

Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts

Pierrick Méaux¹(✉), Anthony Journault²,
François-Xavier Standaert², and Claude Carlet³

¹ INRIA, CNRS, ENS and PSL Research University, Paris, France
`Pierrick.Meaux@ens.fr`

² ICTEAM/ELEN/Crypto Group, Université catholique de Louvain,
Louvain-la-Neuve, Belgium
`{anthony.journault,fstandae}@uclouvain.be`

³ LAGA, Department of Mathematics,
University of Paris VIII and University of Paris XIII, Paris, France
`claude.carlet@gmail.com`

Abstract. Symmetric ciphers purposed for Fully Homomorphic Encryption (FHE) have recently been proposed for two main reasons. First, minimizing the implementation (time and memory) overheads that are inherent to current FHE schemes. Second, improving the homomorphic capacity, *i.e.* the amount of operations that one can perform on homomorphic ciphertexts before bootstrapping, which amounts to limit their level of noise. Existing solutions for this purpose suggest a gap between block ciphers and stream ciphers. The first ones typically allow a constant but small homomorphic capacity, due to the iteration of rounds eventually leading to complex Boolean functions (hence large noise). The second ones typically allow a larger homomorphic capacity for the first ciphertext blocks, that decreases with the number of ciphertext blocks (due to the increasing Boolean complexity of the stream ciphers' output). In this paper, we aim to combine the best of these two worlds, and propose a new stream cipher construction that allows constant and small(er) noise. Its main idea is to apply a Boolean (filter) function to a public bit permutation of a constant key register, so that the Boolean complexity of the stream cipher outputs is constant. We also propose an instantiation of the filter function designed to exploit recent (3rd-generation) FHE schemes, where the error growth is quasi-additive when adequately multiplying ciphertexts with the same amount of noise. In order to stimulate further investigation, we then specify a few instances of this stream cipher, for which we provide a preliminary security analysis. We finally highlight the good properties of our stream cipher regarding the other goal of minimizing the time and memory complexity of calculus delegation (for 2nd-generation FHE schemes). We conclude the paper with open problems related to the large design space opened by these new constructions.

1 Introduction

Purpose: Calculus Delegation. Recent years have witnessed massive changes in communication technologies, that can be summarized as a combination of

two trends: (1) the proliferation of small embedded devices with limited storage and computing facilities, and (2) the apparition of cloud services with extensive storage and computing facilities. In this context, the outsourcing of data and the delegation of data processing gains more and more interest. Yet, such new opportunities also raise new security and privacy concerns. Namely, users typically want to prevent the server from learning about their data and processing. For this purpose, Gentry’s breakthrough Fully Homomorphic Encryption (FHE) scheme [30] brought a perfect conceptual answer. Namely, it allows applying processing on ciphertexts in a homomorphic way so that after decryption, plaintexts have undergone the same operations as ciphertexts, but the server has not learned anything about these plaintexts.¹

Application Scenario. Cloud services can be exploited in a plethora of applications, some of them surveyed in [51]. In general, they are always characterized by the aforementioned asymmetry between the communication parties. For illustration, we start by providing a simple example where data outsourcing and data processing delegation require security and privacy. Let us say that a patient, Alice, has undergone a surgery and is coming back home. The hospital gave her a monitoring watch (with limited storage) to measure her metabolic data on a regular basis. And this metabolic data should be made available to the doctor Bob, to follow the evolution of the post-surgery treatment. Quite naturally, Bob has numerous patients and no advanced computing facilities to store and process the data of all his patients. So this is a typical case where sending the data to a cloud service would be very convenient. That is, Alice’s data could be sent to and stored on the cloud, and associated to both her and the doctor Bob. And the cloud would provide Bob with processed information in a number of situations such as when the metabolic data of Alice is abnormal (in which case an error message should be sent to Bob), or during an appointment between Alice and Bob, so that Bob can follow the evolution of Alice’s data (possibly after some processing). Bob could in fact even be interested by accessing some other patient’s data, in order to compare the effect of different medications. And of course, we would like to avoid the cloud to know anything about the (private) data it is manipulating.

Typical Framework. More technically, the previous exemplary application can be integrated in a quite general cloud service application framework, that can be seen as a combination of 5 steps, combining a symmetric encryption scheme and an asymmetric homomorphic encryption scheme, as summarised in Fig. 1 and described next:

1. *Initialization.* Alice runs the key generation algorithms $H.\text{KeyGen}$ and $S.\text{KeyGen}$ of the two schemes, and sends her homomorphic public key pk^H and the homomorphic ciphertext of her symmetric key $\mathbf{C}^H(\text{sk}_i^S)$.

¹ In the remaining of the paper, and when not specified otherwise, the term FHE will also be used for related schemes such as Leveled HE, SomeWhat HE, Scalable HE, *etc.*.

2. *Storage.* Alice encrypts her data m_i with the symmetric encryption scheme, and sends $C^S(m_i)$ to Claude.
3. *Evaluation.* Claude homomorphically evaluates, with the $H.Eval$ algorithm, the decryption $C^H(m_i)$ of the symmetric scheme on Alice's data $C^S(m_i)$.
4. *Computation.* Claude homomorphically executes the treatment f on Alice's encrypted data.
5. *Result.* Claude sends a compressed encrypted result of the data treatment $c^H(f(m_i))$, obtained with the $H.Comp$ algorithm, and Alice decrypts it.

Note that if we assume the existence of a trusted third party active only during the initialization step, Alice can avoid Step 1, which needs a significant computational and memory storage effort. Note also that this framework is versatile: computation can be done in parallel (in a batch setting) or can be turned into a secret key FHE.

	<u>Alice</u>	<u>Claude</u>
1: Initialization	$(sk^H, pk^H) \leftarrow H.KeyGen(\lambda)$ $sk^S \leftarrow S.KeyGen(\lambda)$ $C^H(sk_i^S), pk^H$	$C^H(sk_i^S), pk^H$
2: Storage	$C^S(m_i) = S.Enc(m_i, sk^S) \xrightarrow{C^S(m_i)}$	$C^S(m_i)$
3: Evaluation		$C^H(m_i)$ $=$ $H.Eval(S.Dec(C^S(m_i), C^H(sk_i^S), pk^H))$
4: Computation	$f \xrightarrow{f}$	$C^H(f(m_i)) = H.Eval(f(C^H(m_i)))$
5: Result	$c^H(f(m_i))$ $f(m_i) = H.Dec(c^H(f(m_i)), sk^H)$	$c^H(f(m_i)) = H.Comp(C^H(f(m_i)))$

Fig. 1. Homomorphic Encryption - Symmetric Encryption framework. H and S respectively refer to homomorphic and symmetric encryption schemes, for algorithms (e.g. $H.KeyGen$) or scheme components (e.g. sk^S).

FHE Bottlenecks. The main limitation for the deployment of cloud services based on such FHE frameworks relates to its important overheads, that can be related to two main concerns: computational and memory costs (especially on the client side) and limited homomorphic capacity (i.e. noise increase). More precisely:

- The computational and memory costs for the client depend overwhelmingly on the homomorphic encryption and decryption algorithms during the steps 1 and 5. The memory cost is mostly influenced by the homomorphic ciphertexts and public key sizes. Solving these two problems consists in building size-efficient FHE schemes with low computational cost [35,38]. On the server side, this computational cost further depends on the symmetric encryption scheme and function to evaluate.
- The homomorphic capacity relates to the fact that FHE constructions are built on noise-based cryptography, where the unbounded amount of homomorphic operations is guaranteed by an expensive bootstrapping technique. The homomorphic capacity corresponds to the amount of operations doable before the noise grows too much forcing to use bootstrapping. Therefore, and in order to reduce the time and computational cost of the framework, it is important to manage the error growth during the homomorphic operations (*i.e.* steps 3 and 4). Furthermore, since the 4th step is the most important one from the application point-of-view (since this is where the useful operations are performed by the cloud), there is strong incentive to minimize the cost of the homomorphic decryption in the 3rd step.

Previous Works. In order to mitigate these bottlenecks, several works tried to reduce more and more the homomorphic cost of evaluating a symmetric decryption algorithm. First attempts in this direction, which were also used as benchmark for FHE implementations, used the AES for this purpose [15,31]. Various alternative schemes were also considered, all with error and sizes depending on the multiplicative depth of the symmetric encryption scheme, such as BGV [9] and FV [26]. Additional optimizations exploited batching and bitslicing, leading to the best results of performing 120 AES decryptions in 4 minutes [31].

Since the multiplicative depth of the AES decryption evaluation was a restrictive bound in these works, other symmetric encryption schemes were then considered. The most representative attempts in this direction are the family of block ciphers LowMC [1] and the stream cipher Kreyvium [11]. These constructions led to reduced and more suitable multiplicative depths. Yet, and intuitively, these attempts were still limited by complementary drawbacks. First for LowMC, the remaining multiplicative depth remains large enough to significantly reduce the homomorphic capacity (*i.e.* increase the noise). Such a drawback seems to be inherent in block cipher structures where the iteration of rounds eventually leads to Boolean functions with large algebraic degree, which inevitably imply a constant per block but high noise after homomorphic evaluation. For example, ciphers dedicated to efficient masking against side-channel attacks [33,34,52], which share the goal of minimizing the multiplicative complexity, suffer from similar issues and it seems hard to break the barrier of one multiplication per round (and therefore of 12 to 16 multiplications for 128-bit ciphers). Second for Kreyvium, the error actually grows with the number of evaluated ciphertexts, which implies that at some point, the output ciphertexts are too noisy, and cannot be decrypted (which requires either to bootstrap or to re-initialize the stream cipher).

Our Contribution. In view of this state-of-the-art, a natural direction would be to try combining the best of these two previous works. That is, to design a cipher inheriting from the constant noise property offered by block ciphers, and the lower noise levels of stream ciphers (due to the lower algebraic degree of their outputs), leading to the following contributions.

First, we introduce a new stream cipher construction, next denoted as a filter permutator (by analogy with filter generators). Its main design principle is to filter a constant key register with a variable (public) bit permutation. More precisely, at each cycle, the key register is (bit) permuted with a pseudorandomly generated permutation, and we apply a non-linear filtering function to the output of this permuted key register. The main advantage of this construction is to always apply the non-linear filtering directly on the key bits, which allows maintaining the noise level of our outputs constant. Conceptually, this type of construction seems appealing for any FHE scheme.

Second, and going deeper in the specification of a concrete scheme, we discuss the optimization of the components in a filter permutator, with a focus on the filtering function (which determines the output noise after homomorphic evaluation). For this purpose, we first notice that existing FHE schemes can be split in (roughly) two main categories. On one hand the so-called *2nd-generation* FHE (such as [9,15]) where the metric for the noise growth is essentially the multiplicative depth of the circuit to homomorphically evaluate. On the other hand, the so-called *3rd-generation* FHE (such as [2,32]) where the error growth is asymmetric, and in particular quasi-additive when considering a multiplicative chain. From these observations, we formalize a *comb* structure which can be represented as a (possibly long) multiplicative chain, in order to take the best advantage of 3rd-generation FHE schemes. We then design a filtering function based on this comb structure (combined with other technical ingredients in order to prevent various classes of possible attacks against stream ciphers) and specify a family of filter permutators (called FLIP).

