Towards Case-Optimized Hybrid Homomorphic Encryption

Featuring the Elisabeth Stream Cipher

Orel Cosseron[1 2 3 ⋆], Clément Hoffmann[4], Pierrick Méaux[5], François-Xavier Standaert[4]

[1] INRIA Lyon, Lyon, France
[2] ENS de Lyon, LIP, Lyon, France
[3] LTCI, Telecom Paris, Institut Polytechnique de Paris, Paris, France
[4] UCLouvain, ICTEAM/ELEN/Crypto Group, Belgium
[5] Luxembourg University, SnT, Luxembourg

Abstract. Hybrid Homomorphic Encryption (HHE) reduces the amount of computation client-side and bandwidth usage in a Fully Homomorphic Encryption (FHE) framework. HHE requires the usage of specific symmetric schemes that can be evaluated homomorphically efficiently. In this paper, we introduce the paradigm of Group Filter Permutator (GFP) as a generalization of the Improved Filter Permutator paradigm introduced by Méaux *et al*. From this paradigm, we specify Elisabeth , a family of stream cipher and give an instance: Elisabeth-4 . After asserting the security of this scheme, we provide a Rust implementation of it and ensure its performance is comparable to state-of-the-art HHE. The true strength of Elisabeth lies in the available operations server-side: while the best HHE applications were limited to a few multiplications server-side, we used data sent through Elisabeth-4 to homomorphically evaluate a neural network inference. Finally, we discuss the improvement and loss between the HHE and the FHE framework and give ideas to build more efficient schemes from the Elisabeth family.

1 Introduction

State-of-the-art. Hybrid Homomorphic Encryption (HHE) is a powerful solution to limit the performance overheads and ciphertext expansion that a direct application of Fully Homomorphic Encryption (FHE) on private data causes. Rather than directly encrypting private data homomorphically, its high-level idea is to encrypt a symmetric key with the (expensive) FHE scheme and send it to the server. Next, a client can send private data encrypted with the symmetric cipher to the server, which will homomorphically carry out symmetric decryption before performing the private computations [36].

While HHE can in principle be instantiated with any symmetric cipher, it was rapidly observed that standard ciphers like the AES may not be the best for this purpose [23,16]. Researchers therefore started to investigate the improved performances that specialized ciphers (e.g. with limited multiplicative complexity and/or depth) can

⋆ Part of this work was done while the first author was working in Zama, Paris, France.

lead to. Popular examples of such ciphers include LowMC [2,3], Kreyvium [5], FLIP & FiLIP [35,34] or RASTA & DASTA [19,29].

As a first step, these ciphers have been optimized towards making their homomorphic decryption as efficient and low-noise as possible, independent of the applications. Recent benchmarks like [20] exhibit that parallel ciphers (e.g. LowMC , RASTA , or DASTA) can reach slightly better throughputs than serial ciphers (e.g. Kreyvium , FLIP or FiLIP) at the cost of higher latency. Yet, while all these ciphers lead to practical performances for their homomorphic decryption, it remains that they are specified in $\mathbb{F}_2$ which, despite being convenient for symmetric cryptography, is not ideal when considering the global optimization goal of performing signal processing on encrypted data. For example, standard machine learning (e.g. classification) algorithms operate on reals or integers, which can then be quantized in $\mathbb{Z}_q$ with preferably large q's. Hence, applying HHE with ciphers specified in $\mathbb{F}_2$ requires expensive conversions from $\mathbb{F}_2$ to $\mathbb{Z}_q$.

Contributions. Building on this state-of-the-art, the general goal of this paper is to show that the optimization of HHE can benefit from considering the full toolchain going from the symmetric cipher to the data processing to perform homomorphically. For this purpose, we first aim to design a symmetric encryption scheme that not only allows efficient (stand-alone) homomorphic decryption but is also well suited for practically-relevant applications such as classification with machine learning algorithms. We next aim to put forward the tradeoff between the HHE performances and the constraints that it implies for machine learning classification.

We first generalize the filter permutator design approach used by the FLIP and FiLIP stream ciphers to arbitrary groups and instantiate this new construction by specifying the Elisabeth stream cipher family. Besides the goal of enabling homomorphic computations in $\mathbb{Z}_q$ without expensive conversions, this new design approach is driven by the recent work of Chillotti et al. [11], which generalizes the TFHE scheme from [10] in order to enable efficient Programmable BootStrapping (PBS). PBS is pushing for ciphers that combine (negacyclic) Look Up Tables (LUTs) and additions in $\mathbb{Z}_q$ with q a power of 2. We select $q = 2^4$ for our first instance of Elisabeth , hence denoted Elisabeth-4 , by analyzing the current cost of a PBS as a function of the LUT sizes. Overall, Elisabeth-4 only requires two levels of PBS and a few additions in $\mathbb{Z}_q$, therefore enabling a very efficient homomorphic decryption. We study the security of the group filter permutator approach and our proposed instantiations. We also show that the (still stand-alone) performances of Elisabeth-4 compare well with (and sometimes improve) the aforementioned state-of-the-art ciphers optimized for FHE.

Next, we discuss the application of HHE to a classification based on a simple neural network. As a first step in this direction, we use the Fashion-MNIST dataset for this purpose [40]. It contains 28x28 grayscale images of 70,000 fashion products from 10 categories. This case study allows us to benchmark a full protocol, including the (one-time) key exchange, the homomorphic (symmetric) decryption, the homomorphic classification, and the result decryption. It confirms the interest in the HHE approach compared to the direct application of FHE and the interest of Elisabeth-4 over binary ciphers (in particular FLIP and FiLIP which share a similar design approach). It also

emphasizes the tradeoff between the accuracy of the machine learning classification and the HHE performances.

We conclude the paper with a critical discussion of its limitations, which allows us to identify important directions for further research. In particular, we put forward (*i*) how improving TFHE to enable efficient PBS for larger LUTs could improve the accuracy of machine learning classifications with larger instances of Elisabeth , and (*ii*) how public preprocessing and other optimizations could improve the efficiency of homomorphic classification.

