\right)$

  We note by $\mathsf{KSK}_{i,j} = (\boldsymbol{A}_{i,j},B_{i,j}) \in \mathrm{GLWE}_{\boldsymbol{S}^{[\phi_{\mathsf{out}}]}_{\mathsf{out}}}\left(\frac{q}{\beta^{j+1}}S_{\mathsf{in},i}\right)$, *for all* $k_{\mathsf{out}} \leq i < k_{\mathsf{in}}$ *for all* $0\leq j<\ell$, *where* $B_{i,j} = \sum_{\tau=0}^{k_{\mathsf{out}}-1}A_{i,j,\tau}\cdot S_{\mathsf{out},\tau}+\frac{q}{\beta^{j+1}}S_{\mathsf{in},i}+E_{\mathsf{ksk},i,j}$, *and* $E_{\mathsf{ksk},i,j} = \sum_{\tau=0}^{N-1}e_{\mathsf{ksk},i,j,\tau}\cdot X^\tau$ *and* $e_{\mathsf{ksk},i,j,m} \hookleftarrow \mathcal{N}_{\sigma^2_{\mathsf{ksk}}}$.

We note $\mathsf{KSK}_{k_{\mathsf{out}}-1,j} = (\boldsymbol{A}_{k_{\mathsf{out}}-1,j}, B_{k_{\mathsf{out}}-1,j}) \in \mathrm{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta^{j+1}}\bar{S}\right)$ *for all* $0 \leq j < \ell$*, where* $B_{k_{\mathsf{out}}-1,j} = \sum_{\tau=0}^{k_{\mathsf{out}}-1} A_{k_{\mathsf{out}}-1,j,\tau} \cdot S_{\mathsf{out},\tau} + \frac{q}{\beta^{j+1}}\bar{S} + E_{\mathsf{ksk},k_{\mathsf{out}}-1,j}$*, and* $E_{\mathsf{ksk},k_{\mathsf{out}}-1,j} = \sum_{\tau=0}^{N-1} e_{\mathsf{ksk},k_{\mathsf{out}}-1,j,\tau} \cdot X^{\tau}$ *and* $e_{\mathsf{ksk},k_{\mathsf{out}}-1,j,m} \hookleftarrow \mathcal{N}_{\sigma^2_{\mathsf{ksk}}}$.

The output of this algorithm is: $\mathsf{CT}_{\mathsf{out}} = (\boldsymbol{A}_{\mathsf{out}}, B_{\mathsf{out}}) \in \mathrm{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}(\Delta \cdot M) \subseteq \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$.

By definition, for any polynomial $A_{\mathsf{in},i}$*, we have the decomposition (described in Supplementary Material B),* $\mathbf{Dec}^{(\beta,\ell)}(A_{\mathsf{in},i}) = \left(\tilde{A}_{\mathsf{in},i,1}, \cdots, \tilde{A}_{\mathsf{in},i,\ell}\right)$ *such that* $\tilde{A}_{\mathsf{in},i} = \sum_{j=0}^{\ell-1} \frac{q}{\beta^{j+1}} \tilde{A}_{\mathsf{in},i,j}$*. Now, we can decrypt:*

$$\begin{aligned}
B_{\mathsf{out}} - \left\langle \boldsymbol{A}_{\mathsf{out}}, \boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \right\rangle &= \left\langle (\boldsymbol{A}_{\mathsf{out}}, B_{\mathsf{out}}), \left(-\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}, 1\right) \right\rangle \\
&= \Bigg\langle (A_{\mathsf{in},0}, \cdots, A_{\mathsf{in},k_{\mathsf{out}}-1}, 0 \cdots, 0, B_{\mathsf{in}}) - \mathbf{Dec}^{(\beta,\ell)}(A_{\mathsf{in},k_{\mathsf{out}}-1})\,\mathsf{KSK}_{k_{\mathsf{out}}-1} \\
&\quad - \sum_{i=k_{\mathsf{out}}}^{k_{\mathsf{in}}-1} \mathbf{Dec}^{(\beta,\ell)}(A_{\mathsf{in},i})\,\mathsf{KSK}_i, \left(-\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}, 1\right) \Bigg\rangle \\
&= B_{\mathsf{in}} - \sum_{i=0}^{k_{\mathsf{out}}-1} A_{\mathsf{in},i} S_{\mathsf{out},i} - \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},k_{\mathsf{out}}-1,j} \left\langle \mathsf{KSK}_{k_{\mathsf{out}}-1,j}, \left(-\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}, 1\right) \right\rangle \\
&\quad - \sum_{i=k_{\mathsf{out}}}^{k_{\mathsf{in}}-1} \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},i,j} \left\langle \mathsf{KSK}_{i,j}, \left(-\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}, 1\right) \right\rangle \\
&= B_{\mathsf{in}} - \sum_{i=0}^{k_{\mathsf{out}}-2} A_{\mathsf{in},i} S_{\mathsf{in},i} - A_{\mathsf{in},k_{\mathsf{out}}-1} \underline{S} - \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},k_{\mathsf{out}}-1,j} \left(\frac{q}{\beta^{j+1}} \bar{S} + E_{\mathsf{ksk},k_{\mathsf{out}}-1,j} \right) \\
&\quad - \sum_{i=k_{\mathsf{out}}}^{k_{\mathsf{in}}-1} \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},i,j} \left(\frac{q}{\beta^{j+1}} S_{\mathsf{in},i} + E_{\mathsf{ksk},i,j} \right) \\
&= \underbrace{B_{\mathsf{in}} - \sum_{i=0}^{k_{\mathsf{out}}-1} A_{\mathsf{in},i} S_{\mathsf{in},i} - A_{\mathsf{in},k_{\mathsf{out}}-1} \underline{S}}_{(I)} \underbrace{- \tilde{A}_{\mathsf{in},k_{\mathsf{out}}-1} \bar{S} - \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},k_{\mathsf{out}}-1,j} \cdot E_{\mathsf{ksk},k_{\mathsf{out}}-1,j}}_{(II)} \\
&\quad \underbrace{- \sum_{i=k_{\mathsf{out}}}^{k_{\mathsf{in}}-1} \tilde{A}_{\mathsf{in},i} S_{\mathsf{in},i} - \sum_{i=k_{\mathsf{out}}}^{k_{\mathsf{in}}-1} \sum_{j=0}^{\ell-1} \tilde{A}_{\mathsf{in},i,j} \cdot E_{\mathsf{ksk},i,j}}_{(III)}
\end{aligned}$$