Third, and in order to stimulate further investigations, we instantiate a few version of FLIP designs, for 80-bit and 128-bit security. We then provide a preliminary evaluation of their security against some of the prevailing cryptanalysis from the open literature – such as (fast) algebraic attacks, (fast) correlation attacks, BKW-like attacks [6], guess and determine attacks, *etc.* – based on state-of-the-art tools. We also analyze the noise brought by their filtering functions in the context of 3rd-generation FHE. In this respect, our main result is that we can limit the noise after the homomorphic evaluation of a decryption to a level of the same order of magnitude as for a single homomorphic multiplication - hence essentially making the impact of the symmetric encryption scheme as small as possible.

We finally observe that our FLIP designs have a very reduced multiplicative depth, which makes them suitable for 2nd-generation FHE schemes as well, and provide preliminary results of prototype implementations using HELib that confirm their good behavior compared to state-of-the-art block and stream ciphers designed for efficient FHE.

Overall, filter permutators in general and FLIP instances in particular open a large design space of new symmetric constructions to investigate. Hence, we conclude the paper with a list of open problems regarding these algorithms, their best cryptanalysis, the Boolean functions used in their filter and their efficient implementation in concrete applications.

2 Background

2.1 Boolean Functions

In this section, we recall the cryptographic properties of Boolean functions that we will need in the rest of the paper (mostly taken from [12]).

Definition 1 (Boolean Function). *A Boolean function f with n variables is a function from \mathbb{F}_2^n to \mathbb{F}_2 . The set of all Boolean functions in n variables is denoted by \mathcal{B}_n .*

Definition 2 (Walsh Transform). *Let $f \in \mathcal{B}_n$ a Boolean function. Its Walsh Transform W_f at $a \in \mathbb{F}_2^n$ is defined as:*

$$W_f(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) + \langle a, x \rangle},$$

where $\langle a, x \rangle$ denotes the inner product in \mathbb{F}_2^n .

Definition 3 (Balancedness). *A Boolean function $f \in \mathcal{B}_n$ is said to be balanced if its outputs are uniformly distributed over $\{0, 1\}$.*

Definition 4 (Non-linearity). *The non-linearity NL of a Boolean function $f \in \mathcal{B}_n$, where n is a positive integer, is the minimum Hamming distance between f and all the affine functions g :*

$$NL(f) = \min_g \{d_H(f, g)\},$$

with $d_H(f, g) = \#\{x \in \mathbb{F}_2^n \mid f(x) \neq g(x)\}$ the Hamming distance between f and g . The non-linearity of a Boolean function can also be defined by its Walsh Transform:

$$NL(f) = 2^{n-1} - \frac{1}{2} \max_{a \in \mathbb{F}_2^n} |W_f(a)|.$$

Definition 5 (Resiliency). *A Boolean function $f \in \mathcal{B}_n$ is said m -resilient if any of its restrictions obtained by fixing at most m of its coordinates is balanced. We will denote by $\text{res}(f)$ the resiliency m of f and set $\text{res}(f) = -1$ if f is unbalanced.*

Definition 6 (Algebraic Immunity). *The algebraic immunity of a Boolean function $f \in \mathcal{B}_n$, denoted as $\text{Al}(f)$, is defined as:*

$$\text{Al}(f) = \min_{g \neq 0} \{\text{deg}(g) \mid fg = 0 \text{ or } (f \oplus 1)g = 0\},$$

where $\text{deg}(g)$ is the degree of g . The function g is called an annihilator of f (or $(f \oplus 1)$).

Definition 7 (Fast Algebraic Immunity). *The fast algebraic immunity of a Boolean function $f \in \mathcal{B}_n$, denoted as $\text{FAI}(f)$, is defined as:*

$$\text{FAI}(f) = \min\{2\text{Al}(f), \min_{1 \leq \text{deg}(g) < \text{Al}(f)} (\max[\text{deg}(g) + \text{deg}(fg), 3\text{deg}(g)])\}.$$

Summarizing, the good balancedness, non-linearity and resiliency properties have to be ensured to withstand correlation attacks [56] and fast correlation attacks [48]. The high algebraic immunity and fast algebraic immunity have to be ensured to withstand algebraic attacks [13].

2.2 (Ring) Learning with Errors

In this section, we recall useful notations and definitions needed about the decisional LWE problem and its ring variation. For an integer modulus q , we denote by \mathbb{Z}_q the quotient ring of integers modulo q . We denote vectors with bold letters \mathbf{e} and matrices with bold capital letters \mathbf{A} . The notation $s \leftarrow_{\$} S$ (resp. $s \leftarrow_{\$} \chi$) denotes that s is picked uniformly at random from a finite set S (resp. from a distribution χ).

The decisional Learning With Error problem (dLWE) was introduced by Regev [53].

Definition 8 (dLWE). *For an integer $q = q(n) \geq 2$, an adversary \mathcal{A} and an error distribution $\chi = \chi(n)$ over \mathbb{Z}_q , we define the following advantage function:*

$$\text{Adv}_{\mathcal{A}}^{\text{dLWE}_{n,m,q,\chi}} := |\Pr[\mathcal{A}(\mathbf{A}, \mathbf{z}_0) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{z}_1) = 1]|,$$

where

$$\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow_{\$} \mathbb{Z}_q^n, \mathbf{e} \leftarrow_{\$} \chi^m, \mathbf{z}_0 := \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top \quad \text{and} \quad \mathbf{z}_1 \leftarrow_{\$} \mathbb{Z}_q^m.$$

The $\text{dLWE}_{n,m,q,\chi}$ assumption asserts that for all PPT adversaries \mathcal{A} , the advantage $\text{Adv}_{\mathcal{A}}^{\text{dLWE}_{n,m,q,\chi}}$ is a negligible function in n .

The ring variant was introduced by Lyubashevsky, Peikert and Regev in [46].

Definition 9 (dR-LWE). *For a polynomial ring $R = \mathbb{Z}[X]/f(X)$ with f of degree n , an integer $q \geq 2$, an adversary \mathcal{A} and an error distribution χ over $R_q = R/qR$, R^\vee being R dual fractional ideal, we define the following advantage function:*

$$\text{Adv}_{\mathcal{A}}^{\text{dRLWE}_{R,q,\chi}} := |\Pr[\mathcal{A}(a, z_0) = 1] - \Pr[\mathcal{A}(a, z_1) = 1]|,$$

where

$$a \leftarrow_{\S} R_q, s \leftarrow_{\S} R_q^{\vee}, e \leftarrow_{\S} \chi, z_0 := a \cdot s + e \quad \text{and} \quad z_1 \leftarrow_{\S} R.$$

With $f(X)$ a cyclotomic polynomial, the $\text{dRLWE}_{R,q,\chi}$ assumption asserts that for all PPT adversaries \mathcal{A} , the advantage $\text{Adv}_{\mathcal{A}}^{\text{dRLWE}_{R,q,\chi}}$ is a negligible function in n .

For our constructions, we need to take the distribution χ as a subgaussian random variable which we define hereafter. More details about the subgaussian distribution and the lemmas’ proof can be found in [2,58].

Definition 10 (Subgaussian Random Variables). *Let X be a random variable. We say X is subgaussian with parameter σ if there exists σ such that:*

$$\forall t \in \mathbb{R}, \mathbb{E}[e^{tX}] \leq e^{\sigma^2 t^2 / 2},$$

where $\mathbb{E}[e^{tX}]$ is the moment generating function of X .

Lemma 1 (Subgaussian Random Variables Properties). *Let X, X' be independent subgaussian random variables of parameter σ and σ' respectively. Assuming $\mathbb{E}(X) = \mathbb{E}(X') = 0$ we have the following properties:*

- *Tails:* $\forall t \geq 0$ we have $\Pr[|X| \geq t] \leq 2e^{-\pi t^2 / \sigma^2}$.
- *Homogeneity:* $\forall c \in \mathbb{R}$, cX is subgaussian with parameter $|c|\sigma$.
- *Pythagorean additivity:* $X + X'$ is subgaussian with parameter $\sqrt{\sigma^2 + \sigma'^2}$.

We extend the notion of subgaussianity to vectors and polynomials. Since the coefficients of a polynomial are seen as a vector, we call subgaussian vector of parameter σ a vector where each coefficient follows an independent subgaussian distribution with parameter σ .

Lemma 2 (Subgaussian Vector Norm, Adapted from [2], Lemma 2.1). *Let $\mathbf{x} \in \mathbb{R}^n$ be a random vector where each coordinate follows an independent subgaussian distribution of parameter σ . Then for some universal constant $C > 0$ we have $\Pr[||\mathbf{x}||_2 > C\sigma\sqrt{n}] \leq 2^{-\Omega(n)}$ and therefore $||\mathbf{x}||_2 = \mathcal{O}(\sigma\sqrt{n})$.*

2.3 Fully Homomorphic Encryption

In this section we recall the definition of (Fully) Homomorphic Encryption and present the Homomorphic Encryption schemes we will use, both based on GSW [32].

Definition 11 (Homomorphic Encryption Scheme). *Let \mathcal{M} be the plaintext space, \mathcal{C} the ciphertext space and λ the security parameter. A homomorphic encryption scheme consists of four algorithms:*

- $H.\text{KeyGen}(1^\lambda)$. Output pk^H and sk^H the public and secret keys of the scheme.
- $H.\text{Enc}(m, pk^H)$. From the plaintext $m \in \mathcal{M}$ and the public key, output a ciphertext $c \in \mathcal{C}$.
- $H.\text{Dec}(c, sk^H)$. From the ciphertext $c \in \mathcal{C}$ and the secret key, output $m' \in \mathcal{M}$.
- $H.\text{Eval}(f, c_1, \dots, c_k, pk^H)$. With $c_i = H.\text{Enc}(m_i, pk^H)$ for $1 \leq i \leq k$, output a ciphertext $c_f \in \mathcal{C}$ such that $H.\text{Dec}(c_f) = f(m_1, \dots, m_k)$.

A homomorphic encryption scheme is called a Fully Homomorphic Encryption (FHE) scheme when f can be any function and $|\mathcal{C}|$ is finite. A simpler primitive to consider is the SomeWhat Homomorphic Encryption (SWHE) scheme, where f is restricted to be any univariate polynomial of finite degree.

Since the breakthrough work of Gentry [30], the only known way to obtain FHE consists in adding a bootstrapping technique to a SWHE. As bootstrapping computational cost is still expensive in comparison to the other FHE algorithms, in the following part of the article we will only consider SWHE for our applications.

GSW Homomorphic Encryption Scheme. In 2013, Gentry, Sahai and Waters [32] introduced a Homomorphic Encryption scheme based on LWE using a new technique stemming from the *approximate eigenvector problem*. This new technique led to a new family of FHE, called 3rd-generation FHE, consisting in Homomorphic Encryption schemes such that the multiplicative error growth is quasi-additive. Hereafter, we present two schemes belonging to this generation, the first one with security based on dLWE and the second one based on dRLWE. We first set some useful notations considering the different schemes.

For a matrix \mathbf{E} we refer to the i -th row as \mathbf{e}_i^\top and to the j -th column as \mathbf{e}_j . The $\log q$ notation refers to the logarithm in base 2 of q . The notation $[a]_q$ is for $a \bmod q$ and $\lfloor [a]_q \rfloor_2 \in \{0, 1\}$ is a function in $a \in \mathbb{Z}_q$ giving 1 if $\lfloor \frac{a}{4} \rfloor \leq a \leq \lfloor \frac{3q}{4} \rfloor \bmod q$ and 0 otherwise. We denote by $[n]$ the set of integers $\{1, \dots, n\}$. We finally use $|x|$ and $\|x\|_2$ for the standard norms 1 and 2 on vectors $x \in \mathbb{R}^n$.