Related works. The observation that non-binary ciphers could be beneficial to HHE has also been made in other recent works. In [27], the authors describe MASTA a variant of RASTA adapted to $\mathbb{Z}_q$ where $q = 2^{16} + 1$. In [13], the authors describe the cipher HERA which is defined over $\mathbb{Z}_q$ with $q > 2^{16}$. In [28], the authors describe the RUBATO cipher, adapted to $\mathbb{Z}_q$ where $q \approx 2^{25}$. In [20], the authors describe the family of ciphers PASTA which is defined over $\mathbb{F}_p$ with p a 16-bit prime. They also discuss how it can be combined with the homomorphic computation of a 5x5 matrix-vector multiplication. Their main difference with our proposal is the need for a larger modulus, which is not compatible with the efficient PBS of the generalized TFHE scheme we target. As a result, they are not ideal for application to our machine learning classification problem with this scheme.

We insist that we make no claim regarding the possibility to adapt HERA , PASTA , or other FHE-friendly ciphers to our case study. In particular, some specific choices of parameters such as the selection of $q = 2^4$, are motivated by current technological constraints (which also motivates the group generalization of filter permutators). Instances of Elisabeth with larger q's (discussed in conclusion) or HERA -like & PASTA -like ciphers with smaller moduli will admittedly look more and more similar, and it is an interesting long-term problem to determine which cipher to use for which range of applications. By contrast, our claim is that taking into account the data processing necessary to perform homomorphic operations when designing an FHE-friendly cipher can lead to improved global performances, as the comparison between Elisabeth-4 and FiLIP showcases. In this respect, we believe focusing on machine learning algorithms is a relevant direction since they typically raise important privacy concerns [37]. Yet, since the comparative performances of HHE schemes may be application-dependent, our results also suggest the identification of relevant case studies for benchmarking as an interesting topic of discussion for the FHE research community.

2 Background

General notations. We use $[n]$ to denote $\{1, \ldots, n\}$ and more generally $[a, b]$ for the set of integers c such that $a \leq c \leq b$. We will use the notation $\mathbb{G}$ to denote a group with operation $+$, 0 for its neutral element, and $-$ to denote the inverse. For a vector $a \in \mathbb{G}^n$ we denote $\mathsf{w_H}(a)$ its Hamming weight: $\mathsf{w_H}(a) = |\{a_i \neq 0, i \in [n]\}|$. For two vectors $a, b \in \mathbb{G}^n$ we denote $\mathsf{d_H}(a)$ their Hamming distance: $\mathsf{d_H}(a, b) = \mathsf{w_H}(a - b)$. $\log$ refers to the base-2 logarithm.

2.1 TFHE

By definition, a fully homomorphic encryption scheme can evaluate any arbitrary operation on its ciphertexts. While this is technically true, depending on the FHE scheme, some operations will be way easier to compute than others. The efficiency of an HHE scheme therefore deeply relies on how the symmetric cipher can use the efficient operations of the FHE scheme.

The Elisabeth stream cipher family has been conceived to take advantage of the efficient operations of the FHE scheme TFHE [10] and implemented using the operations available in the Concrete library [9].

In this section, we recall the core concepts of TFHE needed to understand our contributions, for a detailed explanation of TFHE operations we refer the reader to [9], from which we took most of the notations.

Encryption and decryption. The encryption scheme at the core of TFHE[6] is based on Learning With Error (LWE). A variant relying on the General Learning With Error (GLWE) assumption introduced in [4] is used to perform some homomorphic operations. The generality of GLWE allows encompassing constructions relying on LWE [38], Ring LWE (RLWE) [32] or trade-offs between both under the same structure [31].

Let $\mathbb{B}$ be the set $\{0, 1\}$, $q = 2^p$, $\Delta = 2^\delta$ and $N = 2^d$ be three powers of two. We note $\mathbb{B}_N[X] := \mathbb{B}[X]/\langle X^N + 1\rangle$ and $\mathbb{Z}_{q,N}[X] := \mathbb{Z}_q[X]/\langle X^N + 1\rangle$.

Definition 1 (GLWE **ciphertexts [4]).** *An encryption $\mathbf{c}$ of a message $\mu \in \mathbb{Z}_{q/\Delta,N}[X]$ under secret key $\mathbf{s} := (s_1(X), \ldots, s_k(X)) \in \mathbb{B}_N[X]^k$ is given by $\mathbf{c} \in (\mathbb{Z}_{q,N}[X])^{k+1}$ such that:*

$$\mathbf{c} := \left(a_1(X), \ldots, a_k(X), b(X) := \sum_{i=1}^{k} a_i(X) \cdot s_i(X) + \Delta \cdot \mu(X) + \varepsilon(X) \right)$$

where the coefficients of the a_i are uniformly sampled over $\mathbb{Z}_q$ and the ones of ε are sampled from a small discretized Gaussian noise.

We call $\mathbf{a} := (a_1, \ldots, a_n)$ the **mask** and b the **body** of the ciphertext. Given a ciphertext $\mathbf{c}$ and the secret key $\mathbf{s}$, one decrypts $\mathbf{c}$ by computing $\mu^*(X) := b(X) - \sum_{i=1}^{k} a_i(X) \cdot s_i(X) = \Delta \cdot \mu(X) + \varepsilon(X)$.

By choosing Δ such that $\Delta > ||\varepsilon||_\infty$, one can recover μ from μ^* by dividing each coefficient by Δ (which is equivalent to shifting each coefficient δ bits to the right). This means that the value of δ fixes the noise budget, that is the maximal number of bits one allows the coefficients of ε to be written on. So, for a given value of q, a bigger noise budget means a smaller message size.

In the following of the paper, we refer to LWE for a GLWE ciphertext with $N = 1$ and to RLWE for a GLWE ciphertext with $k = 1$. The LWE ciphertexts are the core of the TFHE scheme, whereas the other GLWE ciphertexts only play a role in the product and bootstrapping.

[6] i.e. Regev's encryption scheme [38]

Addition and multiplication. GLWE ciphertexts encrypted under the same secret key can be added coefficient-wise, the result of which gives an encryption of the sum of the plaintexts in $\mathbb{Z}_{q,N}[X]$. By iterating, it is then possible to multiply a ciphertext by a plain scalar. Note that, in both cases, the noises increase, resulting in linear growth of the noise inside the ciphertext.

