Transciphering, Using FiLIP and TFHE for an Efficient Delegation of Computation

Clément Hoffmann[1(✉)], Pierrick Méaux[1(✉)], and Thomas Ricosset[2(✉)]

[1] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Louvain la neuve, Belgium
{clement.hoffmann,Pierrick.meaux}@uclouvain.be
[2] Thales, Gennevilliers, France
thomas.ricosset@thalesgroup.com

Abstract. Improved filter permutators are designed to build stream ciphers that can be efficiently evaluated homomorphically. So far the transciphering with such ciphers has been implemented with homomorphic schemes from the second generation. In theory the third generation is more adapted for the particular design of these ciphers. In this article we study how suitable it is in practice. We implement the transciphering of different instances of the stream cipher family FiLIP with homomorphic encryption schemes of the third generation using the TFHE library.

We focus on two kinds of filter for FiLIP. First we consider the direct sum of monomials, already evaluated using HElib and we show the improvements on these results. Then we focus on the XOR-threshold filter, we develop strategies to efficiently evaluate any symmetric Boolean function in an homomorphic way, allowing us to give the first timings for such filters. We investigate different approaches for the homomorphic evaluation: using the leveled homomorphic scheme TGSW, an hybrid approach combining TGSW and TLWE schemes, and the gate boostrapping approach. We discuss the costs in time and memory and the impact on delegation of computation of these different approaches, and we perform a comparison with others transciphering schemes.

Keywords: Homomorphic encryption · TFHE · Improved filter permutator · Transciphering

1 Introduction

Fully homomorphic encryption (FHE) enables to perform computations over encrypted data without decryption nor learning information on the data. Since the first construction due to Gentry [21] in 2009, FHE is considered as the main solution to conceive a secure delegation of computation. The principle is the following: the delegating party, say Alice, encrypts her data using a FHE scheme and sends it to the computing party, say Bob. He evaluates the functions asked by Alice on her encrypted data, and sends back the encrypted results, without learning the values of the data sent by Alice. Two main efficiency problems arise with this framework: the FHE ciphers are costly to compute for Alice, and the expansion factor between the plaintext size and the ciphertext size is prohibitive. Instead, an efficient framework to delegate computations is obtained with

C. Hoffmann—This work has been done during an internship at Thales.

K. Bhargavan et al. (Eds.): INDOCRYPT 2020, LNCS 12578, pp. 39–61, 2020.
https://doi.org/10.1007/978-3-030-65277-7_3

an hybrid scheme, combining symmetric encryption (SE) and homomorphic encryption [31]. In this framework, Alice uses a classic symmetric scheme to encrypt her data before sending it to the server. The advantages of the SE schemes are the computation by devices with limited computational power or storage, and the optimal expansion factor: the ciphertext size is exactly the data size. Then, Bob transciphers these SE ciphertexts: he homomorphically evaluates the SE decryption to get homomorphic ciphertexts of Alice's data. Finally, Bob performs the computations required by Alice on the encrypted data and sends back the encrypted results, as in the initial framework. With such hybrid framework, an efficient transciphering gives an efficient delegation of computation.

Fully homomorphic encryption allows to evaluate any function, then the efficiency of this evaluation depends on how the function can be expressed in the native operations of the FHE scheme. Following Gentry's blueprint, FHE schemes are based on a somewhat encryption scheme and a bootstrapping. The somewhat (or leveled) scheme allows to perform a limited amount of two different operations, for example XOR and AND. In an homomorphic ciphertext the message is masked by a quantity of noise, and this noise is increasing during the operations, more or less importantly depending on the operation nature. The bootstrapping is the key technique that resets a ciphertext to be used again in the somewhat encryption scheme. The different costs in time (and storage) between the homomorphic operations and the bootstrapping depend on the FHE scheme, usually[1] there is the following hierarchy of cost: one operation is more efficient than the other, and the bootstrapping is way more costly than the operations. Therefore, an efficient transciphering is obtained by a SE scheme which decryption function is fitting with the FHE cost hierarchy.

State-of-the-Art. The so-called second generation (2G) of FHE which is represented by the schemes [5,6,20,26] has been widely used to implement transcipherings, using open-source libraries such as HElib [24] and SEAL [10]. In these schemes the multiplication increases the noise way more than the addition and the bootstrapping is very costly. Therefore, when evaluating a function as a circuit, the multiplicative depth dictates the efficiency. The SE schemes considered for transciphering have been standard schemes with low multiplicative depth such as: AES [15,22], Simon [25], Prince [17] and Trivium [8]. More recent SE schemes are designed for advanced primitive such as multi-party computation, zero-knowledge and FHE and share the property of having a low multiplicative depth, and they give the most efficient transciherings so far. It is the case of the block-cipher LowMC [3,4], and the stream-ciphers Kreyvium [8], FLIP [30], Rasta [16], and FiLIP [29].

The third generation (3G), beginning with GSW scheme [23] has a very different cost hierarchy. The multiplication is still more costly than the addition, but the error growth is asymmetric in the two operands, then long chains of multiplications can be evaluated without bootstrapping. The multiplicative depth does not dictate the efficiency in this case, and other SE schemes could give a better transciphering. This generation also allows the gate-bootstraping [11,18], to evaluate a Boolean gate and perform the bootstrapping at the same time, as quick as 13 ms [12] over the TFHE library [14]. The 3G is promising for transciphering but none has been realized so far. The main reasons were the crippling sizes of the original ciphertexts in comparison with the 2G, and the difficulty to adapt a SE scheme to this particular cost hierarchy.

[1] The situation is different for FHE schemes that apply a bootstraping at each gate.

In this work we realize a transciphering with a FHE scheme of third generation. The SE scheme we consider is FiLIP [29], a stream-cipher designed for homomorphic evaluation. The principle of this stream cipher is to restrict the homomorphic computation to the evaluation of its filter: only one Boolean function f. For the security point of view, the filter needs to fulfill various cryptographic criteria on Boolean functions to resist known attacks up to a fixed level of security. If this filter can be evaluated appropriately with the homomorphic cost hierarchy, then the whole transciphering is efficient. We implement the transciphering using the TFHE library [14], offering different HE schemes of the third generation. We focus primarily on a version of the TGSW scheme, a variant of GSW [23] over the torus.

Contributions. We analyse the homomorphic evaluation of FiLIP with the TGSW scheme, we implement the transcipherings with two families of filters, using three homomorphic schemes of the third generation.

We study the homomorphic error growth of FiLIP with TGSW for two kinds of filters: direct sum of monomials (DSM) [29] and XOR-threshold functions suggested in [28]. For the DSM filter the bound on the error generalizes the bound of [30] on FLIP functions with GSW. To analyze the error growth of the second filter we show how to efficiently evaluate any symmetric Boolean function in 3G, and more particularly threshold functions. Then we bound the ciphertext error for XOR-threshold filters, confirming that a function with high multiplicative depth can be efficiently evaluated.

We implement the two different kinds of filters for instances designed for 128 bit-security with TGSW. We analyse the noise in practice and the timings of this transciphering, which gives a latency of less than 2 s for the whole transciphering. We give a comparison with transcipherings from former works using the second homomorphic generation (on HElib for instance). For an equivalent resulting noise and security level, the latency of our transciphering is better than for the ones already existing.

Finally, we implement the same variants of FiLIP with an a hybrid TGSW/TLWE scheme and with the gate-bootstrapping FHE of [12], reaching a latency of 1.0s for an only-additive homomorphic scheme. We provide comparisons between the three evaluations of FiLIP we implemented and the evaluation over HElib in [29].

Roadmap. In Sect. 2, we remind definitions and properties from the TFHE scheme and FiLIP and describe the TGSW scheme we will use. In Sect. 3, we study the homomorphic evaluation of FiLIP filters and give a theoretical bound the noise after the transciphering. Finally, we present our practical results (resulting noises and timings) in Sect. 4 and compare our implementations to the ones already existing.

2 Preliminaries

We use the following notations:

- $\mathbb{B}$ denotes the set $\{0, 1\}$, and $[n]$ the set of integers from 1 to n.
- $x \xleftarrow{\$} S$ means that x is uniformly randomly chosen in a set S.
- $\mathbb{T}$ denotes the real torus $\mathbb{R}/\mathbb{Z}$, *i.e.* the real numbers modulo 1.
- $\mathbb{T}_N[X]$ denotes $\mathbb{R}[X]/(X^N + 1) \pmod 1$, $\mathfrak{R}$ denotes the ring of polynomials $\mathbb{Z}[X]/(X^N + 1)$ and $\mathbb{B}_N[X]$ denotes $\mathbb{B}[X]/(X^N + 1)$.

- Vectors and matrices are denoted in bold (*e.g.* $\mathbf{v}, \mathbf{A}$). $\mathcal{M}_{m,n}(S)$ refers to the space of $m \times n$ dimensional matrices with coefficients in S.
- $\mathsf{w_H}$ denotes the Hamming weight of a binary vector.
- MUX refers to the multiplexer: for binary variables $\mathsf{MUX}(x_1, x_2, x_3)$ gives x_2 if $x_1 = 0$ and x_3 if $x_1 = 1$. We call x_1 the control bit, x_2 the value at 0 and x_3 the value at 1.

2.1 Homomorphic Encryption and TFHE

In this section, we start by introducing definitions and properties from [11] on homomorphic encryption schemes and operations implemented in the TFHE library [14]. In a second time we describe the leveled homomorphic encryption scheme we will use for transciphering based on the TFHE definitions.