Batched GSW. This scheme is a batched version of GSW presented in [36], enabling to pack independently r plaintexts in one ciphertext. From the security parameter λ and the considered applications, we can derive the parameters n, q, r, χ of the scheme described below.

$H.\text{KeyGen}(n, q, r, \chi)$. On inputs the lattice dimension n , the modulus q , the number of bits by ciphertext r and the error distribution χ do:

- Set $\ell = \lceil \log q \rceil$, $m = \mathcal{O}(n\ell)$, $N = (r + n)\ell$, $\mathcal{M} = \{0, 1\}^r$ and $\mathcal{C} = \mathbb{Z}_q^{(r+n) \times N}$.
- Pick $\mathbf{A} \leftarrow_{\S} \mathbb{Z}_q^{n \times m}$, $\mathbf{S}' \leftarrow_{\S} \chi^{r \times n}$ and $\mathbf{E} \leftarrow_{\S} \chi^{r \times m}$.
- Set $\mathbf{S} = [\mathbf{I} \mid -\mathbf{S}'] \in \mathbb{Z}_q^{r \times (r+n)}$ and $\mathbf{B} = \begin{bmatrix} \mathbf{S}'\mathbf{A} + \mathbf{E} \\ \mathbf{A} \end{bmatrix}_q \in \mathbb{Z}_q^{(r+n) \times m}$.
- For all $\mathbf{m} \in \{0, 1\}^r$:
 - Pick $\mathbf{R}_m \leftarrow_{\S} \{0, 1\}^{m \times N}$.

- Set $\mathbf{P}_m = \left[\mathbf{BR}_m + \begin{pmatrix} m_1 \cdot \mathbf{s}_1^\top \\ \vdots \\ m_r \cdot \mathbf{s}_r^\top \\ \mathbf{0} \end{pmatrix} \mathbf{G} \right]_q \in \mathbb{Z}_q^{(r+n) \times N}$.

with \mathbf{s}_i^\top the i -th row of \mathbf{S} and $\mathbf{G} = (2^0, \dots, 2^{\ell-1})^\top \otimes \mathbf{I} \in \mathbb{Z}_q^{(r+n) \times N}$.

- Output $\text{pk}^H := (\{\mathbf{P}_m\}, \mathbf{B})$ and $\text{sk}^H := \mathbf{S}$.

$H.\text{Enc}(\text{pk}^H, \mathbf{m})$. On input pk^H , and $\mathbf{m} \in \{0, 1\}^r$, do:

- Pick $\mathbf{R} \leftarrow_{\S} \{0, 1\}^{m \times N}$, and output $\mathbf{C} = [\mathbf{BR} + \mathbf{P}_m]_q \in \mathbb{Z}_q^{(r+n) \times N}$.

$H.\text{Dec}(\mathbf{C}, \text{sk}^H)$. On input the secret key sk^H , and a ciphertext \mathbf{C} , do:

- For all $i \in [r] : m'_i = \lfloor \lfloor \langle \mathbf{s}_i^\top, \mathbf{c}_{i\ell} \rangle \rfloor_q \rfloor_2$ where $\mathbf{c}_{i\ell}$ is the column $i\ell$ of \mathbf{C} .
- Output $m'_1, \dots, m'_r \in \{0, 1\}^r$.

Note that $\mathbf{SC} = \mathbf{SBR} + \mathbf{SP}_m = \mathbf{ER} + \mathbf{ER}_m + \begin{pmatrix} m_1 \cdot \mathbf{s}_1^\top \\ \vdots \\ m_r \cdot \mathbf{s}_r^\top \end{pmatrix} \mathbf{G} = \mathbf{E}' + \begin{pmatrix} m_1 \cdot \mathbf{s}_1^\top \\ \vdots \\ m_r \cdot \mathbf{s}_r^\top \end{pmatrix} \mathbf{G}$.

The $H.\text{Eval}$ algorithm finally consists in iterating, following a circuit f , the homomorphic operations $H.\text{Add}$ and $H.\text{Mul}$:

- $H.\text{Add}(\mathbf{C}_1, \mathbf{C}_2) : \mathbf{C}_+ = \mathbf{C}_1 + \mathbf{C}_2$.
- $H.\text{Mul}(\mathbf{C}_1, \mathbf{C}_2) : \mathbf{C}_\times = \mathbf{C}_1 \times \mathbf{G}^{-1} \mathbf{C}_2$ with \mathbf{G}^{-1} a function such that $\forall \mathbf{C} \in \mathbb{Z}_q^{(r+n) \times N}, \mathbf{G}\mathbf{G}^{-1}(\mathbf{C}) = \mathbf{C}$ and the values of $\mathbf{G}^{-1}(\mathbf{C})$ follow a subgaussian distribution with parameter $\mathcal{O}(1)$ (see [49] for the existence and proof of \mathbf{G}^{-1}).

The correctness and security of this scheme are proven in the extended version of this paper.

Remark 1. For practical use, we only need to store $r + 1$ matrices \mathbf{P}_m , namely the $r + 1$ ones with \mathbf{m} of hamming weight equal to 0 or 1 are sufficient to generate correct encryption of all $\mathbf{m} \in \{0, 1\}^r$ with at most r additions of the corresponding \mathbf{P}_m matrices.

Ring-GSW. This scheme is a ring version of GSW presented in [38], transposing the *approximate eigenvector problem* into the ring setting. From λ the security parameter and the considered applications, we can derive the parameters n, q and \mathcal{M} of the scheme described below.

$H.\text{KeyGen}(n, q, \chi, \mathcal{M})$. On inputs the lattice dimension n , which is set to a power of 2, the modulus q , the error distribution χ and the plaintext space \mathcal{M} do:

- Set $R = \mathbb{Z}[X]/(X^n + 1)$, $R_q = R/qR$, $\ell = \lceil \log q \rceil$, $N = 2\ell$ and $\mathcal{C} = R_q^{2 \times N}$.
- Set $R_{0,1} = \{P \in R_q, p_i \in \{0, 1\}, 0 \leq i < n\}$.
- Pick $a \leftarrow_{\S} R_q$, $s' \leftarrow_{\S} \chi$ and $e \leftarrow_{\S} \chi$.

- Set $\mathbf{s} = [1 \mid -s']^\top \in R_q^{1 \times 2}$ and $\mathbf{b} = \begin{pmatrix} s'a + e \\ a \end{pmatrix} \in R_q^{2 \times 1}$.
- Output $\text{pk}^H := \mathbf{b}$ and $\text{sk}^H := \mathbf{s}$.

$H.\text{Enc}(\text{pk}^H, m)$. On input pk^H , and $m \in \mathcal{M}$, do:

- Pick $\mathbf{E} \leftarrow_{\$} \chi^{2 \times N}$.
- Pick $\mathbf{r} \leftarrow_{\$} R_{0,1}^N$, and output $\mathbf{C} = [\mathbf{b}\mathbf{r}^\top + m\mathbf{G} + \mathbf{E}]_q \in R_q^{2 \times N}$.

$H.\text{Dec}(\mathbf{C}, \text{sk}^H)$. On input the secret key sk^H , and a ciphertext \mathbf{C} , do:

- Compute $m' = \lfloor \langle \mathbf{s}, \mathbf{c}_l \rangle \rfloor_q \rfloor_2$.
- Output $m' \in R_q$.

The $H.\text{Eval}$ algorithm finally consists in iterating $H.\text{Add}$ and $H.\text{Mul}$:

- $H.\text{Add}(\mathbf{C}_1, \mathbf{C}_2) : \mathbf{C}_+ = \mathbf{C}_1 + \mathbf{C}_2$.
- $H.\text{Mul}(\mathbf{C}_1, \mathbf{C}_2) : \mathbf{C}_\times = \mathbf{C}_1 \times \mathbf{G}^{-1}\mathbf{C}_2$.

The correctness and security of this scheme are proven in the extended version of this paper.

Remark 2. The plaintext space \mathcal{M} has a major influence on the considered application in terms of quantity of information contained in a single ciphertext and error growth. For our application we choose \mathcal{M} as the set of polynomials with all coefficients of degree greater than 0 being zero, and the constant coefficient being bounded.

3 New Stream Cipher Constructions

In this section, we introduce our new stream cipher construction. We first describe the general filter permutator structure. Next we list a number of Boolean building blocks together with their necessary cryptographic properties. Third, we specify a family of filter permutators (denoted as FLIP) and analyze its security based on state-of-the art cryptanalysis and design tools. Finally, we propose a couple of parameters to fully instantiate a few examples of FLIP designs.

3.1 Filter Permutators

The general structure of filter permutators is depicted in Fig. 2. It is composed of three parts: a register where the key is stored, a (bit) permutation generator parametrised by a Pseudo Random Number Generator (PRNG) [7, 37] (which is initialized with a public IV), and a filtering function which generates a keystream. The filter permutator can be compared to a filter generator, in which the LFSR is replaced by a permuted key register. In other words, the register is

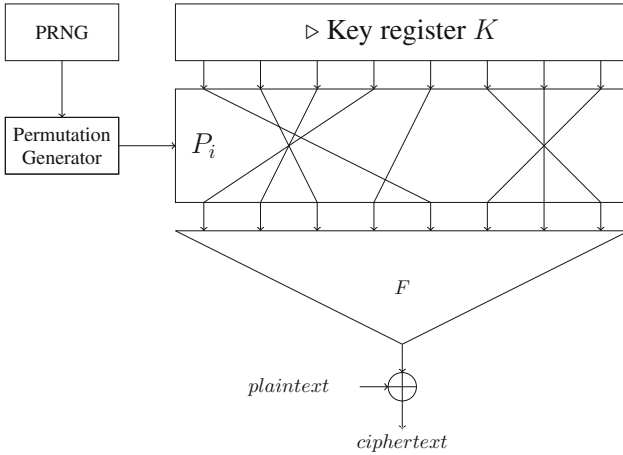


Fig. 2. Filter permutator construction.

no longer updated by means of the LFSR, but with pseudorandom bit permutations. More precisely, at each cycle (*i.e.* each time the filtering function outputs a bit), a pseudo-random permutation is applied to the register and the permuted key register is filtered. Eventually, the encryption (*resp.* decryption) with a filter permutator simply consists in XORing the bits output by the filtering function with those of the plaintext (*resp.* ciphertext).

3.2 Boolean Building Blocks for the Filter Permutator

We will first exploit direct sums of Boolean functions defined as follows:

Definition 12 (Direct Sum). Let $f_1(x_0, \dots, x_{n_1-1})$ and $f_2(x_{n_1}, \dots, x_{n_1+n_2-1})$ be two Boolean functions in respectively n_1 and n_2 variables. The direct sum of f_1 and f_2 is defined as $f = f_1 \oplus f_2$, which is a Boolean function in $n_1 + n_2$ variables such that:

$$f(x_0, \dots, x_{n_1+n_2-1}) = f_1(x_0, \dots, x_{n_1-1}) \oplus f_2(x_{n_1}, \dots, x_{n_1+n_2-1}).$$

They inherit from the following set of properties, proven in the extended version of this paper.

Lemma 3 (Direct Sum Properties). Let f be the direct sum of f_1 and f_2 with n_1 and n_2 variables respectively. Then f has the following cryptographic properties:

1. Non Linearity: $NL(f) = 2^{n_2}NL(f_1) + 2^{n_1}NL(f_2) - 2NL(f_1)NL(f_2)$.
2. Resiliency: $res(f) = res(f_1) + res(f_2) + 1$.
3. Algebraic Immunity: $AI(f_1) + AI(f_2) \geq AI(f) \geq \max(AI(f_1), AI(f_2))$.
4. Fast Algebraic Immunity: $FAI(f) \geq \max(FAI(f_1), FAI(f_2))$.