To perform the multiplication of a GLWE ciphertext by an encrypted (polynomial) constant while managing the noise growth, TFHE uses another kind of ciphertext. These ciphertexts are the generalization of the GSW scheme [24] corresponding to GLWE, hence dubbed General GSW ciphertexts. Then, an external product between GGSW and GLWE ciphertexts allows the multiplication between encrypted plaintexts.

Definition 2 (GGSW **Ciphertexts).** *Let $B = 2^b \in \mathbb{N}$ and $\ell \in \mathbb{N}$. A* GGSW *ciphertext $\mathbf{C}$ of a message $m \in \mathbb{Z}_{q,N}[X]$ under secret key $\mathfrak{s} := (\mathfrak{s}_1, \ldots, \mathfrak{s}_k) \in \mathbb{B}_N[X]^k$ is defined as:*

$$\mathbf{C} = \left(\mathrm{GLWE}_{\mathfrak{s}}(-\mathfrak{s}_i \tfrac{q}{B^j} m)\right)_{(i,j)\in[k+1]\times[\ell]} \in (\mathbb{Z}_{q,N}[X])^{\ell(k+1)\times(k+1)},$$

with $\mathfrak{s}_{k+1} = -1$.

We call B the basis and ℓ the number of levels of the ciphertext $\mathbf{C}$. A GSW (resp. RGSW) ciphertext is a GGSW ciphertext made from LWE (resp. RLWE).

External Product. The external product between GGSW and GLWE ciphertexts gives a GLWE ciphertext of the product of the input plaintexts. The external product of a GGSW ciphertext $\mathbf{C}$ and a GLWE ciphertext $\boldsymbol{c}$ is defined as $\mathbf{C} \boxdot \boldsymbol{c} = \mathrm{Decomp}(c) \cdot \mathbf{C}$, where Decomp is a transformation that flattens a vector of $k+1$ polynomials into a vector of $\ell(k+1)$ polynomials with coefficients in $[-B/2; B/2]$.

Programmable boostrapping. Most of the homomorphic operations make the noise grow. In order to achieve fully homomorphic encryption, it is necessary to have an operation to decrease the noise. This is what bootstrapping is used for, by resetting the noise to a level independent of the noise level in the input ciphertext. In addition to noise control, TFHE's bootstrapping on LWE ciphertexts, also noted PBS for Programmable BootStrapping, allows the evaluation of a negacyclic look-up table L. Instead of giving a ciphertext of μ, it directly gives a bootstrapped ciphertext of $L[\bar{\mu}]$, where $\bar{\mu}$ is a noisy rescaling of μ on $[0, 2N-1]$ and N is determined by the GLWE scheme used. More information on PBS can be found in Appendix A.

Definition 3 **(Negacyclic Look-Up Table** (NLUT)**).** *A negacyclic look-up table over a group $\mathbb{G}$ is a look-up table L of length $2N$ that verifies the following property:*

$$\forall i \in [0, N-1], L[i+N] = -L[i] \in \mathbb{G},$$

where $=$ and $-$ denote equivalence and inverse defined on $\mathbb{G}$.

Keyswitch. The ciphertext output by the bootstrapping might be encrypted under a different, possibly larger, key than the input ciphertext. The keyswitching operation allows to convert a ciphertext encrypted under $\boldsymbol{s}_{in}$ into a ciphertext encrypted under $\boldsymbol{s}_{out}$ thanks to a keyswitching key $\mathbf{K}$. $\mathbf{K}$ is a GGSW encryption of 1, up to two details:

1. The last ℓ rows of the matrix are cropped;
2. The ciphertexts composing each line are encrypted under $\boldsymbol{s}_{out}$ rather than $\boldsymbol{s}_{in}$.

The keyswitching of $\boldsymbol{c} = (\boldsymbol{a}, b)$ is then computed as $(\mathbf{0}, b) - \mathrm{Decomp}(\boldsymbol{a}) \cdot \mathbf{K}$. It allows switching back to the scheme with the first key.

2.2 Boolean functions and cryptographic criteria

In this part, we introduce Boolean functions and relevant cryptographic criteria (taken from [6]) that we will use to study the security of the main stream-cipher construction we propose in this article. Then we generalize the classic cryptographic criteria on Boolean functions to functions from $\mathbb{G}^n$ to $\mathbb{G}$ where $\mathbb{G}$ is a group.

Definition 4 ((Vectorial) Boolean function). *An (n, m) vectorial Boolean function F is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. When $m = 1$ the $(n, 1)$ vectorial Boolean function is simply referred to as a Boolean function and we denote the space of n-variable Boolean function as $\mathcal{B}_n$. We call coordinate functions of F the m Boolean functions f_i associating for each $x \in \mathbb{F}_2^n$ the i-th binary output of $F(x)$. We call component functions of F the $2^m - 1$ non-trivial linear combinations of the coordinate functions of F.*

Definition 5 (Algebraic Normal Form (ANF) and degree). *We call Algebraic Normal Form of a Boolean function f its n-variable polynomial representation over $\mathbb{F}_2$ (i.e. belonging to $\mathbb{F}_2[x_1, \ldots, x_n]/(x_1^2 + x_1, \ldots, x_n^2 + x_n)$):*

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

where $a_I \in \mathbb{F}_2$. The algebraic degree of f equals the global degree of its ANF: $\mathsf{deg}(f) = \max_{\{I \mid a_I = 1\}} |I|$ (with the convention that $\mathsf{deg}(0) = -\infty$).

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

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

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

Definition 7 (Balancedness and Resiliency). *A Boolean function $f \in \mathcal{B}_n$ is said to be balanced if its output is uniformly distributed, that is: $|\{x \mid f(x) = 0\}| = |\{x \mid f(x) = 1\}|$.*

A Boolean function f is called be m-resilient if any of its restrictions obtained by fixing at most m of its coordinates is balanced. We denote by $\mathsf{res}(f)$ the maximum resiliency (also called resiliency order) m of f and set $\mathsf{res}(f) = -1$ if f is unbalanced.