TFHE Toolbox. The TFHE library [14] implements a gate-by-gate bootstrapping based on [11–13]. Different homomorphic encryption schemes are combined for this bootstrapping: LWE, TLWE and TGSW. We present only definitions and properties needed for the evaluation of the TGSW leveled encryption scheme we will use in this work, and refer to [11] for more details.

Definition 1 (LWE samples). *Let $k \in \mathbb{N}$ a dimension parameter, $\alpha \in \mathbb{R}^+$ a noise parameter, $\mathbf{s}$ a uniformly distributed secret in some bounded set $S \subset \mathbb{Z}^n$. A LWE sample under the key $\mathbf{s}$ with noise parameter α is a couple $(\mathbf{a}, b) \in \mathbb{T}^n \times \mathbb{T}$ where $b - \langle \mathbf{s}, \mathbf{a} \rangle$ follows a Gaussian distribution of standard deviation α.*

Definition 2 (TLWE samples). *Let $k \geq 1$, N a power of 2, α a noise parameter, $\mathbf{s} \xleftarrow{\$} \mathbb{B}_N[X]^k$ a TLWE secret key. A fresh TLWE sample of a message $\mu \in \mathbb{T}_N[X]$ with noise parameter α under the key $\mathbf{s}$ is a couple $(\mathbf{a}, b) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$, where $\mathbf{a}$ is uniformly chosen in $\mathbb{T}_N[X]^k$ and $b - \langle \mathbf{s}, \mathbf{a} \rangle$ follows a Gaussian distribution of standard deviation α centered in μ.*

The scheme introduced in [11] gives a gate-bootstrapping of LWE ciphers. Instead, we focus on the homomorphic properties of TLWE: TLWE samples can be used to encrypt messages $\mu \in \mathcal{P} \subset \mathbb{T}_N[X]$ as $c = (\mathbf{a}, b) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$, where $b = \langle \mathbf{a}, \mathbf{s} \rangle + \mu + e \in \mathbb{T}_N[X]$. This variant of Regev's secret key encryption scheme is additively homomorphic as far as each coefficient of e is smaller than half the minimal distance between the coefficients of two possible messages. The introduction of TGSW ciphers with a decomposition of TLWE ciphers gives us a multiplicative homomorphic scheme.

Definition 3 (Gadget decomposition). *For $B_g \in \mathbb{N}$, let define the gadget matrix $\mathbf{H} \in \mathcal{M}_{(k+1)\cdot\ell, k+1}(\mathbb{T}_N[X])$ as:*

$$\mathbf{H} = \left(\begin{array}{c|c|c} 1/B_g & \dots & 0 \\ \vdots & \ddots & \vdots \\ 1/B_g^\ell & \dots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \dots & 1/B_g \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1/B_g^\ell \end{array}\right)$$

$Decomp_{h,\beta,\epsilon}(\mathbf{v})$ is a decomposition algorithm on the gadget $\mathbf{H}$ with quality β and precision ϵ if and only if, for any TLWE sample $\mathbf{v} \in \mathbb{T}_N[X]^{k+1}$, it efficiently and publicly outputs a small vector $\mathbf{u} \in \mathfrak{R}^{(k+1)\ell}$ such that $||\mathbf{u}||_\infty \leq \beta$ and $||\mathbf{u}\times\mathbf{H}-\mathbf{v}||_\infty \leq \epsilon$. Furthermore, the expectation of $\mathbf{u}\times\mathbf{H}-\mathbf{v}$ must be 0 when $\mathbf{v}$ is uniformly distributed in $\mathbb{T}_N[X]^{k+1}$.

Such a decomposition with $\beta = \frac{B_g}{2}$ and $\epsilon = \frac{1}{2B_g^\ell}$ exists and an example is described in [11]. It allows to define TGSW samples:

Definition 4 (TGSW samples). *Let ℓ and $k \geq 1$ two integers, $\alpha \geq 0$ a noise parameter and $\mathbf{H}$ the gadget matrix. Let $\mathbf{s} \in \mathbb{B}_N[X]^k$ a TLWE key, then $\boldsymbol{C} \in \mathcal{M}_{(k+1)\ell,k+1}(\mathbb{T}_N[X])$ is a fresh TGSW sample of $\mu \in \mathfrak{R}$ such that $\mu \cdot \mathbf{H} \neq 0$ with noise parameter α if and only if $\boldsymbol{C} = \mathbf{Z} + \mu \cdot \mathbf{H}$ where each row of $\mathbf{Z} \in \mathcal{M}_{(k+1)\ell,k+1}(\mathbb{T}_N[X])$ is a TLWE cipher of 0 with noise parameter α.*

Note that the product between μ and $\mathbf{H}$ is the $\mathfrak{R}$-module product, which means that each coefficient of $\mathbf{H}$ is multiplied by μ. TGSW ciphers remain homomorphically additive, and we can now introduce the homomorphic multiplications:

Definition 5 (External and internal products). *Let define the external product $\boxdot$ and internal product $\boxtimes$ as:*

$$\begin{aligned} \boxdot : &\mathit{TGSW} \times \mathit{TLWE} \to \mathit{TLWE} \\ &(\boldsymbol{A}, \mathbf{b}) \mapsto \boldsymbol{A} \boxdot \mathbf{b} = \mathtt{Decomp}_{\mathbf{H},\beta,\epsilon}(\mathbf{b}) \cdot \boldsymbol{A} \end{aligned}$$

$$\begin{aligned} \boxtimes : &\mathit{TGSW} \times \mathit{TGSW} \to \mathit{TGSW} \\ &(\boldsymbol{A}, \boldsymbol{B}) \mapsto \boldsymbol{A} \boxtimes \boldsymbol{B} = \begin{pmatrix} \boldsymbol{A} \boxdot \mathbf{b}_1 \\ \vdots \\ \boldsymbol{A} \boxdot \mathbf{b}_{(k+1)\ell} \end{pmatrix} \end{aligned}$$

where $\forall i \in \{1, \dots, (k+1)\ell\}, \mathbf{b}_i$ corresponds to the i-th line of $\boldsymbol{B}$.

A TGSW Somewhat Homomorphic Scheme. We describe a version of TGSW as a somewhat homomorphic encryption scheme, allowing to perform a bounded number of additions and multiplications. We consider a secret key scheme with plaintext space $\mathcal{P} = \{0, 1\}$.

Definition 6 (TGSW leveled homomorphic encryption scheme). *Let $k, N \in \mathbb{N}^*$, N a power of 2 the dimension parameters. Let $\ell, B_g \in \mathbb{N}$ the decomposition parameters. Let $\alpha \in \mathbb{R}^+$ the noise parameter.*

- $\mathsf{KeyGen}(k, N)$. *From dimension parameters k, N, output* $\mathsf{sk} \xleftarrow{\$} \mathbb{B}_N[X]^k$.
- $\mathsf{Enc}(\mathsf{sk}, \mu, \ell, B_g, \alpha)$. *Using as inputs* $\mathsf{sk} \in \mathbb{B}_N[X]^k, \mu \in \{0,1\}$, ℓ, B_g *decomposition parameters, and α the noise parameter:*
  - *Pick* $\mathbf{A} \xleftarrow{\$} \mathcal{M}_{(k+1)\ell,k}(\mathbb{T}_N[X])$.
  - *Compute* $\mathbf{e} = (e_i)_{i \in [(k+1)\ell]} \in \mathbb{T}_N[X]^{(k+1)\ell}$ *where each e_i follows a centered Gaussian distribution of standard deviation α.*
  - *Compute* $Z = \left(\begin{array}{c|c} & \langle \mathbf{a_1}, \mathsf{sk} \rangle + e_1 \\ \mathbf{A} & \vdots \\ & \langle \mathbf{a_{(k+1)\ell}}, \mathsf{sk} \rangle + e_{(k+1)\ell} \end{array} \right) \in \mathcal{M}_{(k+1)\ell,k+1}(\mathbb{T}_N[X])$.
  - *Output* $\mathbf{C} = \mathbf{Z} + \mu \cdot \mathbf{H}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathbf{C}, \ell, B_g)$. *Using as inputs the secret key* sk*, and a ciphertext* $\mathbf{C}$*,*
  - *Denote* $(\mathbf{a}, b) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$ *the* $(k\ell+1)^{th}$ *line of* $\mathbf{C}$*, compute* $\varphi = b - \langle \mathsf{sk}, \mathbf{a} \rangle \in \mathbb{T}_N[X]$.
  - *Round up the constant coefficient of φ to the closest* $\frac{i}{B_g} \in \mathbb{T}$ *where* $i \in \mathbb{N}$*, denoted* $\frac{i_m}{B_g}$.
  - *Output* $m \in \{0,1\}$ *the parity of* i_m.
- *The* Eval *algorithm consists in iterating the homomorphic operations* Add *and* Mul.
  - $\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2) : \mathbf{C}_+ = \mathbf{C}_1 + \mathbf{C}_2$.
  - $\mathsf{Mul}(\mathbf{C}_1, \mathbf{C}_2, B_g, \ell) : \mathbf{C}_\times = \mathbf{C}_1 \boxtimes \mathbf{C}_2$.

With this scheme, a TGSW ciphertext remains valid as far as the error terms are lower than $\frac{1}{2B_g}$. To follow the noise evolution with the homomorphic computations we use a worst case bound on the error coefficients (infinite norm), or an average bound on the variance of these coefficients, using the independence assumption formalized in [11]. To relate the error norm and the variance we use the Markov's inequality applied on subgaussians as in [11], it allows to estimate the maximal variance that can be tolerated for a fixed decryption failure.