Our direct sums will then be based on three parts: a linear function, a quadratic function and triangular functions, defined as follows.

Definition 13 (Linear Functions). Let $n > 0$ be a positive integer, the L_n linear function is a n -variable Boolean function defined as:

$$L_n(x_0, \dots, x_{n-1}) = \sum_{i=0}^{n-1} x_i.$$

Definition 14 (Quadratic Functions). Let $n > 0$ be a positive integer, the Q_n linear function is a $2n$ -variable Boolean function defined as:

$$Q_n(x_0, \dots, x_{2n-1}) = \sum_{i=0}^{n-1} x_{2i}x_{2i+1}.$$

Definition 15 (Triangular Functions). Let $k > 0$ be a positive integer. The k -th triangular function T_k is a $\frac{k(k+1)}{2}$ -variable Boolean function defined as:

$$T_k(x_0, \dots, x_{\frac{k(k+1)}{2}-1}) = \sum_{i=1}^k \prod_{j=0}^{i-1} x_{j+\sum_{\ell=0}^{i-1} \ell}.$$

For example, the 4th triangular function T_4 is:

$$T_4 = x_0 \oplus x_1x_2 \oplus x_3x_4x_5 \oplus x_6x_7x_8x_9.$$

These three types of functions allow us to guarantee the following properties.

Lemma 4 (Linear Functions Properties). Let L_n be a linear function in n variables, then L_n has the following cryptographic properties:

1. Non Linearity: $NL(L_n) = 0$.
2. Resiliency: $res(L_n) = n - 1$.
3. Algebraic Immunity: $AI(L_n) = 1$.
4. Fast Algebraic Immunity: $FAI(L_n) = 2$.

Lemma 5 (Quadratic Functions Properties). Let Q_n be a linear function in $2n$ variables, then Q_n has the following cryptographic properties:

1. Non Linearity: $NL(Q_n) = 2^{2n-1} - 2^{n-1}$.
2. Resiliency: $res(Q_n) = -1$.
3. Algebraic Immunity: $AI(Q_1) = 1$ and $\forall n > 1, AI(Q_n) = 2$.
4. Fast Algebraic Immunity: $FAI(Q_1) = 2$ and $\forall n > 1, FAI(Q_n) = 4$.

Lemma 6 (Triangular Functions Properties). Let k a positive integer and let T_k the k -th triangular function. Then the following properties hold:

1. Non Linearity follows the recursive formula defined as:
 - (i) $NL(T_1) = 0$,
 - (ii) $NL(T_{k+1}) = (2^{k+1} - 2)NL(T_k) + 2^{k(k+1)/2}$.
2. Resiliency: $res(T_k) = 0$.
3. Algebraic Immunity: $AI(T_k) = k$.
4. Fast Algebraic Immunity: $FAI(T_k) = k + 1$.

The proof of Lemma 6 can be found in the extended version of this paper.

3.3 The FLIP Family of Stream Ciphers

Based on the previous definitions, we specify the FLIP family of stream ciphers as a filter permutator using a forward secure PRNG [5] based on the AES-128 (e.g. as instantiated in the context of leakage-resilient cryptography [57]), the Knuth shuffle (see below) as bit permutation generator and such that the filter F is the N -variable Boolean function defined by the direct sum of three Boolean functions f_1 , f_2 and f_3 of respectively n_1 , n_2 and n_3 variables, such that:

- $f_1(x_0, \dots, x_{n_1-1}) = L_{n_1}$,
- $f_2(x_{n_1}, \dots, x_{n_1+n_2-1}) = Q_{n_2/2}$,
- $f_3(x_{n_1+n_2}, \dots, x_{n_1+n_2+n_3-1})$ is the direct sum of nb triangular functions T_k , i.e. such that each T_k acts on different and independent variables, that we denote as ${}^{nb}\Delta^k$.

That is, we have $F : \mathbb{F}_2^{n_1+n_2+n_3} \rightarrow \mathbb{F}_2$ the Boolean function such that:

$$F(x_0, \dots, x_{n_1+n_2+n_3-1}) = L_{n_1} \oplus Q_{n_2/2} \oplus \bigoplus_{i=1}^{nb} T_k.$$

In the following section, we provide a preliminary security analysis of the FLIP filter permutators against a couple of standard attacks against stream ciphers, based on state-of-the-art tools. For this purpose, we will assume that no additional weaknesses arise from its PRNG and bit permutation generator. In this respect, we note that our forward secure PRNG does not allow malleability, so it should be hard to obtain a collision in the chosen IV model better than with birthday probability. This should prevent collisions on the generated permutations. Besides, the Knuth shuffle [41] (or Fisher-Yates shuffle) is an algorithm allowing to generate a random permutation on a finite set. This algorithm has the interesting property of giving the same probability to all permutations if used with a random number generator. As a result, we expect that any deviation between a bit permutation based on a Knuth shuffle fed with the PRNG will be hard to exploit by an adversary. Our motivation for this assumption is twofold. First, it allows us to focus on whether the filter permutator construction is theoretically sound. Second, if such a choice was leading to an exploitable weakness, it remains possible to build a pseudorandom permutation based on standard cryptographic constructions [45].

Remark 3. Since the permutation generation part of FLIP has only birthday security (with respect to the size of the PRNG), it implies that it is only secure up to 2^{64} PRNG outputs when implemented with the AES-128. Generating more keystream using larger block ciphers should be feasible. However, in view of the novelty of the FLIP instances, our claims are only made for this limited (birthday) data complexity so far, which should not be limiting for the intended FHE applications. We leave the investigation of their security against attacks using larger data complexities as a scope for further research. Besides, we note that using a PRNG based on a tweakable block cipher [44] (where a part of the larger IV would be used as tweak) could be an interesting track to reduce the impact of a collision on the PRNG output in the known IV model, which we also leave as an open research direction.

3.4 Security Analysis

Since the filter permutator shares similarities with a filter generator, it is natural to start our investigations with the typical attacks considered against such types of stream ciphers. For this purpose, we next study the applicability of algebraic attacks and correlation attacks, together with more specialized cryptanalyses that have been considered against stream ciphers. Note that the attacks considered in the rest of this section frequently require to solve systems of equations and to implement a Gaussian reduction. Our complexity estimations will consider Strassen's algorithm for this purpose and assume $\omega = \log 7$ to be the exponent in a Gaussian reduction. Admittedly, approaches based on Gröbner bases [27] or taking advantage of the sparsity of the matrices [59] could lead to even faster algorithms. We ignore them for simplicity in these preliminary investigations. Note also that we only claim security in the single-key setting.

Algebraic Attacks were first introduced by Courtois and Meier in [18] and applied to the stream cipher Toyocrypt. Their main idea is to build an over-defined system of equations with the initial state of the LFSR as unknown, and to solve this system with Gaussian elimination. More precisely, by using a nonzero function g such that both g and $h = gF$ have low algebraic degree, an adversary is able to obtain T equations with monomials of degree at most $AI(f)$. It is easily shown that g can be taken equal to the annihilator of F or of $F \oplus 1$, *i.e.* such that $gF = 0$ or $g(F \oplus 1) = 0$. After a linearisation step, the adversary obtains a system of T equations in $D = \sum_{i=0}^{AI(F)} \binom{N}{i}$ variables. Therefore, the time complexity of the algebraic attack is $\mathcal{O}(D^\omega)$, that is, $\mathcal{O}(N^{\omega AI(f)})$.

Fast Algebraic Attacks are a variation of the previous algebraic attacks introduced by Courtois at Crypto 2003 [17]. Considering the relation $gF = h$, their goal is to find and use functions g of low algebraic degree e , possibly smaller than $AI(f)$, and h of low but possibly larger degree d , and to lower the degree of the resulting equations by an off-line elimination of the monomials of degrees larger than e (several equations being needed to obtain each one with degree at most e). Following [4], this attack can be decomposed into four steps:

1. The search of the polynomials g and h generating a system of $D+E$ equations in $D+E$ unknowns, where $D = \sum_{i=0}^d \binom{N}{i}$ and $E = \sum_{i=0}^e \binom{N}{i}$. This step has a time complexity in $\mathcal{O}(\sum_{i=0}^d \binom{n}{i} + \sum_{i=0}^e \binom{n}{i})^\omega$.
2. The search of linear relations which allow the suppression of the monomials of degree more than e . This step has a time complexity in $\mathcal{O}(D \log^2(D))$.
3. The elimination of monomials of degree larger than e using the Berlekamp-Massey algorithm. This step has a time complexity in $\mathcal{O}(ED \log(D))$.
4. The resolution of the system. This step has a time complexity in $\mathcal{O}(E^\omega)$.

Given the FAI of F , the time complexity of this attack is in $\mathcal{O}(N^{\text{FAI}})$, or more precisely $\mathcal{O}(D \log^2 D + E^2 D + E^\omega)$ (ignoring Step 1 which is trivial for our choice of F).

Correlation Attacks. In their basic versions, correlation attacks try to distinguish the output sequence of a stream cipher from a random one, by exploiting the bias δ of the filtering function. We can easily rule out such attacks by considering a (much) simplified version of filter permutator where the bit permutations P_i 's would be made on two independent permutations P_i^1 and $P_i^{2,3}$ (respectively acting on the $n_1 + 1$ bits of the linear part and the $n_2 + n_3 - 1$ bits of the non-linear part of F). Suppose for simplicity that P_i^1 is kept constant t times, then the output distribution of F has a bias δ and it can be distinguished for the right choice of the $n_1 + 1 = \text{res} + 1$ bits of the linear part. In this case, a correlation attack would have a data complexity of $\mathcal{O}(\delta^{-2})$ and a time complexity

of $\mathcal{O}(2^{\text{res}(F)+1}\delta^{-2})$, with $\delta = \frac{1}{2} - \left(\frac{\text{NL}(F)}{2^N}\right)$. For simplicity, we will consider this

conservative estimation in our following selection of security parameters. Yet, we note that since the permutation P_i of a filter permutator is acting on all the N bits of the filter F , the probability that the linear part of F is kept invariant by the permutations t times is in fact considerably smaller than what is predicted by the resilience.

BKW-like Attack. The BKW algorithm was introduced in [6] as a solution to solve the LPN problem using smart combinations of well chosen vectors and their associated bias. Intuitively, our stream cipher construction simplified as just explained (with two independent permutations P_i^1 and $P_i^{2,3}$ rather than a single one P_i) also shares similarities with this problem. Indeed, we could see the linear part as the parity of an LPN problem and the non-linear one (with a small bias) as a (large) noise. Adapting the BKW algorithm to our setting amounts to XOR some linear parts of F in order to obtain vectors of low Hamming weight, and then to consider a distinguishing attack with the associated bias. Denoting h the target Hamming weight, x the log of the number of XORs and δ the bias, the resulting attack (which can be viewed as an extension of the previous correlation attack) has data complexity $\mathcal{O}(2^h \delta^{-2(x+1)})$ (more details are given in the extended version of this paper).