After decrypting, we can split the previous result in three distinct part and analyze the noise provide by each of them. The first part of the result (term (I)*) is only composed of the noise present in the* B_{in}*.*

The second part of the result (term (II)*) can be seen as a key switching with partial key (Algorithm 4) from* $\bar{S}$ *to* S_{out}*. The proof of noise add by this part follows the proof of Theorem 2.*

As for the second part of the result, the third part of the result (term (III)*) can be seen as a key switching with partial key (Algorithm 4) from* $(S_{\mathsf{in},k_{\mathsf{out}}}, \cdots, S_{\mathsf{in},k_{\mathsf{in}}-1})$ *to* S_{out}*. The proof of noise add by this part follows as well the proof of Theorem 2.*

By adding this different noises, we will obtain $\mathsf{Var}(e_{\mathsf{out}}) = \mathsf{Var}(I) + \mathsf{Var}(II) + \mathsf{Var}(III)$ *where :*

$$\begin{aligned}
\mathsf{Var}(I) &= \sigma_{\mathsf{in}}^2 \\
\mathsf{Var}(II) &= (Nk_{\mathsf{out}} - \phi_{\mathsf{out}}) \left(\frac{q^2 - \beta^{2\ell}}{12\beta^{2\ell}}\right) \left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right) \\
&\quad + \frac{Nk_{\mathsf{out}} - \phi_{\mathsf{out}}}{4} \mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \ell N \sigma_{\mathsf{ksk}}^2 \frac{\beta^2 + 2}{12} \\
\mathsf{Var}(III) &= (\phi_{\mathsf{in}} - Nk_{\mathsf{out}}) \left(\frac{q^2 - \beta^{2\ell}}{12\beta^{2\ell}}\right) \left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right) \\
&\quad + \frac{\phi_{\mathsf{in}} - Nk_{\mathsf{out}}}{4} \mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \ell(k_{\mathsf{in}} - k_{\mathsf{out}} - 1) N \sigma_{\mathsf{ksk}}^2 \frac{\beta^2 + 2}{12}
\end{aligned}$$

To conclude we have :

$$\begin{aligned}
\mathsf{Var}(e_{\mathsf{out}}) = \sigma_{\mathsf{in}}^2 + (\phi_{\mathsf{in}} - \phi_{\mathsf{out}}) \left(\frac{q^2 - \beta^{2\ell}}{12\beta^{2\ell}}\right) \left(\mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \mathbb{E}^2\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right)\right) \\
+ \frac{\phi_{\mathsf{in}} - \phi_{\mathsf{out}}}{4} \mathsf{Var}\left(\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}\right) + \ell(k_{\mathsf{in}} - k_{\mathsf{out}}) N \sigma_{\mathsf{ksk}}^2 \frac{\beta^2 + 2}{12}
\end{aligned}$$

□

B Algorithms From Literature

In this section we recall a few useful algorithms from the literature such as the GLWE-to-GLWE key switch with and without a secret product, and the decomposition algorithm.