Assumption 1 (Independence Heuristic ([11] Assumption 3.6, [13] Assumption 3.11)). *All the coefficients of the error of TLWE or TGSW samples that occur in all the operations we consider are independent and concentrated. More precisely, they are σ-subgaussian where σ is the square-root of their variance.*

Proposition 1 (Markov's inequality on subgaussians [11]). *Let X be a σ-subgaussian then* $\forall t > 0, P(|X| > t) \leq 2e^{-\frac{t^2}{2\sigma^2}}$.

We summarize the noise evolution during the homomorphic evaluation proven in [11]. The equations are simplified since $\mu \in \{0,1\}$: since the plaintexts are binary, they directly appear in the noise formula.

Proposition 2 (TGSW noise evolution, adapted form [11]). *Using the notations of Definition 6, for $i \in [3]$ let $\mathbf{C_i}$ be a TGSW cipher of μ_i with error noise variance V_i and infinite norm ε_i. We denote by V_+ and ε_+ the error variance and infinite norm of $\mathbf{C_1}+\mathbf{C_2}$, by $V_\times$ and $\varepsilon_\times$ the error variance and infinite norm of $\mathbf{C_1}\boxtimes\mathbf{C_2}$, by V_M and ε_M the error variance and infinite norm of $\mathsf{MUX}(\mathbf{C_1},\mathbf{C_2},\mathbf{C_3}) = \mathbf{C_1}\boxtimes(\mathbf{C_3}-\mathbf{C_2})+\mathbf{C_2}$. Then:*

$$\begin{cases} \varepsilon_+ \le \varepsilon_1+\varepsilon_2, & V_+ \le V_1+V_2, \\ \varepsilon_\times \le c_1 e_1 + \mu_1(c_2+e_2), & V_\times \le c_3 V_1 + \mu_1(c_4+V_2), \\ \varepsilon_M \le \max(\varepsilon_2,\varepsilon_3) + c_1 e_1 + c_2, & V_M \le \max(V_2,V_3) + c_3 V_1 + c_4, \end{cases}$$

where $c_1 = (k+1)\ell N(B_g/2)$, $c_2 = (1+kN)/(2B_g^\ell)$, $c_3 = (k+1)\ell N(B_g/2)^2$, $c_4 = (1+kN)/(2B_g^\ell)^2$. The variances bounds are obtained assuming Assumption 1.

2.2 Boolean Functions and FiLIP

We recall the definitions of a Boolean function, their common representation, and some families of functions.

Definition 7 (Boolean function). *A Boolean function f with n variables is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$.*

Definition 8 (Algebraic Normal Form (ANF)). *We call Algebraic Normal Form of a Boolean function f its n-variable polynomial representation over $\mathbb{F}_2$:*

$$f(x) = \sum_{I\subseteq[n]} a_I \left(\prod_{i\in I} x_i\right) = \sum_{I\subseteq[n]} a_I x^I,$$

where $a_I \in \mathbb{F}_2$.

- *The algebraic degree of f equals the global degree $\max_{\{I \mid a_I=1\}} |I|$ of its ANF.*
- *Any term $\prod_{i\in I} x_i$ in such an ANF is called a monomial and its degree equals $|I|$.*

Definition 9 (Direct sum of monomials & direct sum vector [30]). *Let f be a Boolean function of n variables, we call f a Direct Sum of Monomials (or DSM) if the following holds for its ANF: $\forall(I,J)$ such that $a_I = a_J = 1$, $I\cap J \in \{\emptyset, I\cup J\}$.*

Let f a DSM, we define its direct sum vector: $\mathbf{m}_f = [m_1, m_2, \ldots, m_k]$ of length $k = \mathsf{deg}(f)$, where m_i is the number of monomials of degree i of f: for $i>0$, $m_i = |\{a_I = 1, \text{ such that } |I| = i\}|$.

Note that DSM corresponds to functions where each variable appears at most once in the ANF.

Definition 10 (Threshold function). *Let $n \in \mathbb{N}^*$, for any positive integers $d \le n+1$ we define the Boolean function $\mathsf{T}_{d,n}$ as:*

$$\forall x = (x_1,\ldots,x_n) \in \mathbb{F}_2^n, \quad \mathsf{T}_{d,n}(x) = \begin{cases} 1 & \text{if } \mathsf{w_H}(x) \ge d, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 11 (XOR − THR function). *For any positive integers k, d and n such that $d \leq n+1$ we define* $\mathsf{XTHR}_{k,d,n}$ *for all* $z = (x_1, \ldots, x_k, y_1, \ldots, y_n) \in \mathbb{F}_2^{k+n}$ *as:*

$$\mathsf{XTHR}_{k,d,n}(z) = x_1 + \cdots + x_k + \mathsf{T}_{d,n}(y_1, \ldots, y_n) = \mathsf{XOR}_k(x) + \mathsf{T}_{d,n}(y).$$

The symmetric encryption schemes we will evaluate are binary streamciphers following the improved filter permutator construction [29], illustrated in Fig. 1. The encryption process is the following: denoting by N the size of the key, n the size of the input of the filtering function ($n \leq N$) and f the n-variables filtering function:

- The forward secure PRNG outputs the subset, the permutation and the whitening at each clock cycle,
- the subset S_i is chosen as a subset of n elements over N,
- the permutation P_i from n to n elements is chosen,
- the whitening $w_i \in \mathbb{F}_2^n$ is chosen,
- the key-stream bit is computed as $s_i = f(P_i(S_i(K)) + w_i)$.

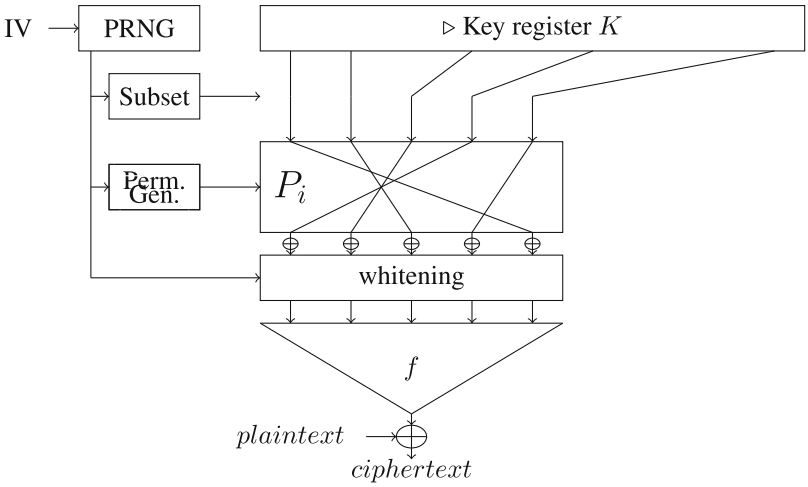

Fig. 1. Improved filter permutator constructions.

The streamcipher family FiLIP is an instantiation of the improved filter permutator paradigm where the PRNG is a variant of AES in counter mode, and the wire-cross permutations are generated by the Knuth shuffle. We will focus on 3 candidates proposed for 128-bit security:

- FiLIP-1216 [29], DSM filter, $\mathbf{m}_f = [128, 64, 0, 80, 0, 0, 0, 80]$, $n = 1216$, $N = 2^{14}$,
- FiLIP-1280 [29], DSM filter, $\mathbf{m}_f = [128, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64]$, $n = 1280$, $N = 2^{12}$,
- FiLIP-144 [28], XOR-THR filter, $f = \mathsf{XTHR}_{81,32,63}$, $n = 144$, $N = 2^{14}$.

3 Homomorphic Evaluation of FiLIP

During the transciphering with FiLIP only the filter evaluation has an impact on the ciphertexts' noise. Indeed, as explained in [29] the subset selection and wire-cross permutation are performed in clear. The whitening gives indexes where a homomorphic NOT needs to be applied, which can be realized without increasing the noise.

In this section we study how to evaluate FiLIP filters efficiently with TGSW. First we give a Boolean circuit to compute any symmetric Boolean functions with few gates. This circuit improves the homomorphic evaluation for several FHE schemes, but its design is primarily based on optimizing GSW-like FHE evaluation. It will be the core of the evaluation of XOR-THR filters. Then we specify which representation and therefore which circuit we use for the two kinds of filters, giving a lower bound on the Boolean gates to homomorphically evaluate in each case. Finally we give an upper bound on the noise in the TGSW ciphertext obtained after evaluating the different filters.

3.1 Evaluation of Symmetric Boolean Functions

In this part we focus on efficiently evaluating threshold functions. Since they are symmetric Boolean functions (the result does not depend on the order of the n inputs), they can be evaluated in different ways. The ANF of such functions is characterized in [27]; when $n \geq 2^{\lceil \log d \rceil}$ all the monomials of this degree appear in the ANF, which seems prohibitive for any evaluation. Nevertheless, evaluating branching programs [7] or finite automata [11] has been shown to be very promising with the third generation. Hence, we consider the evaluation of thresholds function, or other symmetric Boolean function, with MUX gates (multiplexers) rather than based on the ANF representation.