Higher-Order Correlation Attacks were introduced by Courtois [16] and exploit the so-called XL algorithm. They look for good correlations between F and an approximation g of degree $d > 1$, in order to solve a linearised system based on the values of this approximation. The value ε is defined such that g is equal to F with probability greater than $1 - \varepsilon$. Such attacks have a (conservative) time complexity estimate:

$$\mathcal{O}\left(\binom{N}{D}^\omega (1 - \varepsilon)^{-m}\right), \text{ where } D \geq d \text{ and } m \geq \frac{\binom{N}{D}}{\binom{N}{D-d}}.$$

Guess and Determine Attacks. Note that this section has been motivated by a private communication from Sébastien Duval, Virginie Lallemand and Yann Rotella, of which the details will be available in an upcoming ePrint report [25]. Guess and determine attacks are generic attacks which consist in guessing ℓ bits of the key in order to cancel some monomials. In our context, it allows an adversary to focus on a filtering function restricted to a subset of variables. This weaker function can then be cryptanalyzed, *e.g.* analyzed with the four aforementioned attacks, *i.e.* the algebraic attack, the fast algebraic attack, the correlation/BKW-like attacks and the higher-order correlation attack. The complexity of a guess and determine attack against a function F of N variables is $\min_{\ell} \{2^{\ell} C(F[\ell])\}$ where $F[\ell]$ is a function of $N[\ell]$ variables obtained by fixing ℓ variables of F , $C(F)$ is the complexity of the best of the four attacks considered on the filtering function F and the minimum is taken over all ℓ 's. The case $\ell = 0$ corresponds to attacking the scheme without guess and determine. We next bound the minimal complexity over these four attacks considering the weakest functions obtained by guessing. To do so, we introduce some notations and criteria allowing us to specify the cryptographic properties of Boolean functions obtained by guessing ℓ variables of Boolean functions being direct sums of monomials. As the impact of guessing is most relevant for fast algebraic attacks and CA/BKW-like attacks, we defer the other part of the analysis and extra lemmas to the extended version of this paper.

Definition 16 (Direct Sum Vector). For a boolean function F of N variables obtained as a direct sum of monomials we associate its **direct sum vector** : \mathbf{m}_F of length $k = \deg(F)$: $[m_1, m_2, \dots, m_k]$ such that m_i is the number of monomials of degree i of F and $N = \sum_{i=1}^k im_i$. We define two quantities related to this vector:

- \mathbf{m}_F^* is the number of nonzero values of \mathbf{m}_F .
- $\delta_{\mathbf{m}_F} = \frac{1}{2} - \frac{\text{NL}(F)}{2^N}$.

These notations will be useful to quantify the impact of guessing some bits on the cryptographic properties of a Boolean function obtained by direct sums. \mathbf{m}_F , \mathbf{m}_F^* and $\delta_{\mathbf{m}_F}$ are easily computable from the description of F , the latter can be computed recursively using Lemma 3.

Lemma 7 (Guessing and Direct Sum Vector). For all guessing of $0 \leq \ell \leq N$ variables of a Boolean function F in N variables obtained by direct sums associated with \mathbf{m}_F , we obtain a function $F[\ell]$ in $N[\ell]$ variables associated with $\mathbf{m}_{F[\ell]}$ such that:

1. $\sum_{i=1}^{\deg(F[\ell])} m_i[\ell] \geq (\sum_{i=1}^{\deg(F)} m_i) - \ell$.
2. $\mathbf{m}_{F[\ell]}^* \geq \mathbf{m}_F^* - \lfloor \frac{\ell}{\min_{1 \leq i \leq \deg(F)} m_i} \rfloor$.
3. $\delta_{\mathbf{m}_{F[\ell]}} \leq \delta_{\mathbf{m}_F} 2^{\ell}$.

Hereafter we describe the bounds we have used in order to assess the security of our instances.

Lemma 8 (Guess And Determine & Fast Algebraic Attacks). *Let F be a boolean function in N variables and $C_{GDFAA}(F)$ be the minimum complexity of the Guess And Determine with Fast Algebraic Attacks on F , then:*

$$C_{GDFAA}(F) \geq \min_{0 \leq \ell \leq N} \left[2^\ell \binom{\min N[\ell]}{\mathbf{m}_F^*[\ell]} \log^2 \binom{\min N[\ell]}{\mathbf{m}_F^*[\ell]} + (\min N[\ell])^2 \binom{\min N[\ell]}{\mathbf{m}_F^*[\ell]} + (\min N[\ell])^\omega \right],$$

where $\mathbf{m}_F^*[\ell] = \mathbf{m}_F^* - \lfloor \frac{\ell}{\min_{i \in [\deg(F)]} m_i} \rfloor$.

Lemma 9 (Guess and Determine & CA/BKW-like Attacks). *Let F be a boolean function in N variables and $C_{GDCA/BKW}(F)$ be the minimum complexity of the Guess And Determine with Correlation/BKW Attacks on F , then:*

$$C_{GDCA/BKW}(F) \geq \min_{0 \leq \ell \leq N} \{2^{-\ell} \delta_{\mathbf{m}_F}^{-2}\}.$$

Other Attacks. Besides the previous attacks that will be taken into account quantitatively when selecting our concrete instances of FLIP designs, we also investigated the following other cryptanalyses. First, *fast correlation attacks* were introduced by Meier and Staffelbach at Eurocrypt 1988 [48]. A recent survey can be found in [47]. The attack is divided into two phases. The first one aims at looking for relations between the output bits a_i of the LFSR to generate a system of parity-check equations. The second one uses a fast decoding algorithm (e.g. the belief propagation algorithm) in order to decode the words of the code $z_i = F(a_i)$ satisfying the previous relations, where the channel has an error probability $p = \frac{NL(F)}{2^N}$. The working principles of this attack are quite similar to the ones of the previously mentioned correlation attacks and BKW-like attacks. So we assume that the previous (conservative) complexity estimates rule out this variation as well. Besides, note that intuitively, the belief propagation algorithm is best suited to the decoding of low-density parities, which is what our construction (and the LPN problem) typically avoids.

Second, *weak keys* (i.e. keys of low or high hamming weights) can produce a keystream sufficiently biased to determine this hamming weight, and then to recover the key among the small amount of possible ones. The complexity of such attacks can be computed from the resiliency of F . However, since our N parameter will typically be significantly larger than the bit-security of our filter permutator instances, we suggest to restrict the key space to keys of Hamming weight $N/2$ to rule out this concern. For this purpose, master keys can simply be generated by applying a first (secret) random permutation to any stream with Hamming weight $N/2$.

Third, *augmented function attacks* are attacks focusing on multiple outputs of the function rather than one. The goal is to find coefficients j_1, \dots, j_r such that a relation between the key and the outputs $s_{i+j_1}, \dots, s_{i+j_r}$ can be exploited. This relation can be a correlation (as explained in [3]) or simply algebraic [28]. In both cases, a prerequisite is that the relation holds on a sufficient number of i . As each bit output by FLIP depends on a different permutation, we believe that there is no exploitable relation between different outputs.

Eventually, *cube attacks* were introduced by Dinur and Shamir at Eurocrypt 2009 [20] as a variant of algebraic attacks taking advantage of the public parameters of a cryptographic protocol (plaintext in block ciphers, IV in stream cipher) in order to generate a system of equations of low degree. However in filter permutator constructions, the only such public parameter is the seed of the PRNG allowing to generate the pseudo-random bit permutations P_i . Since controlling this seed hardly allows any control of the F function’s inputs, such attacks do not seem applicable. A similar observation holds for conditional differential cryptanalysis [39] and for integral/zero-sum distinguishers [8, 40].

3.5 Cautionary Note and Design Tweaks

As already mentioned, all the previous analyzes are based on standard cryptanalysis and design tools. In particular, the security of our FLIP designs is based on properties of Boolean functions that are generally computed assuming a uniform input distribution. Yet, for filter permutators this condition is not strictly respected since the Hamming weight of the key register is fixed (we decided to set it to $N/2$ in order to avoid weak keys, but even without this condition, it would be fixed to an unknown value). This means the input distribution of our linear, quadratic and triangle functions is not uniform. We verified experimentally that the output of FLIP is sufficiently balanced despite this non-uniformity. More precisely, we could not detect biases larger than $2^{\frac{q}{2}}$ when generating 2^q bits of keystream (based on small-scale experiments with $q = 32$). But we did not study the impact of this non-uniformity for other attacks, which we leave as an important research scope, both from the cryptanalysis and the Boolean functions points-of-view.

Note that in case the filter permutator of Sect. 3.1 turns out to have weaknesses specifically due to the imbalanced F function’s inputs, there are tweaks that could be used to mitigate their impact. The simplest one is to apply a public whitening to the input bits of the non-linear parts of F (using additional public PRNG outputs), which has no impact on the homomorphic capacity. The adversary could then bias the F function’s inputs based on his knowledge of the whitening bits, but to a lower extent than with our fixed Hamming weight keys. Alternatively, one could add a (more or less complex) linear layer before the non-linear part of F , which would then make the filter permutator conceptually more similar to filter generators, and (at least for certain layers) only imply moderate cost from the FHE point-of-view.

3.6 80- & 128-bit Security Instances

We selected a few instances aiming at 80- and 128-bit security based on the previous bounds, leading to the attack complexities listed in Table 1, where $\text{FLIP}(n_1, n_2, {}^{nb}\Delta^k)$ denotes the instantiation of FLIP with linear part of n_1 bits, quadratic part of n_2 bits and nb triangular functions of degree k . These instances are naturally contrasted. On the one hand, the bounds taken are conservative with respect to the attacks considered: if these attacks were the best ones, more

aggressive instances could be proposed (*e.g.* in order to reduce the key size). On the other hand, filter permutators are based on non-standard design principles, and our security analysis is only preliminary, which naturally suggests the need of security margins. Overall, we believe the proposed instances are a reasonable trade-off between efficiency and security based on our current understanding of filter permutators, and therefore are a good target for further investigations.

Table 1. Attack complexities in function of n_1 , n_2 and ${}^{nb}\Delta^k$. AA stands for algebraic attacks, FAA stands for fast algebraic attacks, CA/BKW stands for correlation or BKW-like attacks, HOC stands for higher-order correlation attacks and ℓ stands for the number of bits guessed leading to the best complexity for guess and determine attacks. For the CA/BKW column, we reported the minimum complexity between the correlation and BKW-like attack. Eventually, λ stands for the security parameter of F and is simply taken as the minimum between AA, FAA, CA/BKW and HOC.

Instance	N	AA	ℓ	FAA	l	CA/BKW	ℓ	HOC	ℓ	λ
FLIP(42, 128, ${}^8\Delta^9$)	530	95	56	81	0	86	72	94	55	81
FLIP(46, 136, ${}^4\Delta^{15}$)	662	91	52	81	52	80	72	90	48	80
FLIP(82, 224, ${}^8\Delta^{16}$)	1394	156	112	140	40	134	120	155	109	134
FLIP(86, 238, ${}^5\Delta^{23}$)	1704	149	105	137	105	133	124	128	74	128

3.7 Indirect Sums

Before analyzing the FHE properties of filter permutators, we finally suggest FLIP designs based on indirect sums as another interesting topic for evaluation, since they lead to quite different challenges. Namely, the main motivation to use direct sums in the previous sections was the possibility to assess their cryptographic properties based on existing tools. By contrast, filter permutator designs based on indirect sums seem harder to analyze (both for designers and cryptanalysts). This is mainly because in this case, not only the inputs of the Boolean functions vary, but also the Boolean functions themselves. For illustration, we can specify “multi-FLIP” designs, next denoted as b -FLIP designs, such that we compute b instances of FLIP in parallel, each with the same filtering function but with different permutations, and then to XOR the b computed bits in order to produce a keystream bit. We conjecture that such b -FLIP designs could lead to secure stream ciphers with smaller states, and suggest 10-FLIP(10, 20, ${}^1\Delta^{20}$) and 15-FLIP(15, 30, ${}^1\Delta^{30}$) as exemplary instances for 80- and 128-bit security.