Definition 8 (Nonlinearity). *The nonlinearity* NL *of an* n*-variable Boolean function* f*, where* n *is a positive integer, is the minimum Hamming distance between* f *and all the affine functions in* $\mathcal{B}_n$*:*

$$\mathsf{NL}(f) = \min_{g,\, \mathsf{deg}(g) \leq 1} \{d_H(f, g)\},$$

with $d_H(f, g) = \#\{x \in \mathbb{F}_2^n \mid f(x) \neq g(x)\}$ *the Hamming distance between* f *and* g*; and with* $g(x) = a \cdot x + \varepsilon$*,* $a \in \mathbb{F}_2^n, \varepsilon \in \mathbb{F}_2$ *(where* $\cdot$ *is some inner product in* $\mathbb{F}_2^n$*; any choice of an inner product will give the same definition).*

Additionally, we denote NL^d *the order-*d *nonlinearity of* f*, the minimum Hamming distance between* f*, and all the functions of degree at most* d*.*

Definition 9 (Direct Sum). *Let* f *be a Boolean function of* n *variables and* g *a Boolean function of* m *variables,* f *and* g *depending on distinct variables, the direct sum* h *of* f *and* g *is defined by:*

$$h(x, y) = \mathsf{DS}(f, g) = f(x) \oplus g(y), \quad \text{where } x \in \mathbb{F}_2^n \text{ and } y \in \mathbb{F}_2^m.$$

Additionally we denote $\mathsf{DS}^t(f)$ *when is the direct sum of* t *times the function* f*.*

Extending to groups. The notions defined above on Boolean functions can easily be extended to functions from $\mathbb{G}^n$ to $\mathbb{G}$, we denote these extended notions by the subscript $\mathbb{G}$.

Definition 10 (Cryptographic criteria over $\mathbb{G}$). *For a function* f *from* $\mathbb{G}^n$ *to* $\mathbb{G}$ *we denote:*

- $\mathsf{deg}_{\mathbb{G}}(f)$ *the degree over* $\mathbb{G}$*. It corresponds to the minimum degree over the polynomial representations of* f *in the polynomial ring* $(\mathbb{G}, \cdot)[x_1, \ldots, x_n]$ *when such representations exist.*
- $\mathsf{res}_{\mathbb{G}}(f)$ *the resiliency order over* $\mathbb{G}$*.* f *is balanced if and only if:*

  $$\forall a \in \mathbb{G} : \ |\{x \,|\, f(x) = a\}| = |\mathbb{G}|^{n-1}.$$

  f *is* m*-resilient if all the sub-functions obtained by fixing up to* m *variables are balanced.*
- $\mathsf{NL}_{\mathbb{G}}(f)$ *the nonlinearity over* $\mathbb{G}$*. The nonlinearity is taken as the minimum Hamming distance between* f *and the affine functions:* $a_0 + \sum_{i=1}^n a_i x_i$ *where the* a_i *describe* $\mathbb{G}^{n+1}$*. Additionally,* $\mathsf{NL}_{\mathbb{G}}^d(f)$ *denotes the order-*d *nonlinearity over* $\mathbb{G}$*.*

We also denote $\mathsf{DS}_{\mathbb{G}}(f, g)$ the direct sum $f(x) + g(y)$ where $x \in \mathbb{G}^n$ and $y \in \mathbb{G}^m$ and f and g are two n-variable and m-variable functions defined on distinct variables.

3 Group filter permutator & Elisabeth-4

The improved filter permutator (IFP) paradigm [34] led to binary stream ciphers efficient for HHE [35,34,30] where the homomorphic evaluation (of the stream cipher decryption) is reduced to the evaluation of a unique filtering function. Its connection with

Goldreich's pseudorandom generator [25], the different security analyses [35,15,34] and cryptanalysis studies [21,8], built confidence in the soundness of this design. They make it a natural starting point to design HHE with a non-binary symmetric scheme.

Section 3.1 introduces the group filter permutator which generalizes the IFP paradigm over $\mathbb{F}_2$ to any group. We then move towards the instantiation of group filter permutators. For this purpose, we observe that concrete choices of parameters are driven by technological constraints (e.g., operations that are efficient in TFHE (C)). Section 3.2 discusses these choices and explains how future advances in the Concrete implementation could benefit the stream cipher. The discussion leads to the design of the stream cipher family Elisabeth and an instantiation in Section 3.3 (full algorithm are detailed in Appendix B.2). Finally, the performances of the transciphering alone are provided in Section 3.4.

3.1 Design and stream cipher families

The group filter permutator is defined by a group $\mathbb{G}$ with operation noted $+$, a forward secure PRNG, a key size N, a subset size n, and a filtering function f from $\mathbb{G}^n$ to $\mathbb{G}$. To encrypt m elements of $\mathbb{G}$ under a secret key $K \in \mathbb{G}^N$, the public parameters of the PRNG are chosen and then the following process is executed for each key-stream s_i (for $i \in [m]$):

- The PRNG is updated, its output determines a subset, a permutation, and a length-n vector of $\mathbb{G}$.
- the subset S_i is chosen, as a subset of n elements over N,
- the permutation P_i (a re-ordering) from n to n elements is chosen,
- the vector, called whitening and denoted by w_i, from $\mathbb{G}^n$ is chosen,
- the key-stream element s_i is computed as $s_i = f(P_i(S_i(K)) + w_i)$, where $+$ denotes the element-wise addition of $\mathbb{G}$.

The GFP, depicted in Figure 1, is a generalization of the improved filter permutator [34] where $\mathbb{G} = \mathbb{F}_2$. The XOR is replaced by the addition of $\mathbb{G}$ and the Boolean function by a function from $\mathbb{G}^n$ to $\mathbb{G}$.