Decomposition Algorithms. Let $\beta \in \mathbb{N}^*$ be a decomposition base and $\ell \in \mathbb{N}^*$ be a decomposition level. The decomposition algorithm with respect to β and ℓ is noted $\mathbf{Dec}^{(\beta,\ell)}$: it takes as input an integer $x \in \mathbb{Z}_q$ and outputs a decomposition vector of integers $(x_1, \cdots, x_\ell) \in \mathbb{Z}_q^\ell$ such that:

$$\left\langle \mathbf{Dec}^{(\beta,\ell)}(x), \left(\frac{q}{\beta} \cdots \frac{q}{\beta^\ell}\right) \right\rangle = \left\lfloor x \cdot \frac{\beta^\ell}{q} \right\rceil \cdot \frac{q}{\beta^\ell} \in \mathbb{Z}_q.$$

Usually, the decomposition is done starting from the most significant bits. When applying the decomposition on a vector of integers, the result is a vector of decomposition vectors of integers.

Note that it is possible to decompose an integer polynomials $X \in \mathfrak{R}_q$ with this algorithm i.e.,

$$\left\langle \mathbf{Dec}^{(\beta,\ell)}(X), \left(\frac{q}{\beta} \cdots \frac{q}{\beta^\ell}\right) \right\rangle = \left\lfloor X \cdot \frac{\beta^\ell}{q} \right\rceil \cdot \frac{q}{\beta^\ell} \in \mathfrak{R}_q$$

After applying this kind of decomposition on a vector of polynomials, the output is a vector of vector of polynomials.

```
Algorithm 4: $\mathsf{CT}_{\mathsf{out}} \leftarrow \mathsf{GlweKeySwitch}(\mathsf{CT}_{\mathsf{in}}, \mathsf{KSK})$
Context: $\begin{cases} (k_{\mathsf{in}}-1)N < \phi_{\mathsf{in}} \leq k_{\mathsf{in}}N \text{ and } (k_{\mathsf{out}}-1)N < \phi_{\mathsf{out}} \leq k_{\mathsf{out}}N \\ \boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}} : \text{the input partial secret key (Definition 6)} \\ \boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} = (S_{\mathsf{in},0}, \cdots, S_{\mathsf{in},k_{\mathsf{in}}-1}) \\ \boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}} : \text{the output partial secret key (Definition 6)} \\ \ell \in \mathbb{N} : \text{ the number of levels in the decomposition} \\ \beta \in \mathbb{N} : \text{ the base in the decomposition} \end{cases}$
Input: $\begin{cases} \mathsf{CT}_{\mathsf{in}} = (A_0, \cdots, A_{k_{\mathsf{in}}-1}, B) \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}}(P) \subseteq \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}, \text{ with } P \in \mathfrak{R}_{q,N} \\ \mathsf{KSK} = \{\mathsf{KSK}_i = \{\mathsf{KSK}_{i,j}\}_{0 \leq j \leq \ell-1}\}_{0 \leq i \leq k_{\mathsf{in}}-1}, \text{ with} \\ \quad \mathsf{KSK}_{i,j} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta^j} \cdot S_{\mathsf{in},i}\right), \text{ for } 0 \leq i \leq k_{\mathsf{in}}-1 \text{ and } 0 \leq j \leq \ell-1 \end{cases}$
Output: $\mathsf{CT}_{\mathsf{out}} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}(P)$

/* Keep the $B$ part */
Set $\mathsf{CT}_{\mathsf{out}} := (0, \cdots, 0, B) \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$
for $i \in [\![0; k_{\mathsf{in}}-1]\!]$ do
    /* Decompose the mask */
    Update $\mathsf{CT}_{\mathsf{out}} = \mathsf{CT}_{\mathsf{out}} - \left\langle \boldsymbol{K}_i, \mathbf{Dec}^{(\beta,\ell)}(A_i) \right\rangle$
return $\mathsf{CT}_{\mathsf{out}}$
```

```
Algorithm 5: $\mathsf{CT}_{\mathsf{out}} \leftarrow \mathsf{SecretProductGlweKeySwitch}(\mathsf{CT}_{\mathsf{in}}, \mathsf{KSK})$
Context: $\begin{cases} \boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}} : \text{the input partial secret key (Definition 6)} \\ \boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} = (S_{\mathsf{in},0}, \cdots, S_{\mathsf{in},k_{\mathsf{in}}-1}) \\ \boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}} : \text{the output partial secret key (Definition 6)} \\ (k_{\mathsf{in}}-1)N < \phi_{\mathsf{in}} \leq k_{\mathsf{in}}N \text{ and } (k_{\mathsf{out}}-1)N < \phi_{\mathsf{out}} \leq k_{\mathsf{out}}N \\ Q \in \mathfrak{R}_{q,N} \\ \mathsf{CT}_{\mathsf{in}} = (A_0, \cdots, A_{k_{\mathsf{in}}-1}, B) \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1} \\ \mathsf{CT}_{i,j} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta^j} \cdot Q \cdot S_{\mathsf{in},i}\right), \text{ for } 0 \leq i \leq k_{\mathsf{in}}-1 \text{ and } 0 \leq j \leq \ell-1 \\ \mathsf{CT}_{k_{\mathsf{in}},j} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta^j} \cdot Q\right), \text{ for } 0 \leq j \leq \ell-1 \\ \ell \in \mathbb{N} : \text{ the number of levels in the decomposition} \\ \beta \in \mathbb{N} : \text{ the base in the decomposition} \end{cases}$
Input: $\begin{cases} \mathsf{CT}_{\mathsf{in}} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}}(P), \text{ with } P \in \mathfrak{R}_{q,N} \\ \mathsf{KSK} = \{\boldsymbol{K}_i = (\mathsf{CT}_{i,0}, \cdots, \mathsf{CT}_{i,\ell-1})\}_{0 \leq i \leq k_{\mathsf{in}}} \end{cases}$
Output: $\mathsf{CT}_{\mathsf{out}} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}(Q \cdot P)$

/* Decompose the $B$ part */
Set $\mathsf{CT}_{\mathsf{out}} = \left\langle \boldsymbol{K}_{k_{\mathsf{in}}}, \mathsf{Decomp}^{(\beta,\ell)}(B) \right\rangle$
for $i \in [\![0; k-1]\!]$ do
    /* Decompose the mask */
    Update $\mathsf{CT}_{\mathsf{out}} = \mathsf{CT}_{\mathsf{out}} - \left\langle \boldsymbol{K}_i, \mathsf{Decomp}^{(\beta,\ell)}(A_i) \right\rangle$
return $\mathsf{CT}_{\mathsf{out}}$
```