4 Application to FHE

4.1 80- & 128-bit Security Parameters

For the security parameters choices, we follow the analysis of Lindner and Peikert [43] for the hardness of LWE and RLWE, considering distinguishing and decoding

attacks using BKZ [14,55]. We assume that the distribution χ in the considered LWE instances is the discrete Gaussian distribution with mean 0 and standard deviation σ . First we compute the best root Hermite factor δ of a basis (see [29]) computable with complexity 2^λ from the conservative lower bound of [43]:

$$\log(\delta) = 1.8/(110 + \lambda). \tag{1}$$

The distinguishing attack described in [43,50,54] is successful with advantage ε by finding vectors of length $\alpha \frac{q}{\sigma}$ with $\alpha = \sqrt{\ln(1/\varepsilon)/\pi}$. The length of the shortest vector that can be computed is $2^{2\sqrt{n \log q \log \delta}}$, leading to the inequality:

$$\alpha \frac{q}{\sigma} < 2^{2\sqrt{n \log q \log \delta}}. \tag{2}$$

Given $\sigma \geq 2\sqrt{n}$ from Regev’s reduction [53], we can choose parameters for n and q matching Eq. (2) for the considered security parameter λ . The parameters we select for our application are summarized in Table 2.

Table 2. (R)LWE parameters used in our applications.

Security λ	n	$\log q$
80	256	80
128	512	120

4.2 Noise Analysis

Considering our framework of Fig. 1, Claude has at its disposal the homomorphic encryption of the symmetric key $\mathbf{C}^H(\text{sk}_i^S)$, the homomorphic public key pk^H and the symmetric encrypted messages $\mathbf{C}^S(m_i)$. He has to perform the homomorphic evaluation of the symmetric decryption circuit, *i.e.* to perform homomorphic operations on the ciphertexts $\mathbf{C}^H(\text{sk}_i^S)$ in order to get $\mathbf{C}^H(m_i)$, the homomorphic encryption of m_i . In this section, we study the error growth in these ciphertexts after the application of the homomorphic operations. As we are considering SWHE, we need to control the magnitude of the error and keep it below a critical level to ensure the correctness of a final ciphertext. This noise management is crucial for the applications, it is directly linked with the quantity of computation that the server can do for the client. We now study the error growth stemming from the homomorphic evaluation of FLIP. In this case, all the ciphertexts used by the server in the computation step will have a same starting error. The knowledge of this starting error (defined by some parameter) and its growth for additions and multiplications (in a chosen order) is enough to determine the amount of computation that can be performed correctly by the server.

In the remainder of this section we proceed in three steps. First we recall the error growth implied by the $H.\text{Add}$ and $H.\text{Mul}$ operations: for GSW-like HE it has already been done in [2,10,24,32,36]. As our homomorphic encryption schemes

are slightly differently written to fit our applications (batched version to perform in parallel the same computations, generic notations for various frameworks), we give these error growth with our notations for completeness and consistency of the paper. Then we analyse the error for a sub-case of homomorphic product, namely *H.Comb*, which gives a practical tool to study the error growth in FLIP. As the asymmetric property of GSW multiplication and plaintext norm have been pointed out relatively to the error growth, we consider important to focus on both when analysing this error metric. Considering *H.Comb* types of operations is therefore suited to be consistent with this metric and is very important for practical purpose (in term of real life applications). Finally we analyse the error in a ciphertext output by FLIP and study some optimizations to reduce the noise growth further.

Error Growth in *H.Add* and *H.Mul*. We first need to evaluate the error growth of the basic homomorphic operations, the addition and the multiplication of ciphertexts. We use the analysis of [2] based on subgaussian distributions to study the error growth in these homomorphic operations. From a coefficient or a vector following a subgaussian distribution of parameter σ , we can bound its norm with overwhelming probability and then study the evolution of this parameter while performing the homomorphic operations. Hence we can bound the final error to ensure correctness.

For simplicity we use two notations arising from the error growth depending on the arithmetic of the underlying ring of the two schemes, γ the expansion factor (see [9]) and $Norm(m_j)$ such that:

- Batched GSW: $\gamma = 1$ and $Norm(m_j) = |m_j|$ (arithmetic in \mathbb{Z}).
- Ring GSW: $\gamma = n$ and $Norm(m_j) = \|m_j\|_2$ (arithmetic in R).

Lemma 10 (*H.Add Error Growth*). *Suppose C_i for $1 \leq i \leq k$ are ciphertexts of a GSW based Homomorphic Encryption scheme with error components e_i of coefficients following a distribution of parameter σ_i . Let $C_f = H.Add(C_i, \text{ for } 1 \leq i \leq k)$ and e_f the related error with subgaussian parameter σ' such that:*

$$\sigma' = \sqrt{\sum_{i=1}^k \sigma_i^2} \quad \text{or} \quad \sigma' = \sigma\sqrt{k} \text{ if } \sigma_i = \sigma, \forall i \in [k].$$

Lemma 11 (*H.Mul Error Growth*). *Suppose C_i for $1 \leq i \leq k$ are ciphertexts of a GSW based Homomorphic Encryption scheme with error components e_i , of coefficients following a subgaussian distribution of parameter σ_i , and plaintext m_i . C_f is the result of a multiplicative homomorphic chain such that:*

$$C_f = H.Mul(C_1, H.Mul(C_2, H.Mul(\dots, H.Mul(C_k, \mathbf{G}))),$$

and e_f the corresponding error with subgaussian parameter σ' such that:

$$\sigma' = \mathcal{O} \left(\sqrt{N\gamma} \sqrt{\sigma_1^2 + \sum_{i=2}^k (\sigma_i \prod_{j=1}^{i-1} Norm(m_j))^2} \right).$$

Lemmas 10 and 11 are proven in the extended version of this paper.

Error Growth in $H.Comb$. For the sake of clarity, we formalize hereafter the comb homomorphic product $H.Comb$ and the quantity σ_{comb} which stands for the subgaussian parameter. We study the error growth of $H.Comb$ as we will use it as a tool for the error growth analysis of FLIP.

Definition 17 (Homomorphic Comb $H.Comb$). Let C_1, \dots, C_k be k ciphertexts of a GSW based Homomorphic Encryption scheme with error coefficients from independent distributions with same subgaussian parameter σ . We define $H.Comb(y, \sigma, c, k) = H.Mul(C_1, \dots, C_k, G)$ where:

- $y = \sqrt{N}\gamma$ is a constant depending on the ring,
- $c = \max_{1 \leq i \leq k} (Norm(m_i))$ is a constant which depends on the plaintexts,

and $C_{comb} = H.Comb(y, \sigma, c, k)$ as error components following a subgaussian distribution of parameter $\mathcal{O}(\sigma_{comb})$.

Lemma 12 (σ_{comb} Quantity). Let C_1, \dots, C_k be k ciphertexts of a GSW based Homomorphic Encryption scheme with same error parameter σ and $C_{comb} = H.Comb(y, \sigma, c, k)$. Then we have:

$$\sigma_{comb}(y, \sigma, c, k) = y\sigma c_k, \quad \text{where } c_k = \sqrt{\sum_{i=0}^{k-1} c^{2i}}.$$

Proof. Thanks to Lemma 11 we obtain:

$$\begin{aligned} \sigma_{comb} &= \sqrt{N}\gamma \sqrt{\sigma^2 + \sum_{i=2}^k (\sigma \prod_{j=1}^{i-1} Norm(m_j))^2}, \\ \sigma_{comb} &= y \sqrt{\sigma^2 + \sum_{i=2}^k (\sigma c^{i-1})^2}, \\ \sigma_{comb} &= y\sigma \sqrt{\sum_{i=1}^k (c^{i-1})^2}, \\ \sigma_{comb} &= y\sigma c_k. \end{aligned}$$

□

The compatibility of this comb structure with the asymmetric multiplicative error growth property of GSW enables us to easily quantify the error in our construction, with a better accuracy than computing the multiplicative depth. In order to minimize the quantity σ_{comb} , we choose the plaintext space such that $c = 1$ for freshly generated ciphertexts. The resulting $\sigma_{comb}(y, \sigma, 1, k)$ quantity is therefore $y\sigma\sqrt{k}$, growing less than linearly in the number of ciphertexts. Fixing the constant c to be 1 is usual with FHE. As we mostly consider Boolean circuits, it is usual to use plaintexts in $\{-1, 0, 1\}$ to encrypt bits, leading to $c = 1$ and therefore $c_k = \sqrt{k}$.

Error Growth in FLIP. In the previous paragraphs, we have evaluated the error growth in the basic homomorphic operations $H.Add$, $H.Mul$ and $H.Comb$. We will use them as building blocks in order to evaluate the error growth in the homomorphic evaluation of FLIP. Coming back to the framework of Fig. 1, the

error in the ciphertexts $\mathbf{C}^H(m_i)$ is of major importance as it will determine the possible number of homomorphic computations f that Claude is able to perform.

The main feature of the filter permutator model, considering FHE settings, is that it allows to handle ciphertexts having the same error level, whatever the number of output bits. Consequently all ciphertexts obtained by FLIP evaluation will have the same constant and small amount of noise and will be considered as fresh start for more computation.

Evaluating homomorphically the FLIP decryption (*resp.* encryption) algorithm consists in applying three steps of homomorphic operations on the ciphertexts $\mathbf{C}^H(\text{sk}_i^S)$ in our application framework, each one encoding one bit of the key register. For each ciphertext bit, these steps are: a (bit) permutation, the application of the filtering function F and a XOR with the ciphertext (*resp.* plaintext). The (bit) permutation consists only in a public rearrangement of the key ciphertexts, leading to a noise-free operation. The last XOR is done with a freshly encrypted bit. Hence the error growth depends mostly on the homomorphic evaluation of F .

As $H.\text{Dec}$ outputs quantities modulus 2, we can evaluate the XORs of F by $H.\text{Add}$ and the ANDs by $H.\text{Mul}$. We then determine the subgaussian parameter of the error of a ciphertext from the homomorphic evaluation of F . For a given encrypted key, this parameter will be the same for every homomorphic evaluation of FLIP and is computed from σ_{comb} .