3.2 Look-up tables specifications

In order to be efficiently evaluated homomorphically, Elisabeth filters should rely on the most native TFHE homomorphic functions, which are modular additions and NLUTs (see Definition 3) evaluation. The use of NLUTs leads to some non-standard constraints for the design of Elisabeth . In particular, they are entirely determined by the first half of their values and cannot be bijective. Indeed, 0 can only have an even number of antecedents: given a 2^n size NLUT L, for each $i \in [0, 2^{n-1} - 1]$ such that $L[i] = 0$, we also have $L[i + 2^{n-1}] = -L[i] = 0$. As will be shown next, those peculiarities do not raise fundamental problems for the security of the cipher and it is possible to reach the necessary cryptographic criteria to ensure 128 bits of security with a cipher based on NLUTs and modular additions.

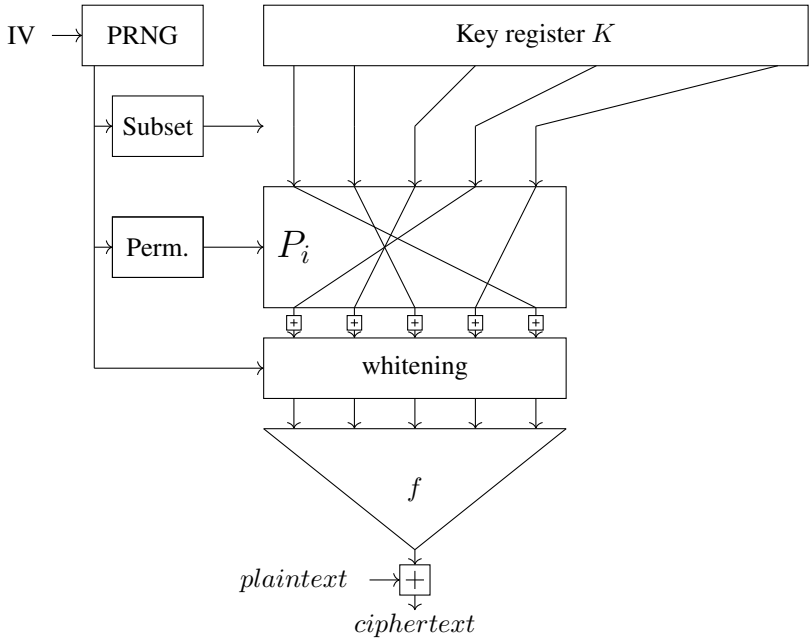


Fig. 1: The group filter permutator design

On the choice of the NLUTs size. The choice of the size of the NLUTs is driven by several conflicting arguments: on the one hand, smaller NLUTs increase the amount of NLUTs that needs to be evaluated to reach a given level of security, which therefore slows down the evaluation of the circuit. On the other hand, bigger NLUTs lead to heavier computation: indeed, the NLUTs are homomorphically computed through a PBS, which is essentially a sequence of polynomial multiplications. The bigger the NLUT, the bigger the polynomials involved in the PBS. Since the cost of a polynomial multiplication is of $\mathcal{O}(N \log N)$, where N is the polynomial degree, there is a clear interest in taking smaller NLUTs. That being said, Elisabeth's final goal is to ensure the evaluation of neural networks over the output of the transciphering. This goal is more easily achieved if the output ciphertext contains bigger chunks of data. To find out where the trade-off between smaller polynomials and bigger chunks of data lies, we designed Table 1 which gives the minimal size of polynomial needed to evaluate a NLUT of given size for different LWE dimensions in TFHE. These values are hard minimums computed before any specific choice of design for the encryption scheme, i.e. if we wanted to evaluate a NLUT on a ciphertext that contains just enough noise to ensure 128 bits of security. Every operation in the scheme performed before a PBS could increase the noise in the ciphertext up to a point where bigger polynomials are needed for the same NLUT size. Considering all the previous arguments, we found that NLUTs of **4** bits would be the best trade-off as it relaxes as much as possible the constraint on polynomial sizes while keeping enough bits of message in a single ciphertext. We chose not to use bigger messages for several reasons: first, this guarantees that once

the specifics of our design have been set, the polynomial degree should remain small. Second, this ensures that Elisabeth-4 will require a smaller noise budget to be evaluated than any computation that could be performed afterward. By doing so, one ensures that transciphering will not be the noisiest part of the evaluation server-side, and thus can be computed with exactly the same parameters as the following circuit. In a nutshell, in the vast majority of usecases, adding Elisabeth-4 before the evaluation of a circuit will not require new parameters to be computed. With the current limitations of TFHE, using a bigger length for the NLUTs could not only multiply the transciphering time by more than two but also slow down the whole evaluation server-side. Nevertheless, the generality of the design enables it to adapt easily to powers of two of the same order once future improvement of TFHE regarding big(ger) integer computation would make it worthwhile to use bigger NLUTs.

LWE dimension NLUT length	**512**	**630**	**650**	**688**	**710**	**750**	**800**	**830**	**1024**	**2048**	**4096**
16	512	512	512	512	512	512	512	512	512	1024	1024
32	1024	1024	1024	1024	1024	1024	1024	1024	1024	2048	2048
64	2048	2048	2048	2048	2048	2048	2048	2048	2048	4096	4096
128	4096	4096	4096	4096	4096	4096	4096	4096	4096	8192	8192
256	8192	8192	8192	8192	8192	8192	8192	8192	8192	16384	16384

Table 1: Polynomial degrees required to evaluate NLUTs containing bigger elements, in function of the LWE dimension. The LWE dimension is the number of elements in the mask of a ciphertext. The NLUT length is both the number of elements in the NLUT and the size of the group $\mathbb{G}$ over which the NLUT is defined.

3.3 Elisabeth-4

In order to benefit from Concrete [9] efficient operations, the design of Elisabeth is a GFP defined over $\mathbb{G} = \mathbb{Z}_{2^\ell}$ where the filtering function f is composed by additions over $\mathbb{G} = \mathbb{Z}_{2^\ell}$ and NLUTs that could be evaluated in parallel. A simple way to follow these guidelines and allow a simpler security analysis is to make the filter f rely on the direct sum construction: f is the addition (over $\mathbb{G}$) of functions acting on independent variables. To be even simpler, each sub-function of f can be chosen to be the same function, which gives a highly parallelizable design. Accordingly, splitting n in t parts of m (different) variables, Elisabeth's filters have the following shape:

$$f(x_1, \ldots, x_n) = \mathsf{DS}^t_{\mathbb{G}}(g(x_1, \ldots, x_m)).$$

The general idea of Elisabeth 's filters design is to use the fewest levels possible of NLUT, in order to parallelize computation efficiently.