C Algorithms in Details

In this section we give details about some algorithms introduced in this paper, such as the different sample extraction algorithms (and its inverse) when dealing with partial GLWE secret keys, the enlarging/shrinking key switch when dealing with shared randomness secret keys, and the GLWE-to-GLWE key switch when dealing with both partial and shared randomness secret keys.

```
Algorithm 6: $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{ConstantSampleExtract}(\mathsf{CT}_{\mathsf{in}})$

Context: $\begin{cases} \boldsymbol{S}^{[\phi]} \in \mathfrak{R}_{q,N}^{k} : \text{a partial secret key (Definition 6)} \\ (k-1)N+1 \leq \phi \leq kN : \text{filling amount of the partial secret key} \\ \bar{\boldsymbol{s}} \in \mathbb{Z}^{\phi} : \text{the flattened version of } \boldsymbol{S}^{[\phi]} \text{ (Definition 7)} \\ P := \sum_{i=0}^{N-1} p_i X^i \in \mathfrak{R}_{q,N} \\ \mathsf{CT}_{\mathsf{in}} = \left(\sum_{i=0}^{N-1} a_{0,i}X^i, \cdots, \sum_{i=0}^{N-1} a_{k-1,i}X^i, \sum_{i=0}^{N-1} b_i X^i\right) \in \mathfrak{R}_{q,N}^{k+1} \end{cases}$
Input: $\mathsf{CT}_{\mathsf{in}} \in \mathsf{GLWE}_{\boldsymbol{S}^{[\phi]}}(P)$ : a GLWE encryption of the plaintext $P$
Output: $\mathsf{ct}_{\mathsf{out}} \in \mathsf{LWE}_{\bar{\boldsymbol{s}}}(p_0)$ : an LWE encryption of the plaintext $p_0$
for $i \in [\![0; \phi-1]\!]$ do
    set $\alpha := \left\lfloor \frac{i}{N} \right\rfloor$, $\beta := (N-i) \mod N$ and $\gamma := 1 - (\beta == 0)$
    set $a_{\mathsf{out},i} := (-1)^{\gamma} \cdot a_{\alpha,\beta}$
return $\mathsf{ct}_{\mathsf{out}} := (a_{\mathsf{out},0}, \cdots, a_{\mathsf{out},\phi-1}, b_0) \in \mathbb{Z}_q^{\phi+1}$
```

```
Algorithm 7: $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{SampleExtract}(\mathsf{CT}_{\mathsf{in}}, \alpha)$

Context: $\begin{cases} \boldsymbol{S}^{[\phi]} \in \mathfrak{R}_{q,N}^{k} : \text{a partial secret key (Definition 6)} \\ (k-1)N+1 \leq \phi \leq kN : \text{filling amount of the partial secret key} \\ \bar{\boldsymbol{s}} \in \mathbb{Z}^{\phi} : \text{the flattened version of } \boldsymbol{S}^{[\phi]} \text{ (Definition 7)} \\ P := \sum_{i=0}^{N-1} p_i X^i \in \mathfrak{R}_{q,N} \end{cases}$
Input: $\begin{cases} \mathsf{CT}_{\mathsf{in}} \in \mathsf{GLWE}_{\boldsymbol{S}^{[\phi]}}(P) : \text{a GLWE encryption of the plaintext } P \\ 0 \leq \alpha \leq N-1 : \text{the coefficient to extract} \end{cases}$
Output: $\mathsf{ct}_{\mathsf{out}} \in \mathsf{LWE}_{\bar{\boldsymbol{s}}}(p_\alpha)$ : an LWE encryption of the plaintext $p_\alpha$
/* Rotation of the GLWE ciphertext */
set $\mathsf{CT} := X^{-\alpha} \cdot \mathsf{CT}_{\mathsf{in}}$
/* Call to Algorithm 6 */
return $\mathsf{ct}_{\mathsf{out}} := \mathsf{ConstantSampleExtract}(\mathsf{CT})$
```

```