Lemma 13 (Error Growth Evaluating F). *Let F be the FLIP filtering function in N variables defined in Sect. 3.3. Assume that \mathbf{C}_i for $0 \leq i \leq N - 1$ are ciphertexts of a GSW HE scheme with same subgaussian parameter σ and $c = 1$. We define $\mathbf{C}_F = H.\text{Eval}(F, \mathbf{C}_i)$ the output of the homomorphic evaluation of the ciphertexts \mathbf{C}_i 's along the circuit F . Then the error parameter σ' is:*

$$\sigma' = \mathcal{O}\left(\sigma\sqrt{n_1 + y^2(n_2 + n_3)}\right) \approx \mathcal{O}\left(\sigma y\sqrt{N}\right).$$

Proof. We first evaluate the noise brought by F for each of its components $L_{n_1}, Q_{n_2}, {}^{nb}\Delta^k$, defining the respective ciphertexts $\mathbf{C}_{L_{n_1}}, \mathbf{C}_{Q_{n_2}}, \mathbf{C}_{T_k}$ (the last one standing for one triangle only) and the subgaussian parameter of the respective error distributions (of the components of the error vectors) $\sigma_{L_{n_1}}, \sigma_{Q_{n_2}}, \sigma_{T_k}$:

- L_{n_1} : $\mathbf{C}_{L_{n_1}} = H.\text{Eval}(L_{n_1}, \mathbf{C}_0, \dots, \mathbf{C}_{n_1-1}) = H.\text{Add}(\mathbf{C}_0, \dots, \mathbf{C}_{n_1-1})$ then $\sigma_{L_{n_1}} = \sigma\sqrt{n_1}$.
- Q_{n_2} : $\mathbf{C}_{Q_{n_2}} = H.\text{Add}(H.\text{Mul}(\mathbf{C}_{n_1+2j}, \mathbf{C}_{n_1+2j+1}, \mathbf{G}))$ for $0 \leq j \leq n_2$.
 $H.\text{Mul}(\mathbf{C}_{n_1+2j}, \mathbf{C}_{n_1+2j+1}, \mathbf{G}) = H.\text{Comb}(y, \sigma, 1, 2)$ has subgaussian parameter $\mathcal{O}(\sigma_{comb}(y, \sigma, 1, 2)) = \mathcal{O}(y\sigma\sqrt{2})$ for $0 \leq j \leq n_2$.
 Then $\sigma_{Q_{n_2}} = \mathcal{O}(y\sigma\sqrt{2}\sqrt{\frac{n_2}{2}}) = \mathcal{O}(y\sigma\sqrt{n_2})$.
- T_k : $\mathbf{C}_{T_k} = H.\text{Add}(H.\text{Mul}(\mathbf{C}_{n_1+n_2+j+(i-1)(i-2)/2}; 1 \leq j \leq i); 1 \leq i \leq k)$.
 $\mathbf{C}_{T_k} = H.\text{Add}(H.\text{Comb}(y, \sigma, 1, i), 1 \leq i \leq k)$.
 then $\sigma_{T_k} = \mathcal{O}(\sqrt{\sum_{i=1}^k (y\sigma\sqrt{i})^2}) = \mathcal{O}(y\sigma\sqrt{\frac{k(k+1)}{2}})$.
 As ${}^{nb}\Delta^k$ is obtained by adding nb independent triangles, we get:
 $\mathbf{C}_{{}^{nb}\Delta^k} = H.\text{Add}(\mathbf{C}_{T_k, i}, 1 \leq i \leq nb)$,
 and $\sigma_{{}^{nb}\Delta^k} = \mathcal{O}(y\sigma\sqrt{nb}\sqrt{\frac{k(k+1)}{2}}) = \mathcal{O}(y\sigma\sqrt{n_3})$.

By Pythagorean additivity the subgaussian parameter of \mathbf{C}_F is finally:

$$\sigma' = \mathcal{O}(\sqrt{(\sigma\sqrt{n_1})^2 + (y\sigma\sqrt{n_2})^2 + (y\sigma\sqrt{n_3})^2}) = \mathcal{O}(\sigma\sqrt{n_1 + y^2(n_2 + n_3)}). \quad \square$$

Optimizations. The particular error growth in GSW Homomorphic Encryption enables to use more optimizations to reduce the error norm and perform more operations without increasing the parameter’s size. The error growth in $H.Comb$ depends on the quantity c_k derived from bounds on norms of the plaintexts; these quantities can be reduced using negative numbers. A typical example is in the LWE-based scheme to use $m \in \{-1, 0, 1\}$ rather than $\{0, 1\}$; the c_k quantity is the same and in average the sums in \mathbb{Z} are smaller. Then the norm $|\sum m_i|$ is smaller which is important when multiplying. Conserving this norm as low as possible gives better bounds and c_k coefficients, leading to smaller noise when performing distinct levels of operations. An equivalent way of minimizing the error growth is to still use $\mathcal{M} = \{0, 1\}$ but with $H.Add(\mathbf{C}_1, \mathbf{C}_2) = \mathbf{C}_1 \pm \mathbf{C}_2$. This homomorphic addition is still correct because:

$$\mathbf{S} - \mathbf{C}_2 = -\mathbf{E}'_2 - \begin{pmatrix} m_{2,1} \cdot \mathbf{s}_1^\top \\ \vdots \\ m_{2,r} \cdot \mathbf{s}_r^\top \end{pmatrix} \mathbf{G} = \mathbf{E}''_2 + \begin{pmatrix} -m_{2,1} \cdot \mathbf{s}_1^\top \\ \vdots \\ -m_{2,r} \cdot \mathbf{s}_r^\top \end{pmatrix},$$

where the coefficients in \mathbf{E}''_2 rows follow distribution of same subgaussian parameter as the one in \mathbf{E}'_2 by homogeneity and $-m = m \pmod 2$.

4.3 Concrete Results

Contrary to other works published in the context of symmetric encryption schemes for efficient FHE [1, 11, 31], our primary focus is not on the performances (see SHIELD [38] for efficient implementation of Ring-GSW) but rather on the error growth. As pointed out in [11], in most of these previous works, after the decryption process the noise inside the ciphertexts was too high to perform any other operation on them, whereas it is the main motivation for a practical use of FHE.

In this section, we consequently provide experimental results about this error growth in the ciphertexts after different operations evaluated on the Ring GSW scheme. As the link between subgaussian parameter, ciphertext error and homomorphic computation is not direct, we make some choices for representing these results focusing on giving intuition on how the error behaves.

The choice of the Ring GSW setting rather than Batched GSW is for convenience. It allows to deal with smaller matrices and faster evaluations, providing the same confirmation on the heuristic error growth. We give the parameters n and ℓ defining the polynomial ring and fix $\sigma = 2\lceil\sqrt{n}\rceil$ for the error distribution.

An efficient way of measuring the error growth within the ciphertexts is to compute the difference by applying the rounding $\lfloor \cdot \rfloor_2$ in $H.Dec$ between various

ciphertexts with known plaintext. This difference (for each polynomial coefficient or vector component) corresponds to the amount of noise contained in this ciphertext. The correctness requires this quantity to be inferior to $2^{\ell-2}$. Then, considering its logarithm in base 2, it enables to have an intuitive and practical measure of the ciphertext noise: this quantity grows with the homomorphic operations until this log equals $\ell - 2$. Concretely, in our experiments we encrypt polynomials being $m = 0$ or $m = 1$, compute on the constant coefficient the quantity $e = |(\langle \mathbf{s}, \mathbf{c}_\ell \rangle - m2^{\ell-1}) \bmod q|$, and give its logarithm. We give another quantity in order to provide intuition about the homomorphic computation possibilities over the ciphertexts, by simply computing a percentage of the actual level of noise relatively to the maximal level $\ell - 2$.

Remark 4. The quantity exhibited by our measures is roughly the subgaussian parameter of the distribution of the error contained in the ciphertexts. Considering the simpler case of a real Gaussian distribution $\mathcal{N}(0, \sigma^2)$, the difference that we compute then follows a half normal distribution with mean $\sigma \frac{\sqrt{2}}{\sqrt{\pi}}$.

We based our prototype implementation on the NTL library combined with GMP and the discrete gaussian sampler of BLISS [23]. We report in Table 3 experimental results on the error growth for different RLWE and FLIP parameters, based on an average over a hundred of samples.

The results confirm the quasi-additive error growth when considering the specific metric of GSW given by the asymptotic bounds. The main conclusion of these results is that the error inside the ciphertexts after a homomorphic evaluation of FLIP is of the same order of magnitude as the one after a multiplication. The only difference between these noise increases is a term provided by the root of the symmetric key register size, that is linear in λ . Therefore, with the FLIP construction the error growth is roughly the basic multiplicative error growth of two ciphertexts. Hence, we conclude that filter permutators as FLIP release the bottleneck of evaluating symmetric decryption, and lead the further improvement of the calculus delegation framework to depend overwhelmingly on improvements of the homomorphic operations.

Table 3. Experimental error growth of Ring-GSW. Fresh, *H.Add*, *H.Mul* and *H.Eval(FLIP)* respectively stands for the noise e measure after a fresh homomorphic encryption, the homomorphic addition of two fresh ciphertexts, the homomorphic multiplication of two fresh ciphertexts and the homomorphic evaluation of FLIP on fresh ciphertexts. The first value is the log of the error e inside the corresponding ciphertexts and the percentage represents the proportion of the noise with respect to the capacity of decryption (*i.e.* $\ell - 2$).

Ring (n, ℓ)		FLIP	Fresh		<i>H.Add</i>		<i>H.Mul</i>		<i>H.Eval(FLIP)</i>	
			log e	%	log e	%	log e	%	log e	%
256	80	(42, 128, $^8\Delta^9$)	13,07	17%	13,96	18%	19,82	25%	24,71	31%
512	120	(82, 224, $^8\Delta^{16}$)	14,68	12%	15,14	13%	23,27	20%	28,77	24%

4.4 Performances for 2nd-Generation Schemes

Despite our new constructions are primarily designed for 3rd-generation FHE, a look at Table 4 suggests that also from the multiplicative depth point of view, FLIP instances bring good results compared to their natural competitors such as LowMC [1] and Trivium/Kreyvium [11]. In Trivium/Kreyvium, the multiplicative depth of the decryption circuit is at most 13, while the LowMC family has a record multiplicative depth of 11 which is still larger than our FLIP instances. For completeness, we finally investigated the performances of some instances of FLIP for 2nd-generation FHE schemes using HELib, as reported in Table 5, where the latency is the amount of time (in seconds) needed to homomorphically decrypt ($N_b \cdot \text{Number of Slots}$) bits, and the throughput is calculated as $(N_b \cdot \text{Number of Slots} \cdot 60) / \text{latency}$. As in [11], we have considered two noise levels: a first one that does not allow any other operations on the ciphertexts, and a second one where we allow operations of multiplicative depth up to 7. Note that the (max) parenthesis in the N_b column recalls that for Trivium/Kreyvium, the homomorphic capacity decreases with the number of keystream bits generated, which therefore bounds the number of such bits before re-keying. We observe that for 80-bit security, our instances outperform the ones based on Trivium. As for 128-bit security, the gap between our instances and Kreyvium is limited (despite the larger state of FLIP), and LowMC has better throughput in this context. Note also that our results correspond to the evaluation of the F function of FLIP (we verified that the time needed to generate the permutations only marginally affects the overall performances of homomorphic FLIP evaluations). We finally mention that these results should certainly not be viewed as strict comparisons, since obtained on different computers and for relatively new ciphers for which we have limited understanding of the security margins (especially

Table 4. Multiplicative depth of different symmetric ciphers.

Algorithm	Reference	Multiplicative depth	Security
SIMON-32/64	[42]	32	64
Trivium-12	[11]	12	80
Trivium-13	[11]	13	80
LowMc-80	[1]	11	80
FLIP(42, 128, $^8\Delta^9$)	This work	$\lceil \log 9 \rceil = 4$	80
AES-128	[15, 31]	40	128
SIMON-64/128	[42]	44	128
Prince	[22]	24	128
Kreyvium-12	[11]	12	128
Kreyvium-13	[11]	13	128
LowMc-128	[1]	12	128
FLIP(82, 224, $^8\Delta^{16}$)	This work	$\lceil \log 16 \rceil = 4$	128

Table 5. Timings of the homomorphic evaluation of several instances of the Boolean function of FLIP using HELib on an Intel Core i7-3770. The other results are taken from [11]. L and Number of Slots are HELib parameters which stand respectively for the level of noise and the number of bits packed in one ciphertext. (Nb * Number of Slots) corresponds to the number of decrypted bits.