At the moment, as explained in Section 3.2, Concrete's implementation leads to fix $\mathbb{G} = \mathbb{Z}_{16}$. Only one level of NLUTs of this size would not guarantee a secure scheme

for 128-bit security[7]. Hence, for this NLUT size we propose a filter with two levels of NLUTs, following the 5-to-1 construction depicted in Figure 2. The high-level idea of this construction is to mix 4 variables to get strong enough algebraic properties and to use the fifth one to balance the output.

It gives the following instantiation of Elisabeth-4 : $\mathbb{G} = \mathbb{Z}_{16}$, $N = 256$, $n = 60$, $t = 12$, $m = 5$, and g is the 5-to-1 function. As will be shown in Section 4, a random choice of NLUTs has a high probability to lead to secure instances of Elisabeth . Therefore, and to mitigate the possibility of trapdoor the design by targeting particular relations over $\mathbb{G}$, we selected the 8 NLUTs required for g by hashing the sentence "Welcome to Elisabeth, heir of FiLIP!", using the following process[8]:

- The sentence is encoded in UTF-8 and the 256 bits hash is computed using SHA-2. Those 256 bits are then split into 8 blocks of 32 bits. Each NLUT S_i is determined using the i-th block.
- The 32 bits are themselves split in 8 blocks of 4 bits, which directly gives the 8-th first coefficients of the size-16 NLUT, which determines entirely the NLUT.

All the NLUTs obtained with this process are explicitly given in Appendix B.1. We analyze this particular choice of NLUTs in Section 4.

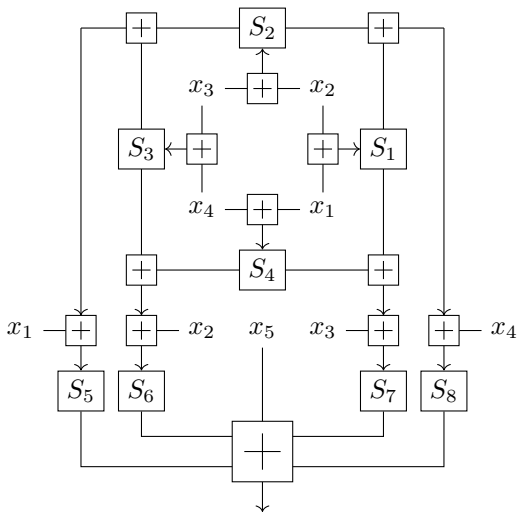


Fig. 2: Elisabeth-4 's 5 to 1 inner function.

[7] Based on the analysis in Section 4, an adversary could retrieve the key by solving an algebraic system of degree at most 4 over $\mathbb{F}_2$

[8] A python implementation of this protocol can be found here: https://github.com/princess-elisabeth/sboxes_generation

3.4 Implementation and performances

Implementation. We implemented Elisabeth-4 over TFHE's library Concrete [9][9]. The symmetric key used by Elisabeth-4 is encrypted under an LWE key s_{in} of length n, and the PBSs are performed under an RLWE key of length k and a polynomial degree N. Since a side effect of the PBS is to switch the secret key under which a message is encrypted from s_{in} to s_{out}, a KeySwitch is needed after each PBS[10]. This KeySwitch can either happen right after the PBS, or after several operations. Since a KeySwitch is a rather noisy operation, the preferred approach was to perform it as close to the next PBS as possible. KeySwitches from s_{out} to s_{in} were thus computed after the sum of the output of 2 NLUTs and before an input ciphertext was added to this sum. The reason why this input ciphertext is summed after rather than before the KeySwitch is that it is encrypted under the secret key s_{in}. For the same reason, a KeySwitch is needed at the final step of the inner function, where we sum the fresh ciphertext of x_5 with the rest of the inner function. This KeySwitch can either be performed on x_5, from s_{in} to s_{out} with another KeySwitching key, or on every other part of the final sum, with the same KeySwitching key than earlier. The first approach allows faster computation, since this KeySwitch can be performed in parallel to the rest of the evaluation of the inner function, but increases the amount of data sent to the server before transciphering and has for side-effect that the output ciphertext and the symmetric key are not encrypted under the same LWE key. See Figure 3 for an illustration of the KeySwitches integration inside Elisabeth-4 's circuit.

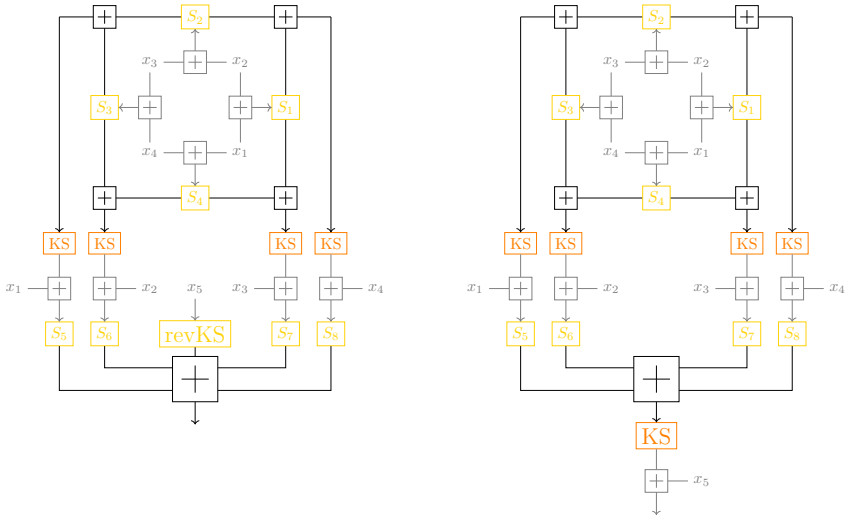
Parameters. Depending on the positioning of the KeySwitches, two sets of homomorphic parameters have been chosen for Elisabeth-4 , specified in Table 2. These parameters have been selected to ensure 128 bits of security and 4 bits of message integrity while optimizing evaluation time and key sizes. The integrity of the message has been controlled using the noise formulas given in [12] and the security is estimated based on the LWE security estimator [1].