Algorithm 8: $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{EnlargingKeySwitch}(\mathsf{ct}_{\mathsf{in}})$

Context: $\begin{cases} \boldsymbol{s}_{\mathsf{in}} \in \mathbb{Z}_q^{n_{\mathsf{in}}} : \text{the input secret key} \\ \boldsymbol{s}_{\mathsf{out}} \in \mathbb{Z}_q^{n_{\mathsf{out}}} : \text{the output secret key} \\ \boldsymbol{s}_{\mathsf{in}} \prec \boldsymbol{s}_{\mathsf{out}} : \text{ shared randomness secret keys (Definition 9)} \\ p \in \mathbb{Z}_q \\ \mathsf{ct}_{\mathsf{in}} = (a_0, \cdots, a_{n_{\mathsf{in}}-1}, b) \in \mathbb{Z}_q^{n_{\mathsf{in}}+1} \end{cases}$
Input: $\mathsf{ct}_{\mathsf{in}} \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{in}}}(p)$
Output: $\mathsf{ct}_{\mathsf{out}} \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{out}}}(p)$

/* Pad with zeros between the mask and the $b$ part */
Set $\mathsf{ct}_{\mathsf{out}} := (a_0, \cdots, a_{n-1}, 0, \cdots, 0, b) \in \mathbb{Z}_q^{n_{\mathsf{out}}+1}$
return $\mathsf{ct}_{\mathsf{out}}$
```

```
Algorithm 9: $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{ShrinkingKeySwitch}(\mathsf{ct}_{\mathsf{in}}, \mathsf{KSK})$

Context: $\begin{cases} \boldsymbol{s}_{\mathsf{in}} = (s_0, \cdots, s_{n_{\mathsf{in}}-1}) \in \mathbb{Z}_q^{n_{\mathsf{in}}} : \text{the input secret key} \\ \boldsymbol{s}_{\mathsf{out}} \in \mathbb{Z}_q^{n_{\mathsf{out}}} : \text{the output secret key} \\ n_{\mathsf{out}} < n_{\mathsf{in}} \\ \boldsymbol{s}_{\mathsf{out}} \prec \boldsymbol{s}_{\mathsf{in}} : \text{ shared randomness secret keys (Definition 9)} \\ p \in \mathbb{Z}_q \\ \mathsf{ct}_{\mathsf{in}} = (a_0, \cdots, a_{n_{\mathsf{in}}-1}, b) \in \mathbb{Z}_q^{n_{\mathsf{in}}+1} \\ \mathsf{ct}_{i,j} \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{out}}}\left(\frac{q}{\beta^j} \cdot s_i\right), \text{ for } n_{\mathsf{out}} \leq i \leq n_{\mathsf{in}} - 1 \text{ and } 0 \leq j \leq \ell - 1 \\ \ell \in \mathbb{N} : \text{ the number of levels in the decomposition} \\ \beta \in \mathbb{N} : \text{ the base in the decomposition} \end{cases}$
Input: $\begin{cases} \mathsf{ct}_{\mathsf{in}} \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{in}}}(p) \\ \mathsf{KSK} = \{\boldsymbol{k}_i = (\mathsf{ct}_{i,0}, \cdots, \mathsf{ct}_{i,\ell-1})\}_{n_{\mathsf{out}} \leq i \leq n_{\mathsf{in}}-1} \end{cases}$
Output: $\mathsf{ct}_{\mathsf{out}} \in \mathsf{LWE}_{\boldsymbol{s}_{\mathsf{out}}}(p)$

/* Keep the beginning of the mask and the $B$ part */
Set $\mathsf{ct}_{\mathsf{out}} := (a_0, \cdots, a_{n_{\mathsf{out}}-1}, b) \in \mathbb{Z}_q^{n_{\mathsf{out}}+1}$
for $i \in [\![n_{\mathsf{out}}; n_{\mathsf{in}} - 1]\!]$ do
    /* Decompose the rest of the mask */
    Update $\mathsf{ct}_{\mathsf{out}} = \mathsf{ct}_{\mathsf{out}} - \left\langle \boldsymbol{k}_i, \mathbf{Dec}^{(\beta,\ell)}(a_i) \right\rangle$
return $\mathsf{ct}_{\mathsf{out}}$
```