Algorithm	Security	Nb	L	Number of Slots	Latency (sec)	Throughput (bits/min)
Trivium-12	80	45 (max)	12	600	1417.4	1143.0
	80	45 (max)	19	720	4420.3	439.8
Trivium-13	80	136 (max)	13	600	3650.3	1341.3
	80	136 (max)	20	720	11379.7	516.3
Kreyvium-12	128	42 (max)	12	504	1715.0	740.5
	128	42 (max)	19	756	4956.0	384.4
Kreyvium-13	128	124 (max)	13	682	3987.2	1272.6
	128	124 (max)	20	420	12450.8	286.8
LowMC-128	$? \leq 128$	256	13	682	3368.8	3109.6
	$? \leq 128$	256	20	480	9977.1	739.0
FLIP(42, 128, $^8\Delta^9$)	80	1	5	378	4.72	4805.08
	80	1	12	600	17.39	2070.16
FLIP(82, 224, $^8\Delta^{16}$)	128	1	6	630	14.53	2601.51
	128	1	13	720	102.51	421.42

for LowMC [19,21] and FLIP). So they should mainly be seen as an indication that besides their excellent features from the FHE capacity point-of-view, filter permutators inherently have good properties for efficient 2nd-generation FHE implementations as well.

5 Conclusions and Open Problems

In the context of our Homomorphic Encryption - Symmetric Encryption framework, where most of the computations are delegated to a server, we have designed a symmetric encryption scheme which fits the FHE settings, with as main goal to get the homomorphic evaluation of the symmetric decryption circuit as cheap as possible, with respect to the error growth. In particular the error growth obtained by our construction, only one level of multiplication considering the metric of third generation FHE, achieves the lowest bound we can get with a secure symmetric encryption scheme. The use of zero-noise operations as permutations enables us to combine the advantages of block ciphers and stream ciphers evaluation, namely constant noise on the one hand and starting low noise on the other hand. As a result, the homomorphic evaluation of filter permutators can be made insignificant relatively to a complete FHE framework.

The general construction of our encryption scheme – *i.e.* the filter permutator – and its FLIP instances are admittedly provocative. As a result, we believe an important contribution of this paper is to open a wide design space of symmetric constructions to investigate, ranging from the very efficient solutions we suggest to more classical stream ciphers such as filter generators. Such a design space leads to various interesting directions for further research. Overall, the main question raised by filter permutators is whether it is possible to build a secure symmetric encryption scheme with aggressively reduced algebraic degree. Such a question naturally relates to several more concrete problems. First, and probably most importantly, additional cryptanalysis is needed in view of the non-standard design principles exploited in filter permutators. It typically includes algebraic attack taking advantage of the sparsity of their systems of equations, attacks exploiting the imbalances at the input of the filter, and the possibility to exploit chosen IVs to improve those attacks. Second, our analyses also raise interesting problems in the field of Boolean functions, *e.g.* the analysis of such functions with non-uniform input distributions and the investigation of the best fixed degree approximations of a Boolean function (which is needed in our study of higher-order correlation attacks). More directly related to the FLIP instances, it would also be interesting to refine our security analyses, with a stronger focus on the attacks data complexity, and to evaluate whether instances with smaller key register could be sufficiently secure. In case of new cryptanalysis results, the design tweaks we suggest in the paper are yet another interesting research path. Eventually, and from the FHE application point-of-view, optimizing the implementations of filter permutators, *e.g.* by taking advantage of parallel computing clusters that we did not exploit so far, would be useful in order to evaluate their applicability to real-world scenarii.

Acknowledgements. We are highly grateful to Sébastien Duval, Virginie Lallemand and Yann Rotella for sharing their ideas about guess and determine attacks before the publication of this paper, which allowed us to modify the instances of FLIP accordingly. We are also indebted to Anne Canteaut for numerous useful suggestions about the design of filter permutators, and for putting forward some important open problems they raise. Finally, we would like to thank Thierry Berger, Sergiu Carpov, Rafaël Delpino, Malika Izabachene, Nicky Mouha, Thomas Prest and Renaud Sirdey for their feedback about early (and less early) versions of this paper. This work was funded in parts by the H2020 ICT COST CryptoAction, by the H2020 ICT Project SAFECrypto, by the H2020 ERC Staring Grant CRASH and by the INNOVIRIS SCAUT project. François-Xavier Standaert is a research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

References

1. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: *Advances in Cryptology - EUROCRYPT 2015–34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Sofia, Bulgaria, April 26–30, 2015, Proceedings, Part I. pp. 430–454 (2015)
2. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference*, Santa Barbara, CA, USA, August 17–21, 2014, Proceedings, Part I, pp. 297–314 (2014)
3. Anderson, R.J.: Searching for the optimum correlation attack. In: *Fast Software Encryption: Second International Workshop*. Leuven, Belgium, 14–16 December 1994, Proceedings, pp. 137–143 (1994)
4. Armknecht, F., Carlet, C., Gaborit, P., Künzli, S., Meier, W., Ruatta, O.: Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, pp. 147–164. Springer, Heidelberg (2006)
5. Bellare, M., Yee, B.S.: Forward-security in private-key cryptography. *IACR Cryptology ePrint Archive* 2001, 35 (2001)
6. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM* **50**(4), 506–519 (2003)
7. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo random bits. *SIAM J. Comput.* **13**(4), 850–864 (1984)
8. Boura, C., Canteaut, A.: Zero-sum distinguishers for iterated permutations and application to keccak-f and hamsi-256. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) *Selected Areas in Cryptography*. LNCS, vol. 6544, pp. 1–17. Springer, Heidelberg (2011)
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: *Innovations in Theoretical Computer Science 2012*, Cambridge, MA, USA, January 8–10, 2012, pp. 309–325 (2012)
10. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: *Innovations in Theoretical Computer Science, ITCS 2014*, Princeton, NJ, USA, January 12–14, 2014, pp. 1–12 (2014)
11. Canteaut, A., Carпов, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext. *IACR Cryptology ePrint Archive* 2015, 113 (2015)
12. Carlet, C.: *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, chap. Boolean Functions for Cryptography and Error Correcting Codes, pp. 257–397 (2010)
13. Carlet, C., Tang, D.: Enhanced Boolean functions suitable for the filter model of pseudo-random generator. *Des. Codes Crypt.* **76**(3), 571–587 (2015)
14. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Wang, X., Lee, D.H. (eds.) *ASIACRYPT 2011*. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
15. Coron, J.-S., Lepoint, T., Tibouchi, M.: Scale-invariant fully homomorphic encryption over the integers. In: Krawczyk, H. (ed.) *PKC 2014*. LNCS, vol. 8383, pp. 311–328. Springer, Heidelberg (2014)
16. Courtois, N.T.: Higher order correlation attacks, XL algorithm and cryptanalysis of toyocrypt. In: Lee, P.J., Lim, C.H. (eds.) *ICISC 2002*. LNCS, vol. 2587, pp. 182–199. Springer, Heidelberg (2003)

17. Courtois, N.T.: Fast algebraic attacks on stream ciphers with linear feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003)
18. Courtois, N.T., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
19. Dinur, I., Liu, Y., Meier, W., Wang, Q.: Optimized interpolation attacks on lowmc. IACR Cryptology ePrint Archive 2015, 418 (2015)
20. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
21. Dobraunig, C., Eichlseder, M., Mendel, F.: Higher-order cryptanalysis of lowmc. IACR Cryptology ePrint Archive 2015, 407 (2015)
22. Doröz, Y., Shahverdi, A., Eisenbarth, T., Sunar, B.: Toward practical homomorphic evaluation of block ciphers using prince. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 208–220. Springer, Heidelberg (2014)
23. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013)
24. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015)
25. Duval, S., Lallemand, V., Rotella, Y.: Cryptanalysis of the FLIP family of stream ciphers. Cryptology ePrint Archive, Report 2016/271 (2016). <http://eprint.iacr.org/>
26. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive 2012, 144 (2012)
27. Faugère, J.C.: A new efficient algorithm for computing grobner bases (f4). *J. Pure Appl. Algebra* **139**(13), 61–88 (1999)
28. Fischer, S.: Algebraic immunity of S-boxes and augmented functions. In: Fischer, S., Meier, W. (eds.) Fast Software Encryption. LNCS, vol. 4593, pp. 366–381. Springer, Heidelberg (2007)
29. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
30. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, pp. 169–178 (2009)
31. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
32. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)
33. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.-X.: Block ciphers that are easier to mask: How far can we go? In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 383–399. Springer, Heidelberg (2013)

34. Grosso, V., Leurent, G., Standaert, F.-X., Varici, K.: LS-Designs: Bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (2015)
35. Halevi, S., Shoup, V.: Algorithms in HELib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 554–571. Springer, Heidelberg (2014)
36. Hiromasa, R., Abe, M., Okamoto, T.: Packing messages and optimizing bootstrapping in GSW-FHE. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 699–715. Springer, Heidelberg (2015)
37. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC Press, Boca Raton (2007)
38. Khedr, A., Gulak, P.G., Vaikuntanathan, V.: SHIELD: Scalable homomorphic implementation of encrypted data-classifiers. IACR Cryptology ePrint Archive 2014, 838 (2014)
39. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of NLFSR-based cryptosystems. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010)
40. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
41. Knuth, D.E.: The Art of Computer Programming. Seminumerical Algorithms. Addison-Wesley, Boston (1969)
42. Lepoint, T., Naehrig, M.: A comparison of the homomorphic encryption schemes FV and YASHE. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 2014. LNCS, vol. 8469, pp. 318–335. Springer, Heidelberg (2014)
43. Lindner, R., Peikert, C.: Better key sizes (and attacks) for lwe-based encryption. In: Kiayias, A. (ed.) Topics in Cryptology – CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
44. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. *J. Cryptology* **24**(3), 588–613 (2011)
45. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.* **17**(2), 373–386 (1988)
46. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
47. Meier, W.: Fast correlation attacks: Methods and countermeasures. In: Joux, A. (ed.) Fast Software Encryption. LNCS, vol. 6733, pp. 55–67. Springer, Heidelberg (2011)
48. Meier, W., Staffelbach, O.: Fast correlation attacks on stream ciphers. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 301–314. Springer, Heidelberg (1988)
49. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012)
50. Micciancio, D., Regev, O.: Lattice-based cryptography. Springer, Heidelberg (2009)
51. Naehrig, M., Lauter, K.E., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011, pp. 113–124 (2011)
52. Piret, G., Roche, T., Carlet, C.: PICARO - A block cipher allowing efficient higher-order side-channel resistance. In: Bao, F., Samarati, P., Zhou, J. (eds.) Applied Cryptography and Network Security. LNCS, vol. 7341, pp. 311–328. Springer, Heidelberg (2012)

53. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22–24, 2005, pp. 84–93 (2005)
54. Rückert, M., Schneider, M.: Estimating the security of lattice-based cryptosystems. IACR Cryptology ePrint Archive 2010, 137 (2010)
55. Schnorr, C., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.* **66**, 181–199 (1994)
56. Siegenthaler, T.: Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Comput.* **34**(1), 81–85 (1985)
57. Standaert, F.-X., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 335–352. Springer, Heidelberg (2013)
58. Vershynin, R.: Introduction to the non-asymptotic analysis of random matrices. CoRR [abs/1011.3027](https://arxiv.org/abs/1011.3027) (2010)
59. Wiedemann, D.H.: Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theor.* **32**(1), 54–62 (1986)