TFHE Parameters												
Mode	n	k	N	$\log(\sigma_{\mathrm{LWE}})$	$\log(\sigma_{\mathrm{GLWE}})$	PBS		KeySwitch		inverse KeySwitch	
						$\log(B)$	ℓ	$\log(B)$	ℓ	$\log(B)$	ℓ
two KS	784	3	512	-18.6658	-38.4997	19	1	6	2	19	1
single KS	863			-20.7494				7		-	-

Table 2: Set of recommended FHE parameters for Elisabeth-4 .

[9] https://www.github.com/princess-elisabeth/Elisabeth

[10] Or, more precisely, between two PBS



(a) Two KeySwitch keys.

(b) Single KeySwitch key.

Fig. 3: Elisabeth-4 's homomorphic circuit. A gray edge represents operations under secret key s_{in} while black edges represent operations under s_{out}. Yellow boxes are operations that take inputs encrypted under s_{in} and output ciphertexts encrypted under s_{out} and orange boxes represent the opposite.

Benchmark. Since Elisabeth-4 has been designed for server usage, it has been tested on a 64-core computer equipped with AMD Ryzen Threadripper 3990X 64-Core Processor[11]. Results are in Table 3.

Timings		
Cipher	Time per ciphertext (ms)	Time per bit (ms)
Elisabeth-4 (two KS)	91.143	22.786
Elisabeth-4 (single KS)	103.810	25.953
Elisabeth-4 (two KS, monothreaded)	1485.0	371.25
Elisabeth-4 (single KS, monothreaded)	1648.6	412.15

Table 3: Comparison of running time of different versions of Elisabeth-4 .

As Elisabeth-4 has been designed for TFHE, it will mainly be compared to FiLIP , which has had its own specific implementation for TFHE [30]. In order to make our comparisons more relevant, FiLIP has been reimplemented over Concrete [12], with the same parameters as described in [30]. For completeness, we also provide a rough comparison of Elisabeth-4 with other state of the art encryption schemes for transciphering,

[11] Note that this particular design is initially optimized for 48 cores.

[12] https://github.com/princess-elisabeth/FiLIP

namely LowMC [2], Kreyvium [5], RASTA [19], FiLIP [34], DASTA [29], MASTA [27] and PASTA [20], all of which have been benchmarked with the tools provided by [20], and HERA [14], on the same machine used for benchmarking Elisabeth-4. Table 4 collects the results of these benchmarks.

Note that since these symmetric schemes have been designed with different FHE cryptosystems in mind, comparisons between their performances must be considered with care. For example, some homomorphic schemes are very efficient at packing, that is storing many cleartexts inside one ciphertext, but depending on the usecase following the transciphering, unpacking these cleartexts can be pretty costly. From a high-level perspective, these results confirm that ciphers that are more serial are also more interesting for latency, whereas the ones more parallel are better suited for throughput. But they are not indicative of any real-world performance in a specific usecase, which motivates the investigations of Section 5.

Cipher	Homomorphic library	Time per ciphertext (s)	Time per bit (ms)
LowMC	TFHE (C)	4283.678	16733
Kreyvium	TFHE (C)	208.255	208255
RASTA 6	TFHE (C)	2424.503	6907
FiLIP 144	Concrete	0.134	134
FiLIP 1216	Concrete	0.586	586
FiLIP 1280	Concrete	0.627	627
DASTA 6	TFHE (C)	2387.674	6802
Elisabeth-4 (two KS)	Concrete	0.091	22.75
Elisabeth-4 (single KS)	Concrete	0.104	26
LowMC	HELib	853.302	3333.21
Kreyvium	HELib	8.222	8222
RASTA 6	HELib	163.131	464.76
DASTA 6	HELib	156.935	447.11
MASTA 5	HELib	22.096	20.31
PASTA 4	HELib	9.827	18.06
HERA	CKKS	14.747	0.01

Table 4: Comparison of running time of different 128-bit security HHE, using either TFHE (TFHE (C) or Concrete), BFV (HELib) or CKKS (CKKS). The parameters used are the ones recommended by their original papers for 128 bits of security.

Note also that the seemingly less efficient performances of some cryptosystems used in combination with TFHE (C) are explained by a trade-off on the homomorphically encrypted key size: in order to make the ciphertexts used smaller, most of their operations are performed through gate bootstrapping, which significantly slow down the computation. FiLIP, which does not rely on this approach, runs much faster but needs way bigger keys. Table 5 gives a comparison of these sizes with Elisabeth-4, when transciphering towards TFHE (C). In the case of Elisabeth-4, we also included in this value the size of the KeySwitch and PBS keys needed to evaluate the transciphering algorithm. Data that would have been sent even without transciphering are not taken into account. Also note that rather than sending or storing a full GLWE or GGSW ciphertext, it is always possible to send only the body part of the ciphertext along with the seed used to generate its mask. When several ordered ciphertexts are sent, a single seed

can be provided for all of them. It is supposed in Table 5 that such compressions are used whenever possible.

Cipher	Key sizes (kB)
FiLIP 144	1610613
FiLIP 1216	402653
FiLIP 1280	1610613
Elisabeth-4 (two KS)	20
Elisabeth-4 (single KS)	8

Table 5: Comparison of public key sizes of FiLIP and Elisabeth-4 when used in combination with TFHE.

Tables 4 & 5 show that Elisabeth-4 gives a reduction of the data overhead compared to FiLIP by a factor 20000 to 200000, for a speedup of a factor 5 to 27.5 per bit. Moreover, Elisabeth-4 outputs are 4-bits integer, which FiLIP could only achieve through a costly series of PBS. Indeed, to reconstruct a multi-bit message, every bit but the most significant one transciphered by FiLIP must be shifted to the right as needed before being summed together. Each shift being performed through a PBS, a 4-bits message would thus cost an additional 3 PBS to be transciphered. The capacity to output multi-bit messages will prove useful in actual usecases.