Algorithm 10: $\mathsf{CT}_{\mathsf{out}} \leftarrow \mathsf{ConstantSampleExtract}^{-1}(\mathsf{ct}_{\mathsf{in}}, k, N)$

```
Context: {
    $\boldsymbol{s} \in \mathbb{Z}_q^n$ : the input LWE secret key
    $\boldsymbol{S}^{[n]} \in \mathfrak{R}_{q,N}^k$ : a partial secret key (Definition 6)
        such that its flattened version is $\boldsymbol{s}$ (Definition 7)
    $R := \sum_{i=1}^{N-1} r_i \cdot X^i \in \mathfrak{R}_{q,N}$, where $r_i$ are uniformly random
    $\mathsf{ct}_{\mathsf{in}} = (a_0, \cdots, a_{n-1}, b) \in \mathbb{Z}_q^{n+1}$
    $p \in \mathbb{Z}_q$
Input: {
    $\mathsf{ct}_{\mathsf{in}} \in \mathsf{LWE}_{\boldsymbol{s}}(p)$ : an LWE encryption of the plaintext $p$
    $k \in \mathbb{N}$ : the output GLWE dimension
    $N \in \mathbb{N}$ : the output polynomial size
Output: $\mathsf{CT}_{\mathsf{out}} \in \mathsf{GLWE}_{\boldsymbol{S}^{[n]}}(p_0 + R)$ : a GLWE encryption
/* put the $b$ part in a polynomial */
set $B' := b \in \mathfrak{R}_{q,N}$
/* put the rest in polynomials */
for $i \in [\![0; k \cdot N]\!]$ do
    set $\alpha := \left\lfloor \frac{i}{N} \right\rfloor$, $\beta := (N - i) \bmod N$ and $\gamma := 1 - (\beta == 0)$
    if $i \leq \phi - 1$ then
        set $a'_{\alpha,\beta} := (-1)^{\gamma} \cdot a_i$
    else
        set $a'_{\alpha,\beta} := 0$
return $\mathsf{CT}_{\mathsf{out}} := \left(A'_0 := \sum_{j=0}^{N-1} a'_{0,j} X^j, \cdots, A'_{k-1} := \sum_{j=0}^{N-1} a'_{k-1,j} X^j, B'\right) \in \mathfrak{R}_{q,N}^{k+1}$
```

Algorithm 11: $\mathsf{CT}_{\mathsf{out}} \leftarrow \mathsf{GlweKeySwitch}'(\mathsf{CT}_{\mathsf{in}}, \mathsf{KSK})$

```
Context:
    $\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}}$ : the input partial secret key (Definition 6)
    $\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} = (S_{\mathsf{in},0}, \cdots, S_{\mathsf{in},k_{\mathsf{in}}-1})$
    $\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}}$ : the output partial secret key (Definition 6)
    $\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} = (S_{\mathsf{out},0}, \cdots, S_{\mathsf{out},k_{\mathsf{out}}-1})$
    $(k_{\mathsf{in}} - 1)N < \phi_{\mathsf{in}} \leq k_{\mathsf{in}}N$ and $(k_{\mathsf{out}} - 1)N < \phi_{\mathsf{out}} \leq k_{\mathsf{out}}N$
    $\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]} \prec \boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}$ : shared randomness secret keys (Definition 9)
    $\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]} \neq \boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}$ and $k_{\mathsf{out}} \leq k_{\mathsf{in}}$
    $k \in \{k_{\mathsf{out}} - 1, k_{\mathsf{out}}\}$ such that $\forall 0 \leq i < k,\ S_{\mathsf{in},i} = S_{\mathsf{out},i}$
    $P \in \mathfrak{R}_{q,N}$
    $\mathsf{CT}_{\mathsf{in}} = (A_0, \cdots, A_{k_{\mathsf{in}}-1}, B) \in \mathfrak{R}_{q,N}^{k_{\mathsf{in}}+1}$
    $\mathsf{CT}_{i,j} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta^j} \cdot S_{\mathsf{in},i}\right)$, for $k_{\mathsf{out}} \leq i < k_{\mathsf{in}}$ and $0 \leq j \leq \ell - 1$
    if $k = k_{\mathsf{out}} - 1$ :
        $\mathsf{CT}_{k,j} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}\left(\frac{q}{\beta^j} \cdot (S_{\mathsf{in},k} - S_{\mathsf{out},k})\right)$, for $0 \leq j \leq \ell - 1$
    $\ell \in \mathbb{N}$ : the number of levels in the decomposition
    $\beta \in \mathbb{N}$ : the base in the decomposition
Input:
    $\mathsf{CT}_{\mathsf{in}} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{in}}^{[\phi_{\mathsf{in}}]}}(P)$
    $\mathsf{KSK} = \{\boldsymbol{K}_i = (\mathsf{CT}_{i,0}, \cdots, \mathsf{CT}_{i,\ell-1})\}_{k \leq i < k_{\mathsf{in}}}$
Output: $\mathsf{CT}_{\mathsf{out}} \in \mathsf{GLWE}_{\boldsymbol{S}_{\mathsf{out}}^{[\phi_{\mathsf{out}}]}}(P)$

/* Keep the $B$ part and the first part of the mask */
Set $\mathsf{CT}_{\mathsf{out}} := (A_0, \cdots, A_{k_{\mathsf{out}}-1}, B) \in \mathfrak{R}_{q,N}^{k_{\mathsf{out}}+1}$
/* Different public material for this potential partial-shared secret key
   polynomial */
if $k = k_{\mathsf{out}} - 1$ then
    Update $\mathsf{CT}_{\mathsf{out}} = \mathsf{CT}_{\mathsf{out}} - \left\langle \boldsymbol{K}_k, \mathbf{Dec}^{(\beta,\ell)}(A_k) \right\rangle$
/* Same process as in Algorithm 4 */
for $i \in [\![k_{\mathsf{out}}; k_{\mathsf{in}} - 1]\!]$ do
    /* Decompose the mask */
    Update $\mathsf{CT}_{\mathsf{out}} = \mathsf{CT}_{\mathsf{out}} - \left\langle \boldsymbol{K}_i, \mathbf{Dec}^{(\beta,\ell)}(A_i) \right\rangle$
return $\mathsf{CT}_{\mathsf{out}}$
```