Definition 12 (Circuit for symmetric functions). *Let $n \in \mathbb{N}$, $n \geq 2$, we define the Boolean circuit $\mathcal{C}_n$ with n inputs x_1 to x_n and $n+1$ outputs y_0 to y_n with the following gates:*

- *n NOT gates N_1 to N_n, where for $i \in [n]$ each N_i has input x_i.*
- *$2(n-1)$ AND gates $A_{i,0}$ and $A_{i,i}$ for $i \in [2,n]$, where:*
  - *the inputs of $A_{2,0}$ are N_1 and N_2, and for $i \in [3,n]$ the inputs of $A_{i,0}$ are N_i and $A_{i-1,0}$,*
  - *the inputs of $A_{2,2}$ are x_1 and x_2, and for $i \in [3,n]$ the inputs of $A_{i,i}$ are x_i and $A_{i-1,i-1}$.*
- *$\binom{n}{2}$ MUX gates $M_{i,j}$ where $i \in [2,n]$ and $j \in [i-1]$, where:*
  - *the inputs of $M_{2,1}$ are x_2 for the control bit, x_1 for the value at 0, and N_1 for the value at 1,*
  - *the inputs of $M_{i,1}$ for $i \in [3,n]$ are x_i for the control bit, $M_{i-1,1}$ for the value at 0, and $A_{i-1,0}$ for the value at 1,*
  - *the inputs of $M_{i,i-1}$ for $i \in [3,n]$ are x_i for the control bit, $A_{i-1,i-1}$ for the value at 0, and $M_{i-1,i-2}$ for the value at 1,*
  - *the inputs of $M_{i,j}$ for $i \in [4,n]$ and $j \in [2,i-2]$ are x_i for the control bit, $M_{i-1,j}$ for the value at 0, and $M_{i-1,j-1}$ for the value at 1.*

The outputs are given by $A_{n,0}, M_{n,1}, \ldots, M_{n,n-1}, A_{n,n}$.

To illustrate we give the example of $\mathcal{C}_7$ in Fig. 2. The principle of this multi-output circuit is to compute the Hamming weight of the vector of inputs, as formalized in the following proposition:

Proposition 3. *Let $n \in \mathbb{N}$, $n \geq 2$, $\forall x \in \mathbb{F}_2^n$ the vector of outputs of $\mathcal{C}_n$ gives 1 at index $\mathsf{w_H}(x)$ and 0 elsewhere.*

Proof. We do the proof by induction. We show that $\forall i \in [2, n]$ only one of the gates $A_{i,j}$ or $M_{i,j}$ (where $j \in [0, i]$) gives 1. The index j of this gate corresponds the Hamming weight of the i-th-length prefix of x: $(x_1, \ldots, x_i)$. Note that the statements always neglect the NOT gates.

For the initialization step we exhaust the four cases, the values for $(x_1, x_2) \in \mathbb{F}_2^2$ of $(A_{2,0}, M_{2,1}, A_{2,2})$ are: $(0, 0; 1, 0, 0)$, $(1, 0; 0, 1, 0)$, $(0, 1; 0, 1, 0)$, and $(1, 1; 0, 0, 1)$. For the four cases only one of the three gates of level 2 gives 1, the one with index $j = \mathsf{w_H}(x_1, x_2)$, which validates the initialization.

Let assume for one $i \in [2, n-1]$ that only the gate of index $j = \mathsf{w_H}(x_1, \ldots, x_i)$ outputs 1 and all the others at level i give 0. Note that at level $i+1$ the AND gate $A_{i+1,0}$ could by written as a MUX gate with control bit x_{i+1}, the output of $A_{i,0}$ for value at 0 and 0 for value at 1. Similarly, the AND gate $A_{i+1,i+1}$ could by written as a MUX gate with control bit x_{i+1}, the output of $A_{i,i}$ for value at 1 and 0 for value at 0. Consequently, the $i + 2$ gates at level $i + 1$ correspond to MUX gates: they all output their value at 0 (*i.e.* value of their right parent) if $x_{i+1} = 0$, or value at 0 (*i.e.* left parent) if $x_{i+1} = 1$. Therefore, the vector of outputs at level $i + 1$ is the one of level i with a 0 appended on the right if $x_{i+1} = 0$ or on the left if $x_{i+1} = 1$. It guarantees that there is only one 1 at level $i + 1$, the index being $\mathsf{w_H}(x_1, \ldots, x_i)$ if $x_{i+1} = 0$ and $\mathsf{w_H}(x_1, \ldots, x_i) + 1$ if $x_{i+1} = 1$, in both cases it corresponds to $\mathsf{w_H}(x_1, \cdots, x_{i+1})$. This proves the induction step, and allows to conclude the induction. Since the property applies for $i = n$ the outputs $y_0, \cdots, y_n$ of $\mathcal{C}_n$ for $x \in \mathbb{F}_2^n$ are such that $y_{\mathsf{w_H}(x)} = 1$ and the others are equal to 0. □

The interest of the circuit $\mathcal{C}_n$ is to compute symmetric Boolean functions with a low number of gates. Note that for $i \in [0, n]$ the output y_i gives the result of the indicator function of the set $\{x \in \mathbb{F}_2^n \mid \mathsf{w_H}(x) = i\}$. Since these functions form a basis of the symmetric n-variable Boolean functions, any symmetric n-variable Boolean function can be computed by xoring outputs of $\mathcal{C}_n$. For the threshold function $\mathsf{T}_{d,n}$ it would consist in xoring the outputs from y_d to y_n. Nevertheless, computing $\mathcal{C}_n$ and xoring some of the outputs involves unnecessary computations, therefore we will prefer a simplified circuit for $\mathsf{T}_{d,n}$ which has around twice less gates.

To simplify the description we introduce a new Boolean gate, used instead of some MUX gates in $\mathcal{C}_n$.

Definition 13 (Modified MUX gate). *Let $x_1, x_2, x_3 \in \mathbb{F}_2^n$, the modified* MUX *gate* WUX *is defined as* $\mathsf{WUX}(x_1, x_2, x_3) = x_1 x_3 + x_2$*, where by analogy with the* MUX *x_1 is called the control bit, x_2 the value at 0, and x_3 the value at 1.*

Note that the only difference with the classic gate is that x_3 is $x_3 + x_2$ in the MUX gate, and the WUX gate is simply computed with one AND and one XOR. The simpler circuit to compute $\mathsf{T}_{d,n}$ is obtained by two modifications. First, we delete the gates only

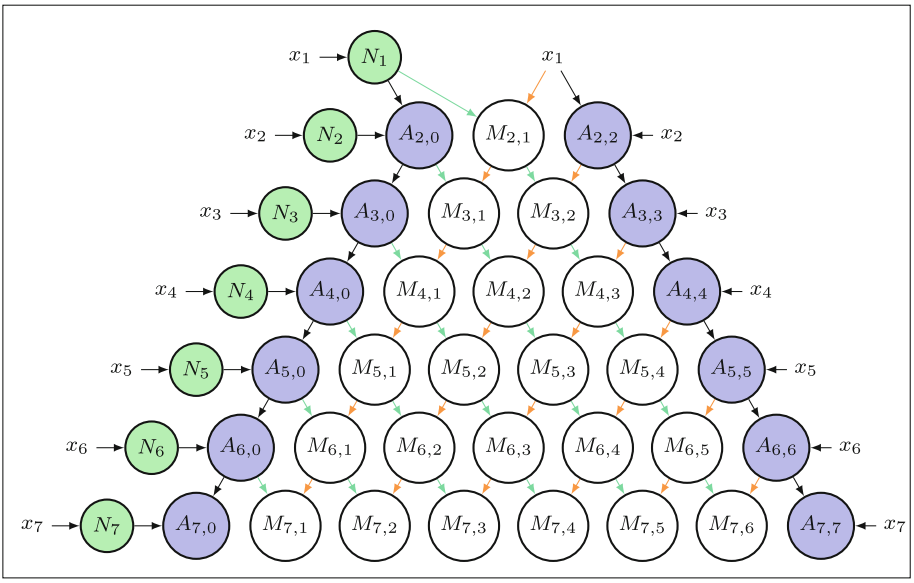

Fig. 2. Boolean circuit $\mathcal{C}_7$. The control bit of each level of MUX is the input x_i on the left side, the left arrow (⟶) in input of a MUX corresponds to the input for a control bit equal to 0 and the right arrow (⟶) for the value at 1.

used to compute the y_i such that $i < d$ (it corresponds to delete $A_{n-d+1,0}$ and all the gates depending on its value). Then, we subside the parts leading to y_i such that $i > d$ by sums as soon as possible in the computation using WUX gates (it corresponds to delete $A_{d+1,d+1}$ and all the gates depending on its value, and converting the MUX gates depending on the value of $A_{d,d}$ into WUX gates). We formalize the obtained circuit in Definition 14, illustrate it with the example of $\mathsf{T}_{4,7}$ in Fig. 3, and assess its property in the following proposition.