D Parameter Sets

In this section we provide the formulae to compute the size of the key switching keys and bootstrapping keys, and we also give the most important parameter sets we used for our experiments. They are displayed in Tables 2, 3 and 4. To compute the filling amount of the FFT-KS ϕ_{in} of the FFT-based shrinking key switch, one needs to compute $\phi_{\mathsf{PBS}} - n$.

D.1 Size of Public Material

The size (in MB) of the traditional LWE-to-LWE key switching key $\mathsf{Size}\left(\mathsf{KSK}\right)$ is computed with the following formula:

$$\mathsf{Size}\left(\mathsf{KSK}\right) := (n+1) \cdot \ell_{\mathsf{KS}} \cdot k \cdot N \cdot 2^{-17}$$

The size (in MB) of the traditional bootstrapping key $\mathsf{Size}\left(\mathsf{BSK}\right)$ is computed with the following formula:

$$\mathsf{Size}\left(\mathsf{BSK}\right) := n \cdot \ell_{\mathsf{PBS}} \cdot (k+1)^2 \cdot N \cdot 2^{-17}$$

The size (in MB) of the FFT-Shrinking keyswitching key $\mathsf{Size}\left(\mathsf{FFT\text{-}KS}\right)$ is computed with the following formula:

$$\mathsf{Size}\left(\mathsf{FFT\text{-}KS}\right) := (k_{\mathsf{out}}+1) \cdot \ell_{\mathsf{KS}} \cdot k_{\mathsf{in}} \cdot N \cdot 2^{-17}$$

The size (in MB) of the Stair keyswitching key $\mathsf{Size}\left(\mathsf{Stair\text{-}KS}\right)$ with 2 steps is computed with the following formula:

$$\mathsf{Size}\left(\mathsf{Stair\text{-}KS}\right) := ((n_{\mathsf{KS}}+1) \cdot \ell_{\mathsf{KS}_1} \cdot (\phi - n_{\mathsf{KS}}) + (n+1) \cdot \ell_{\mathsf{KS}_2} \cdot (n_{\mathsf{KS}} - n)) \cdot 2^{-17}$$

For the shrinking key switch, it is enough to use one of the formulae above and changing some parameters. For instance, for an LWE-to-LWE shrinking (Algorithm 9) key switch one uses $\mathsf{Size}\left(\mathsf{KSK}\right)$ with $n = n_{\mathsf{in}} - n_{\mathsf{out}}$.

D.2 Parameter Sets

The following parameter sets are for a failure probability of $p_{\mathsf{fail}} \leq 2^{-13.9}$.

p	Partial Shared Keys	LWE-KS Algorithm	GLWE Parameters		PBS Parameters		LWE-KS Parameters		Metrics	
			Parameter	Value	Parameter	Value	Parameter	Value	Name	Value
1	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	588 -12.66 5 8 -31.07	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	15 1	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 3	time size	6.6480 58.6
1	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	532 -11.17 5 8 1280 -31.07	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	15 1	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	782 -17.82 9 1 2 4	time size	4.9808 44.45
1	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	534 -11.22 5 8 1280 -31.07	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	15 1	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 3 8 1 9	time size	3.8792 37.76
2	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	668 -14.79 6 8 -37.88	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	18 1	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	4 3	time size	10.185 87.45
2	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	576 -12.34 6 8 1536 -37.88	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	18 1	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	896 -20.85 10 1 2 5	time size	7.7625 66.55
2	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	590 -12.71 6 8 1536 -37.88	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	18 1	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	1 1 10 1 11	time size	5.9151 56.64
3	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	720 -16.17 4 9 -51.49	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	21 1	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	4 3	time size	14.704 104.1
3	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	648 -14.25 3 9 1536 -37.88	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	18 1	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	944 -22.13 7 2 2 6	time size	8.4892 57.83
3	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	686 -15.27 3 9 1536 -37.88	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	18 1	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	1 1 10 1 13	time size	6.0204 43.08
4	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	788 -17.98 2 10 -51.49	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	23 1	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	4 3	time size	16.265 92.39
4	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	664 -14.68 2 10 2048 -51.49	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	22 1	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	1126 -26.97 13 1 2 6	time size	12.658 68.68
4	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	682 -15.16 2 10 2048 -51.49	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	23 1	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 3 9 1 14	time size	8.7397 48.61