Definition 14 (Threshold circuit). *Let $d, n \in \mathbb{N}$, $n \geq 4$, $2 \leq d \leq n-2$, we define the Boolean circuit $\mathcal{T}_{d,n}$ with n inputs x_1 to x_n and one output with the following gates:*

- $n-d$ NOT *gates N_1 to N_{n-d}, where for $i \in [n-d]$ each N_i has input x_i.*
- $n-2$ AND *gates $A_{i,0}$ for $i \in [2, n-d]$ and $A_{i,i}$ for $i \in [2, d]$, where:*
  - *the inputs of $A_{2,0}$ are N_1 and N_2, and for $i \in [3, n-d]$ the inputs of $A_{i,0}$ are N_i and $A_{i-1,0}$,*
  - *the inputs of $A_{2,2}$ are x_1 and x_2, and for $i \in [3, d]$ the inputs of $A_{i,i}$ are x_i and $A_{i-1,i-1}$.*
- $(n-d)(d-1)$ MUX *gates $M_{i,j}$ for $i \in [2, n-1]$, $j \in [\max(1, i-(n-d)), \min(i-1, d-1)]$, where:*
  - *the inputs of $M_{2,1}$ are x_2 for the control bit, x_1 for the value at 0, and N_1 for the value at 1,*
  - *the inputs of $M_{i,1}$ for $i \in [3, n-d+1]$ are x_i for the control bit, $M_{i-1,1}$ for the value at 0, and $A_{i-1,0}$ for the value at 1,*

- *the inputs of* $M_{i,i-1}$ *for* $i \in [3,d]$ *are* x_i *for the control bit,* $A_{i-1,i-1}$ *for the value at* 0*, and* $M_{i-1,i-2}$ *for the value at* 1*,*
- *the inputs of* $M_{i,j}$ *for* $i \in [4,n-1]$ *and* $j \in [\max(1,i-(n-d)),\min(i-1,d-1)]$ *are* x_i *for the control bit,* $M_{i-1,j}$ *for the value at* 0*, and* $M_{i-1,j-1}$ *for the value at* 1*.*

– $(n-d)$ WUX *gates* $W_{i,d}$ *for* $i \in [d+1,n]$ *where:*

- *the inputs of* $W_{d+1,d}$ *are* x_{d+1} *for the control bit,* $A_{d,d}$ *for the value at* 0*, and* $M_{d,d-1}$ *for the value at* 1*,*
- *the inputs of* $W_{i,d}$ *for* $i \in [d+2,n]$ *are* x_i *for the control bit,* $W_{i-1,d}$ *for the value at* 0*, and* $M_{i-1,d-1}$ *for the value at* 1*.*

The output is given by $W_{n,d}$*.*

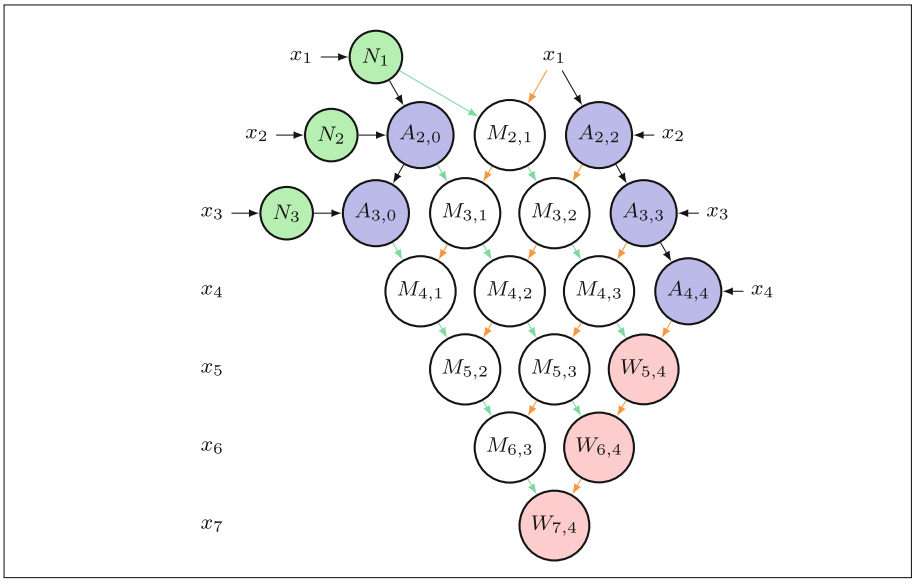

Fig. 3. Boolean circuit $\mathcal{T}_{4,7}$. The control bit of each level of MUX is the input x_i on the left side, the left arrow (→) in input of a MUX (or WUX) corresponds to the input for a control bit equal to 0 and the right arrow (→) for the value at 1.

Proposition 4. *Let* $d,n \in \mathbb{N}$*,* $n \geq 4$*,* $2 \leq d \leq n-2$*,* $\forall x \in \mathbb{F}_2^n$ *the Boolean circuit* $\mathcal{T}_{d,n}$ *computes* $\mathsf{T}_{d,n}(x)$*.*

Proof. By construction $\mathcal{T}_{d,n}$ is obtained from $\mathcal{C}_n$ by two main transformations: deleting $A_{n-d+1,0}$ and the gates depending on its output (plus the d NOT gates such that $i > n-d$), and merging the gates depending on $A_{d,d}$ into one WUX gate at each level i for $i \in [d+1,n]$. We will first prove by induction that in the circuit obtained from $\mathcal{C}_n$ by

merging the gates depending on $A_{d,d}$ in $n-d$ WUX gates $W_{i,d}$, the following property holds: $W_{i,d}$ gives 1 if and only if $\mathsf{w_H}(x_1,\ldots,x_i) \geq d$. Then we will show that deleting all the part depending on $A_{n-d+1,0}$ does not impact the computation of $W_{n,d}$ which gives the output.

We define the Boolean circuit $\mathcal{C}_{d,n}$ as $\mathcal{C}_n$ where for $i \in [d+1,n]$ the AND gate $A_{i,i}$ and the MUX gates $M_{i,j}$ for $j \in [d,i-1]$ are merged into the WUX gate $W_{i,d}$. $W_{d+1,d}$ has control bit x_{d+1}, $A_{d,d}$ as value at 0 and $M_{d,d-1}$ as value at 1. For $i \in [d+2,n]$, $W_{i,d}$ has control bit x_i, $W_{i,d}$ as value at 0 and $M_{i,d-1}$ as value at 1. We show by induction that $\forall i \in [d+1,n]$ $W_{i,d}$ gives 1 if and only if $\mathsf{w_H}(x_1,\ldots,x_i) \geq d$, and only one of the gates $A_{i,0}$, $W_{i,d}$, $M_{i,j}$ for $j \in [d-1]$ gives 1.

For the initialization step, $i = d+1$, $\mathcal{C}_{d,n}$ up to the level d is exactly $\mathcal{C}_d$, therefore using Proposition 3 $M_{d,d-1}$ gives 1 if and only if $\mathsf{w_H}(x_1,\ldots,x_d) = d-1$ and $A_{d,d}$ gives 1 if and only if $\mathsf{w_H}(x_1,\ldots,x_d) = d$. We consider the three possible cases: both gates give 0, $M_{d,d-1}$ gives 1, and $A_{d,d}$ gives 1. In the first case the value of $W_{i,d}$ is $x_{d+1}\cdot 0+0=0$ and since $\mathsf{w_H}(x_1,\ldots,x_d) < d-1$ we get $\mathsf{w_H}(x_1,\ldots,x_{d+1}) < d$. In the second case the value of $W_{i,d}$ is $x_{d+1}\cdot 1+0 = x_{d+1}$ and since $\mathsf{w_H}(x_1,\ldots,x_d) = d-1$ we get $\mathsf{w_H}(x_1,\ldots,x_{d+1}) = d$ if $x_{d+1}=1$ and $\mathsf{w_H}(x_1,\ldots,x_{d+1}) = d-1$ if $x_{d+1}=0$. In the third case the value of $W_{i,d}$ is $x_{d+1}\cdot 0+1=1$ and since $\mathsf{w_H}(x_1,\ldots,x_d) = d$ we get $\mathsf{w_H}(x_1,\ldots,x_{d+1}) \geq d$. Summarizing the three cases $W_{d+1,d}$ gives 1 if and only if $\mathsf{w_H}(x_1,\ldots,x_{d+1}) \geq d$. Note that the gates $A_{d+1,0}$ and $M_{d+1,j}$ for $j \in [d-1]$ are computed exactly as in $\mathcal{C}_{d+1}$. Hence, from Proposition 3 only one of them gives 1 if $\mathsf{w_H}(x_1,\ldots,x_{d+1}) < d$ and none if $\mathsf{w_H}(x_1,\ldots,x_{d+1}) \geq d$. Therefore only one of the gates $A_{d+1,0}$, $W_{d+1,d}$, $M_{d+1,j}$ for $j \in [d-1]$ gives 1, which validates the initialization.

Let assume that for one $i \in [d+2,n]$ $W_{i,d}$ gives 1 if and only if $\mathsf{w_H}(x_1,\ldots,x_i) \geq d$, and only one of the gates $A_{i,0}$, $W_{i,d}$, $M_{i,j}$ for $j \in [d-1]$ outputs 1. Note that the gates $A_{i+1,0}$, $M_{i+1,j}$ for $j \in [d-1]$ are computed exactly as in $\mathcal{C}_{i+1}$, therefore Proposition 3 guarantees that at most one outputs 1: if its index $j = \mathsf{w_H}(x_1,\ldots,x_{i+1})$. We consider the three possible cases: $W_{i,d}$ and $M_{i,d-1}$ both give 0, $M_{i,d-1}$ gives 1, and $W_{i,d}$ gives 1. In the first case the value computed by $W_{i+1,d}$ is 0, since $\mathsf{w_H}(x_1,\ldots,x_i) < d-1$ ($W_{i,d}$ and $M_{i,d-1}$ both null) then $\mathsf{w_H}(x_1,\ldots,x_{i+1}) < d$, and only one of the gates $A_{i+1,0}$, $M_{i+1,j}$ for $j \in [d-1]$ gives 1. In the second case the value computed by $W_{i+1,d}$ is x_{i+1}, therefore if $x_{i+1}=0$ then $\mathsf{w_H}(x_1,\ldots,x_{i+1}) = \mathsf{w_H}(x_1,\ldots,x_i) = d-1$ and $M_{i+1,d-1}$ is the only gate at level $i+1$ giving 1. If $x_{i+1}=1$ then $\mathsf{w_H}(x_1,\ldots,x_{i+1}) = \mathsf{w_H}(x_1,\ldots,x_i)+1 = d$, none of the gates $A_{i+1,0}$, $M_{i+1,j}$ for $j \in [d-1]$ gives 1, hence $W_{i+1,d}$ is the only one giving 1. In the last case the value computed by $W_{i+1,d}$ is 1, $\mathsf{w_H}(x_1,\ldots,x_i) \geq d$ so $\mathsf{w_H}(x_1,\ldots,x_{i+1}) \geq d$ and only the gate $W_{i+1,d}$ gives 1 at level $i+1$. Summarizing the different cases $W_{i+1,d}$ gives 1 if and only if $\mathsf{w_H}(x_1,\ldots,x_{i+1}) \geq d$, and only one of the gates $A_{i+1,0}$, $W_{i+1,d}$, $M_{i+1,j}$ for $j \in [d-1]$ outputs 1, which allows to conclude the induction.

To finalize the proof, note that the output of $\mathcal{C}_{d,n}$ given by $W_{n,d}$, that we call z_d, gives the value of $\mathsf{T}_{d,n}(x)$. None of the gates depending on $A_{n-d+1,0}$ nor N_i for $i \in [n-d+1,n]$ are evaluated in the path leading to z_d. Hence z_d can be computed by the circuit modified from $\mathcal{C}_{d,n}$ by removing all these gates. It corresponds to the circuit $\mathcal{T}_{d,n}$, concluding the proof. □

Remark 1. The restrictions $d \geq 2$ and $n - d \geq 2$ come from the circuit description: always having AND gates on the left and right side. Valid circuits for the remaining values can be obtained by removing some of these gates (changing the general description): the $A_{i,0}$ gates are unnecessary for $d \geq n - 2$ and the $A_{i,i}$ gates are unnecessary for $d \leq 2$.

3.2 Evaluating FiLIP Filters

For the homomorphic evaluation the filter needs to be computed without knowing the value in the input ciphertexts, hence using the same Boolean circuit for the 2^n possible inputs. A circuit to evaluate a Boolean function f can be derived from its ANF: each monomial can be computed as the AND of the variables of this monomial, and the sum over all the monomials in the ANF can be performed with XOR gates. It is the strategy we use to evaluate DSM since they have a very sparse ANF (at most n monomials over 2^n). The situation is different for XOR-THR functions, the threshold part has a dense ANF. The threshold function we consider for FiLIP-144: $\mathsf{T}_{32,63}$ belongs to the subfamily $\mathsf{T}_{2^t,2^{t+1}-1}$ and therefore its ANF consists in the $\binom{2^{t+1}-1}{2^t}$ monomials of degree 2^t (see *e.g.* [27] Theorem 1). In this case the circuit based on the ANF would lead to XOR around $9 \cdot 10^{17}$ AND of 32 terms. Instead we will use the circuit $\mathcal{T}_{d,n}$ of Sect. 3.1.

In the following proposition we give the number (and nature) of Boolean gates required to homomorphically compute the different filters.

Proposition 5 (FiLIP's filter evaluation). *Let f be the filter function of FiLIP, f can be computed with at most the following number of Boolean gates:*

- *f is the DSM with direct sum vector $\mathbf{m}_f = [m_1, \ldots, m_k]$:* 0 NOT, $m - 1$ XOR, *and* $n - m$ AND *where* $m = \left(\sum_{i=1}^{k} m_i\right)$.
- *f is the XOR-threshold function* $\mathsf{XTHR}_{k,d,n}$*:* $n - d$ NOT, $(n-d)(2d-1)+k$ XOR, *and* $(n-d)d + n - 2$ AND.

Proof. For a DSM, since each variable appears only once in the ANF, the result can be computed in $n - 1$ XOR and AND gates. The number of monomials being m, $m - 1$ XOR are needed, which also gives the number of AND.

For a XOR-THR function, using Proposition 4 the Boolean circuit $\mathcal{T}_{d,n}$ computes $\mathsf{T}_{d,n}$, which gives the number of NOT, AND, MUX, and WUX gates to evaluate it. Counting that a MUX can be computed with 2XOR and 1AND, and 1XOR and 1AND for a WUX, and that k XOR are necessary to sum the XOR_k part leads to the final number of gates. □

3.3 Noise Evolution with TGSW

In this part we bound the error infinite norm and variance of the ciphers after transciphering in term of error parameters of the initial ciphers. Each XOR is evaluated by Add, each AND by Mul and NOT gate by subtracting the ciphertext to $\mathbf{H}$.

Proposition 6 (FiLIP error-growth). *Let f be the filter function of FiLIP, let $\mathbf{C}_i$, $i \in [n]$, fresh TGSW ciphertexts with error variance and infinite norm V and ε, and $\mathbf{C}_f$ the ciphertext obtained by homomorphically evaluating f with error variance and infinite norm V_f and ε_f. The following bounds apply:*

- *if f is the DSM with direct sum vector $\mathbf{m}_f = [m_1, \ldots, m_k]$ then:*

$$\varepsilon_f \leq (n-m)(c_1\varepsilon + c_2) + m\varepsilon, \text{ and } V_f \leq (n-m)(c_3 V + c_4) + mV,$$

- *if f is the XOR-threshold function $\mathsf{XTHR}_{k,d,n}$ then:*

$$\varepsilon_f \leq \frac{(n+d-2)(n-d+1)}{2}(c_1\varepsilon + c_2) + (n-d+k+1)\varepsilon, \text{ and}$$

$$V_f \leq \frac{(n+d-2)(n-d+1)}{2}(c_3 V + c_4) + (n-d+k+1)V,$$

where the variance bounds assume the independence heuristic Assumption 1.

Proof. We refer to noise parameters for the two quantities error infinite norm and variance. The results on the error variance assume Assumption 1, not the ones on the error infinite norm.

We begin the proof by considering the DSM filter. Using Proposition 2, the noise parameters associated to a product of $i \in \mathbb{N}^*$ fresh ciphertexts (the noisiest ciphertext always taken as the second operand) are $\varepsilon_{\Pi_i} \leq (i-1)(c_1\varepsilon + c_2) + \varepsilon$ and $V_{\Pi_i} \leq (i-1)(c_3\varepsilon + c_4) + V$. It gives the noise parameters of the ciphertexts corresponding to the monomial of degree i. We use the noise formulas for the addition on the m_i products of i ciphertexts (for $i \in [k]$) and then for the sum of these k ciphertexts. It finally gives:

$$\begin{aligned}\varepsilon_f &\leq \sum_{i=1}^{k} m_i \varepsilon_{\Pi_i} \leq \sum_{i=1}^{k} m_i \left((i-1)(c_1\varepsilon + c_2) + \varepsilon\right) \\ &\leq \sum_{i=1}^{k} i m_i (c_1\varepsilon + c_2) + \sum_{i=1}^{k} m_i \left(\varepsilon - (c_1\varepsilon + c_2)\right) = (n-m)(c_1\varepsilon + c_2) + m\varepsilon,\end{aligned}$$

where m is the number of monomials of f and n its number of variables. Similarly, $V_f \leq (n-m)(c_3 V + c_4) + mV$.

For the XOR-threshold function we start with the noise parameters for the evaluation of $\mathsf{T}_{d,n}$ using the circuit $\mathcal{T}_{d,n}$. We bound the noise parameters of each ciphertext of the binary value obtained at each gate of the circuit since the ciphertext of $\mathsf{T}_{d,n}$ is the output of the circuit. for readability we write *noise parameters of the gate* for the noise parameters of the ciphertext obtained after the evaluation of the circuit up to this gate. We proceed by studying separately the NOT, AND, MUX, and WUX gates, and we refer to their level in the circuit (the index i in Definition 14). First, the noise parameters of NOT gates is the same as its input. Then, the AND gates are obtained by products of fresh ciphertexts then the noise parameters of $A_{i,j}$ are ε_{Π_i} and V_{Π_i}. For the MUX gates we show by induction on the level $i \in [2, n-1]$ that all MUX of level i have noise