Table 2: Parameter sets, benchmarks for PBS+LWE-KS and sizes of public material for CJP and two variants based on both partial and shared randomness secret keys. Note that we use $\log_2(\nu) = p$. Sizes are given in MB and times in milliseconds.

p	Partial Shared Keys	LWE-KS Algorithm	GLWE Parameters		PBS Parameters		LWE-KS Parameters		Metrics	
			Parameter	Value	Parameter	Value	Parameter	Value	Name	Value
5	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	840 −19.36 1 11 −51.49	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	23 1	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 6	time size	24.964 131.3
5	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	732 −16.49 1 11 2048 −51.49	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	23 1	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	1171 −28.17 9 2 2 7	time size	18.638 78.62
5	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	766 −17.39 1 11 2048 −51.49	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	23 1	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 3 9 1 15	time size	13.089 48.58
6	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	840 −19.36 1 12 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	14 2	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 5	time size	67.688 341.4
6	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	748 −16.91 1 12 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	14 2	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	1313 −31.94 16 1 2 8	time size	53.320 224.2
6	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	774 −17.61 1 12 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	14 2	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	1 1 11 1 15	time size	45.647 194.0
7	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	896 −20.85 1 13 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	15 2	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 6	time size	147.05 784.4
7	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	776 −17.66 1 13 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	15 2	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	1332 −32.45 10 2 1 16	time size	111.14 463.3
7	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	818 −18.78 1 13 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	14 2	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	1 1 11 1 16	time size	98.870 409.5
8	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	968 −22.77 1 14 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	11 3	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 6	time size	467.25 2179
8	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	816 −18.72 1 14 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	11 3	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	1359 −33.17 9 2 1 17	time size	351.64 1304
8	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	854 −19.73 1 14 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	11 3	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	1 1 11 1 18	time size	326.44 1282

Table 3: Parameter sets, benchmarks for PBS+LWE-KS and sizes of public material for CJP and two variants based on both partial and shared randomness secret keys. Note that we use $\log_2(\nu) = p$. Sizes are given in MB and times in milliseconds.

p	Partial Shared Keys	LWE-KS Algorithm	GLWE Parameters		PBS Parameters		LWE-KS Parameters		Metrics	
			Parameter	Value	Parameter	Value	Parameter	Value	Name	Value
9	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	1024 −24.26 1 15 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	9 4	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 7	time size	1383.8 5890
9	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	860 −19.89 1 15 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	8 4	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	1388 −33.94 10 2 1 18	time size	1148.1 3525
9	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	902 −21.01 1 15 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	8 4	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	1 1 11 1 18	time size	1058.1 3609
10	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	1096 −26.17 1 16 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	6 6	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	2 12	time size	4794.4 19730
10	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	904 −21.06 1 16 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	6 6	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	1417 −34.71 11 2 1 19	time size	3721.0 10940
10	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	938 −21.97 1 16 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	6 6	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 3 9 1 20	time size	3628.1 11260
11	✗	traditional LWE-to-LWE	n $\log_2(\sigma_n)$ k $\log_2(N)$ $\log_2(\sigma_{k\cdot N})$	1132 −27.13 1 17 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	2 20	$\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	2 13	time size	37795 105300
11	✓	2 steps *(Alg. 2)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	984 −23.19 1 17 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	3 12	n_{KS} $\log_2\left(\sigma_{n_{\mathsf{KS}}}\right)$ $\log_2\left(\beta_{\mathsf{KS}_1}\right)$ ℓ_{KS_1} $\log_2\left(\beta_{\mathsf{KS}_2}\right)$ ℓ_{KS_2}	1471 −36.15 11 2 1 21	time size	18237 47330
11	✓	FFT-based *(Alg. 3)*	n $\log_2(\sigma_n)$ k $\log_2(N)$ ϕ $\log_2(\sigma_\phi)$	1018 −24.10 1 17 2443 −62.00	$\log_2(\beta_{\mathsf{PBS}})$ ℓ_{PBS}	3 13	k_{in} k_{out} $\log_2(N_{\mathsf{KS}})$ $\log_2(\beta_{\mathsf{KS}})$ ℓ_{KS}	3 3 9 1 22	time size	19224 52940

Table 4: Parameter sets, benchmarks for PBS+LWE-KS and sizes of public material for CJP and two variants based on both partial and shared randomness secret keys. Note that we use $\log_2(\nu) = p$. Sizes are given in MB and times in milliseconds.