parameters $\varepsilon_{M_i} \leq (i-1)(c_1\varepsilon + c_2) + \varepsilon$ and $V_{M_i} \leq (i-1)(c_3 V + c_4) + V$ (the same bounds as for ε_{Π_i} and V_{Π_i}). For the initialization step, $i = 2$, the only MUX gate is computed as $\mathsf{MUX}(\mathbf{C}_2, \mathbf{C}_1, \mathbf{H} - \mathbf{C}_i)$. Thereafter, the noise formula of the MUX in Proposition 2 gives $\varepsilon_{M_2} \leq \varepsilon + c_1\varepsilon + c_2$, validating the initialization. For the induction step, we assume that at level i we have the bound $\varepsilon_{M_i} \leq \varepsilon + (i-1)(c_1\varepsilon + c_2)$. By definition of $\mathcal{T}_{d,n}$ each MUX gate at level $i+1$ has input control bit x_{i+1} and the two other inputs are AND or MUX gates of the level i. From the induction hypothesis the error infinite norm relative to both these inputs is lower than or equal to $\varepsilon + (i-1)c_1\varepsilon + c_2$. Hence, the noise formula for the MUX gives $\varepsilon_{M_{i+1}} \leq \varepsilon + (i-1)(c_1\varepsilon + c_2) + c_1\varepsilon + c_2$, validating the induction step. The proof for V_{M_i} follows similar arguments, which allows to conclude the induction proof for the MUX gates. Finally, for the WUX gates we show by induction on $k \in [n-d]$ that the WUX gate $W_{d+k,d}$ has noise parameters $\varepsilon_{W_{d+k}} \leq (2d+k-2)(k+1)(c_1\varepsilon + c_2)/2 + (k+1)\varepsilon$ and similarly $V_{W_{d+k}} \leq (2d + k - 2)(k+1)(c_3 V + c_3)/2 + (k+1)V$ assuming the independence heuristic. Note that $(2d+k-2)(k+1)(c_1\varepsilon + c_2)/2 + (k+1)\varepsilon = \sum_{i=0}^{k}((d-1+i)(c_1\varepsilon + c_2) + \varepsilon)$ which is the formula we will use for the induction. For the initialization step, $k = 1$, the WUX gate $W_{d+1,d}$ is computed as $\mathsf{XOR}(\mathsf{AND}(x_{d+1}, M_{d,d-1}), A_{d,d})$. Thereafter, the noise formulas of Mul and Add of Proposition 2 give $\varepsilon_{W_{d+1}} \leq d(c_1\varepsilon + c_2) + \varepsilon + (d-1)(c_1\varepsilon + c_2) + \varepsilon$, which is equivalent to $\sum_{i=0}^{1}((d-1+i)(c_1\varepsilon + c_2) + \varepsilon)$, validating the initialization. For the induction step, we assume that for one value of k in $[n-d-1]$ we have $\varepsilon_{W_{d+k}} \leq \sum_{i=0}^{k}((d-1+i)(c_1\varepsilon + c_2) + \varepsilon)$. By definition of $\mathcal{T}_{d,n}$ the WUX gate at level $d+k+1$ has control bit x_{d+k+1} and the two other inputs are a MUX and the WUX of level $k+d$. Then, we know the noise parameters of the MUX from the precedent part and the error infinite norm of the WUX gate in input from the induction hypothesis. Hence, the product and addition noise formulas allow to conclude $\varepsilon_{W_{d+k+1}} \leq \sum_{i=0}^{k+1}((d-1+i)(c_1\varepsilon + c_2) + \varepsilon)$, validating the induction step. The proof for $V_{W_{d+k}}$ follows similar arguments, which allows to conclude the induction proof for the MUX gates. The last WUX gate ($W_{n,d}$) corresponds to the ciphertext of $\mathsf{T}_{d,n}$, therefore its noise parameters are $\varepsilon_{\mathsf{T}_{d,n}} \leq (n+d-2)(n-d+1)(c_1\varepsilon + c_2)/2 + (n-d+1)\varepsilon$ and $V_{\mathsf{T}_{d,n}} \leq (n+d-2)(n-d+1)(c_3 V + c_4)/2 + (n-d+1)V$.

Finally, the $\mathsf{XTHR}_{k,d,n}$ filter is evaluated by performing k Add of fresh ciphertexts to the ciphertext corresponding to $\mathsf{T}_{d,n}$. Using the precedent part of the proof and the noise formulas of Proposition 2 for the addition we obtain $\varepsilon_f \leq (n+d-2)(n-d+1)(c_1\varepsilon + c_2)/2 + (n-d+1+k)\varepsilon$ and $V_f \leq (n+d-2)(n-d+1)(c_3\varepsilon + c_4)/2 + (n-d+1+k)V$. □

Remark 2. The homomorphic evaluation of $\mathsf{T}_{d,n}$ can also be based on the circuit $\mathcal{C}_n$, adding the last $n-d$ output ciphertexts. It would lead to an infinite error norm $\varepsilon_{\mathsf{T}_{d,n}}$ such that $\varepsilon_{\mathsf{T}_{d,n}} \leq (n-d+1)((n-1)(c_1\varepsilon + c_2) + \varepsilon)$. With the same strategy, any n-variable symmetric Boolean function f can be evaluated with $\varepsilon_f \leq (n+1)((n-1)(c_1\varepsilon + c_2) + \varepsilon)$.

Also, multiplying the outputs y_i by x_i for $i \in [n]$ of $\mathcal{C}_n$ and summing these n products allows to compute the Hidden Weighted Bit Function (HWBF), which is known to have good cryptographic properties [32]. The corresponding evaluation would lead to $\varepsilon_f \leq n(n(c_1\varepsilon + c_2) + \varepsilon)$.

4 Implementation of FiLIP with TGSW

In this section we present our implemention of the transciphering of FiLIP with the TGSW scheme. First, we precise our selection of parameters and the settings of our implementation. Then, we analyze the noise obtained during the evaluation of FiLIP, and we compare it to the limit bound at which the decryption fails. We give the timings of the transciphering for the different filter choices, and compare these results with the transciphering from other works. Finally, we implement two evaluations of FiLIP with other 3G schemes implemented in TFHE, and compare the different options.

4.1 Selection of Parameters

Since the SE scheme has a bit-security of 128 we select the parameters to ensure the same security for the homomorphic scheme. Furthermore, we decide to fix the maximum probability of failure for the decryption at 2^{-128}, which is more restrictive that the usual choices in homomorphic evaluation. It is more coherent with the SE scheme which decryption is always correct, and it thwarts attacks using low decryption failure [19].

We use the estimator [2] for the concrete security of the LWE problem to fix the parameters k, N and α (α being the noise of a fresh cipher). More precisely for the TGSW scheme we estimate that the modulo for LWE (*i.e.* the coefficient ring) is equal to 2^{32}, since the TFHE library represents elements $x \in \mathbb{T}$ as the integers $\lfloor 2^{32}x \rfloor$ encoded on 32 bits. Fixing $k = 1$ the scheme relies more precisely on RLWE, but since there is no known attacks that are more efficient against RLWE than against LWE, we believe that this estimator is accurate for this scheme. Accordingly, we use the following parameters: $k = 1, N = 1024$ and $\alpha = 1\text{e}{-9}$, which ensures 128-bit security.

Then we choose two sets of decomposition parameters ℓ and B_g:

- Set 1: $k = 1, N = 1024, \alpha = 1\text{e}{-9}, B_g = 2^5, \ell = 6$.
- Set 2: $k = 1, N = 1024, \alpha = 1\text{e}{-9}, B_g = 2, \ell = 20$.

Using Proposition 1 we can determine the maximal variance allowed with these parameters: $2e^{(8B_g^2 V_{max})^{-1}} = 2^{-128} \iff V_{max} = (1032 B_g^2 ln(2))^{-1}$. It gives $V_{max} = 1.37\text{e}{-6}$ for the Set 1 and for the Set 2, $V_{max} = 3.49\text{e}{-5}$. As we will see in the following subsections, the first set of parameters gives better results in term of noise and speed, but the second one is useful for the comparison with the hybrid TGSW/TLWE scheme in Sect. 4.4.

4.2 Noise Evolution in Practice

We experimentally study the noise of the transciphering, it allows to compare it with the theoretical upper bounds of Proposition 6 which does not take into account the rounding error inhenrent in the representation of real numbers. In order to evaluate the concrete noise of the transciphering output ciphertexts, we implement a function that computes the noise of a ciphertext. This function behaves like the decryption one, except it does not round up to the closest message but compare to what a noiseless ciphertext would

be: It starts by isolating the coefficient in which we retrieve the message (the constant coefficient of the polynomial obtained by computing a scalar product with the $(k\ell+1)^{\text{th}}$ line). It then outputs the difference between this value and the closest $\frac{i}{B_g}$, which would be the message in the decryption algorithm.

We compute the average noises and variance noises over samples of at least 100 independent ciphertexts (we do not gain much precision on the variance with more samples). We give the resulting ciphertexts noise in Table 1, where the average noises are given normalized to the decryption limit (*i.e.* divided by $\frac{1}{2B_g}$, since decryption fails if the noise exceeds this limit). We do not present the evaluation of FiLIP-144 with the Set 2 because the noise variance growth is too important. We could get around this issue by allowing a slightly higher probability of decryption failure or by finding a better inequality to bound subgaussians variance. However the Set 2 is mainly presented to compare to the hybrid TLWE/TGSW scheme (which use this set of parameters), and we only implemented this hybrid scheme with DSM filters since they are more efficient.

Table 1. Transciphering noises of FiLIP over TGSW, comparison with different filters and sets of parameters for 128 bits security. The average noise represents the ciphertexts noise normalized by the maximum decryption noise. V'_{em} denotes the measured variance and V_{max} denotes the maximum variance defined above.

Filter	Parameters	Average noise	V'_{em}	V_{max}
FiLIP-1280	Set 1	1.17e−3	1.34e−10	1.37e−6
FiLIP-1216		2.18e−3	2.88e−10	
FiLIP-144		3.26e−3	7.19e−10	
FiLIP-1280	Set 2	9.22e−2	7.41e−6	3.49e−5
FiLIP-1216		1.75e−1	1.62e−5	

With this HE scheme, the noise evolution during the homomorphic product is asymmetrical, and as long as one of the operand is a fresh ciphertext the noise remains small. Unfortunately, when performing transciphering, we loose the ability to use fresh ciphertexts as operands. This limits the number of multiplications we can perform with our resulting ciphertexts (without bootstrapping) with these sets of parameters, allowing only operations over the $\mathfrak{R}$-module. Allowing more multiplications directly would require to increase the LWE modulus in order to get a better exit noise, which is not possible without in-depth changes to the TFHE library [14].

4.3 Performance and Comparisons

We provide the timings for the transciphering of FiLIP, for instances of DSM filter and XOR-threshold filter with the TGSW scheme. The timings in this section were obtained with a personal laptop with processor Intel(R) Core(TM) i5-6600K CPU @ 3.50 GHz. We summarize the results in Table 2, the latency corresponds to the homomorphic evaluation of FiLIP with TGSW, and the key size refers to the FiLIP secret key encrypted

with TGSW. One can notice that the key size is not a limiting factor since it is only used by the party doing the homomorphic computations, and is reasonable for this context.

Since FiLIP is a stream-cipher, its latency is the time per bit necessary to evaluate the decryption. For the DSM filters, FiLIP-1216 has the best latency: less than 2 s/bit for a transciphering of 128 bits security, FiLIP-1280 is slower but allows to use a smaller symmetric key. The XOR-Threshold filter has a competitive latency with the two DSM filters, which shows that other filters than DSM can be used for a similar efficiency. We give a comparison of the transciphering we implemented and former ones. The other works were all evaluated with a 2G scheme [5] on the HElib [24] library. We consider SE schemes of 128 bits security, using homomorphic parameters guaranteeing at least the same security. The transcipherings we compare correspond to the fastest variants: having the smallest ciphertexts allowing a correct decryption after transciphering. We summarize the results in Table 3. The security parameter in the table comes from the HElib library estimator for the first part of the schemes (source [29]), but the real security would be lower [1]. We did not find the security parameters for these timings of Kreyvium but we assume it is more than 128 bits, and the ones for our evaluations come from the estimation in Sect. 4.1.

Table 2. Transciphering timings of FiLIP over TGSW, comparison with 2 DSM filters (FiLIP-1280 and FiLIP-1216) and a XOR-threshold filter (FiLIP-144), with 2 sets of parameters for 128 bits security.

Filter	Set 1		Set 2	
	Latency (s)	Key size (Mo)	Latency (s)	Key size (Mo)
FiLIP-1280	2.2	200	22.7	800
FiLIP-1216	1.9	800	20.0	2680
FiLIP-144	2.5	800	–	–

Our transciphering has a better latency than all the existing implementations with a neat gain: homomorphic FiLIP-1216 decryption is 10 times faster than previous transcipherings. It shows that 3G implementation can lead to a very competitive latency. However, since TGSW does not support message batching yet, it has a way lower throughput than other transcipherings implemented on HElib, which are able to take advantage of homomorphic batching techniques to decrypt hundreds of bits at once, thus compensating the latency. Indeed, homomorphic batching consists in encrypting up to N plaintexts into a single homomorphic ciphertext. After batching encryption, the resulting ciphertext allows only homomorphic evaluation in an SIMD fashion, *i.e.* homomorphic operations perform the corresponding componentwise operations over the plaintext vector. As a consequence, transcipherings have a better bit-rate when batching is supported, but using it limits homomorphic evaluations to vectorized operations and rotations via the Frobenius endomorphism.

Table 3. Performances for minimal FHE parameters, 128 bits security. The latency refers to the time in seconds required to obtained the first homomorphic ciphertext, λ' is the estimated security parameter of the HE scheme, and Source refers to the article presenting these timings.

Cipher	Latency (s)	λ'	Source
LowMCv2(14, 63, 256)	1629.3	132.1	[29]
FLIP-1394	26.53	146.8	
Agrasta (129, 4)	20.26	134.7	
Rasta (525, 5)	277.24	128.9	
Rasta (351, 6)	195.40	128.9	
FiLIP-1216	24.37	186.3	
FiLIP-1280	26.59	146.8	
Kreyvium-12	1547.0		[9]
FiLIP-1216	1.9	157	This work
FiLIP-144	2.5	157	

4.4 Different Homomorphic Evaluations of FiLIP

We implement two other evaluations of FiLIP with 3G schemes. First we use the gate-bootstrapping of TFHE. In this scheme since the noise in not important anymore due to the bootstrapping at each computation the efficiency of a computation can be inferred by the number of gates in its circuit. The other scheme we implement is inspired by the automata circuit in [11]. The idea is to start the FiLIP evaluation with TGSW ciphertexts and a set of parameters such that $B_g = 2$, and switch to TLWE ciphertexts when evaluating the filter. Indeed, the two evaluations of Sect. 3.2 are possible with at most one of the operands being a TGSW at each step of the evaluation. The main idea is to extract a TLWE ciphertext at the beginning of the filter evaluation, and perform external product to get the resulting ciphertext. The transciphering outputs a TLWE ciphers, which means we have a simply homomorphic scheme, we cannot perform products between two such ciphers. This method guarantees good timings since it reduces the number of external products performed, but a bootstrapping to a leveled scheme is necessary for further computations.

We compare the timings of these different approaches, and the evaluation of FiLIP with HElib in Table 4. Evaluating FiLIP-1280 with this scheme gives cipher with the same noise as in the TGSW scheme with the second set of parameters. This scheme allows us to compute transciphering with our best timings: **1018 ms/b**. It is around twice better than with the TGSW but does not allow as many operations, a bootstrapping could be used to compensate it but the final timing would be less competitive. The timing we present for the FHE scheme provided by the TFHE library [14] comes from an implementation we wrote to verify that our leveled scheme is faster in practice. We measured that the average time to compute a gate with bootstrapping on our computer is about 20 ms instead of the 13 ms in [13], which is coherent with the total evaluation time. This approach gives a similar latency as the evaluation on HElib, therefore we can conclude from these timings that the TGSW scheme alone or with TLWE are more

efficient to transcipher FiLIP cipher with a DSM filter. However, using TFHE gives us fully homomorphic ciphers on which we can perform any operation.

Table 4. Transciphering timings of FiLIP over different HE schemes.

HE scheme	FiLIP-1280	FiLIP-1216
TGSW	2.2	1.9
TGSW/TLWE	1.2	1.0
TFHE	25.8	22.7
BGV(HElib)	26.7	24.4

5 Conclusion

In this paper, we presented the first implementation of transciphering with a third generation homomorphic scheme, and give a transciphering with the smallest latency so far: less than 2 s. We have also implemented for the first time the XOR-threshold filter variant of FiLIP, showing competitive timings with the DSM variant. We compared FiLIP evaluation with three different HE schemes which showed that the TGSW scheme was more appropriate in this case. Despite a large improvement for the latency, the throughput is not competitive with transciphers implemented on HElib which compensate by the number of bits batched in each ciphertext.

A possible improvement would be to batch messages by encoding them in all the coefficients of our message polynomial. This would mean we could store N bits of message in a ciphertext instead of 1, which would be a huge gain. In order to implement batching, one could look on the strategies developed in [12] that describes different batching methods for a leveled HE, and try to adapt it for Boolean functions such as the filters we consider.

Acknowledgments. This work has been funded in part by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701), and by the French RISQ project (BPI-France, grant P141580). This work has been funded in part by the European Union (EU) and the Walloon Region through the FEDER project USERMedia (convention number 501907-379156). This work has been funded in part by the European Union (EU) through the ERC project 724725 (acronym SWORD). Pierrick Méaux is funded by a F.R.S. Incoming Post-Doc Fellowship.

References

1. Albrecht, M.R.: On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 103–129. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_4
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046 (2015)

3. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 430–454. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_17
4. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. IACR Crypt. ePrint Arch. 687 (2016)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012, pp. 309–325. ACM, January 2012
6. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd FOCS, pp. 97–106. IEEE Computer Society Press, October 2011
7. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: Naor, M. (ed.) ITCS 2014, pp. 1–12. ACM, January 2014
8. Canteaut, A., et al.: Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 313–333. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_16
9. Canteaut, A., et al.: Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression. J. Cryptol. **31**, 885–916 (2018)
10. Chen, H., Laine, K., Player, R.: Simple encrypted arithmetic library - seal v2.1. Cryptology ePrint Archive, Report 2017/224 (2017)
11. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 3–33. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_1
12. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 377–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_14
13. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**, 34–91 (2020). https://doi.org/10.1007/s00145-019-09319-x
14. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption library, August 2016. https://tfhe.github.io/tfhe/
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). https://doi.org/10.1007/978-3-642-54631-0_18
16. Dobraunig, C., et al.: Rasta: a cipher with low ANDdepth and few ANDs per bit. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 662–692. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_22
17. 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. LNCS, vol. 8438, pp. 208–220. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44774-1_17
18. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_24
19. D'Anvers, J.-P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I.: Decryption failure attacks on IND-CCA secure lattice-based schemes. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 565–598. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_19
20. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012)

21. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC, pp. 169–178. ACM Press, May/June 2009
22. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_49
23. 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.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
24. Halevi, S., Shoup, V.: Algorithms in HElib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 554–571. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_31
25. 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, Cham (2014). https://doi.org/10.1007/978-3-319-06734-6_20
26. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Karloff, H.J., Pitassi, T. (eds.) 44th ACM STOC, ACM Press, May 2012
27. Méaux, P.: On the fast algebraic immunity of majority functions. In: Schwabe, P., Thériault, N. (eds.) LATINCRYPT 2019. LNCS, vol. 11774, pp. 86–105. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30530-7_5
28. Méaux, P., Carlet, C., Journault, A., Standaert, F.: Improved filter permutators: combining symmetric encryption design, Boolean functions, low complexity cryptography, and homomorphic encryption, for private delegation of computations. Cryptology ePrint Archive, Report 2019/483 (2019)
29. Méaux, P., Carlet, C., Journault, A., Standaert, F.-X.: Improved filter permutators for efficient FHE: better instances and implementations. In: Hao, F., Ruj, S., Sen Gupta, S. (eds.) INDOCRYPT 2019. LNCS, vol. 11898, pp. 68–91. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35423-7_4
30. Méaux, P., Journault, A., Standaert, F.-X., Carlet, C.: Towards stream ciphers for efficient FHE with low-noise ciphertexts. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 311–343. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_13
31. 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, 21 October 2011, pp. 113–124 (2011)
32. Wang, Q., Carlet, C., Stănică, P., Tan, C.H.: Cryptographic properties of the hidden weighted bit function. Discret. Appl. Math. **174**, 1–10 (2014)