4 Security analysis

4.1 Security of the group filter permutator

The group filter permutator generalizes the IFP paradigm over $\mathbb{F}_2$ [34] to any group. Accordingly, for the security analysis, we consider how the attacks known on IFP apply in the general case, and we discuss new attacks arising from the group. The security of IFP is discussed in [34], it is built on top of former analyses on filter permutator [35] (FP) and the connection with Goldreich's pseudorandom generator [25]. The main differences between the FP paradigm and the IFP paradigm are the addition of a subset selection before the permutation and a whitening. These modifications allow to mitigate the impact of guess and determine attacks on FP [21] and restricted inputs cryptanalysis [8]. The security analyses on IFP consider that no additional weakness arises from the PRNG which is chosen forward secure to avoid malleability. The subsets, permutations, and whitening are derived with no bias, hence the pseudorandom system obtained is not chosen in an adversarial way and the attacks applying on IFP are the ones targeting the filtering functions. These attacks are adaptations from the one applying on the filtered register, regrouped in [34] as algebraic-like attacks, correlation-like attacks, and guess and determine strategies.

The complexity of the different attacks on IFP is derived from parameters of the filtering function f. In the IFP case, f is a Boolean function and the significant cryptographic criteria of Boolean functions have been studied over the last decades for filtered registers. We argue that if the GFP is defined over a field, such as $\mathbb{F}_q$, the natural generalizations of such Boolean cryptographic criteria over $\mathbb{F}_q$ would give a strong basis for

the security analysis. Since GFP is defined over $\mathbb{G}$ we analyze the security based on the properties over $\mathbb{G}$ and over a field allowing a polynomial representation of f. At a high level since the "+" operation is linear in $\mathbb{G}$ the first view angle aims at verifying the filter breaks the linearity enough. The goal of the second perspective (over $\mathbb{G}'$) is to prevent attacks arising from a cryptographic weakness of the function f in another representation easy to obtain for an adversary. We consider the attacks in the single-key setting, in the known plaintext-ciphertext pairs model, focusing on key-recovery attacks. We make the standard assumption that an adversary is limited by a data complexity of $2^{\frac{\lambda}{2}} = 2^{64}$.

For Elisabeth-4 since $\mathbb{G} = \mathbb{Z}_{16}$ we study the properties of the filter f as a function from $\mathbb{G}^n$ to $\mathbb{G}$. Fixing the representation of each element of $\mathbb{G}$ as an element of $\mathbb{G}' = \mathbb{F}_2^4$ based on its binary decomposition fixes a vectorial Boolean function F from $\mathbb{F}_2^{4n}$ to $\mathbb{F}_2^4$. We study F by analyzing the properties of its 15 non-null component Boolean functions, the 15 functions from $\mathbb{F}_2^{4n}$ to $\mathbb{F}_2$.

Algebraic attacks. The general principle of algebraic attacks is to consider an algebraic system of equations derived for the key-stream: an attacker knowing plaintext-ciphertext pairs can rewrite key-stream outputs as multivariate polynomials in the secret key elements, and solve the system to recover the key. Various techniques can be used to solve such an algebraic system, as simple as linearization (considering each monomial as a new variable) followed by Gaussian elimination and as complex as approaches using Grobner bases (such as [22]). For the security estimation, we will use the approaches using linearization, as they allow to have a good estimation of the attack complexity from the algebraic degree of the system and the number of variables, whereas the complexity of more complex attacks is often difficult to assess or interpolate from toy examples. For a system of equations of degree d with N unknowns the attack has complexity $\mathcal{O}(D)^\omega$, where D is the number of monomials of degree lower or equal to d, and ω is the exponent from the Gaussian elimination (fixed to $\log(7)$ for our estimations).

Analyzing the security of Elisabeth-4 over $\mathbb{G} = \mathbb{Z}_{16}$, an adversary could target a system of equations in the polynomial ring $K[x_1, \ldots, x_N] = (\mathbb{G}, \cdot)[x_1, \ldots, x_N]$ where $\cdot$ denotes the usual product in $\mathbb{G}$. In this case where $\mathbb{G} = \mathbb{Z}_{16}$, K is a ring but not a field, hence not all the functions from K^n to K have a polynomial representation. When f does not correspond to a polynomial over $K[x_1, \ldots, x_N]$ we consider no algebraic attack applies on the representation over $\mathbb{G}$. Considering $K[x]$, since for all $x \in \mathbb{G}$ it holds $x^8 = x^4$, any polynomial in the univariate representation is equivalent to a polynomial of degree at most 7, hence there are at most 16^8 polynomial evaluations. In comparison, there are 16^{16} evaluations from $\mathbb{G}$ to $\mathbb{G}$. Accordingly, we assume that with high probability the function f used for the filter does not admit a polynomial representation over $K[x_1, \ldots, x_n]$, which avoids this attack, and experimentally we verify that the function does not correspond to a low degree polynomial.

Over $\mathbb{F}_2$ more specific attacks have been exhibited. The algebraic attack of Courtois-Meier [18] on filtered LFSR applies to the GFP: instead of considering a system of equations given by a filtering function f (in our case one of the non-null component functions of F) the attacker finds low-degree functions g and h such that $fg = h$

and derive a system of degree $\mathsf{AI}(f) = \max(\mathsf{deg}(g), \mathsf{deg}(h))$ which has the same solutions as the initial system. Unlike the algebraic degree that can be up to n for an n-variable Boolean function, it has been shown in [18] that the algebraic immunity (AI) is at most $\lceil (n+1)/2 \rceil$. Targeting this algebraic system of lower degree the corresponding algebraic-attack has a complexity of $\mathcal{O}(D)^\omega$ where $D = \sum_{i=1}^{\mathsf{AI}(f)} \binom{N}{i}$. The fast algebraic attack [17] is a variant of the previous attack where the adversary considers functions g and h such that $fg = h$ as before with g and h still of low degree but with h of degree potentially higher than $\mathsf{AI}(f)$. This attack can be more efficient than the previous one when the relations between the consecutive key-stream equations allow eliminating the monomials of degree between $\mathsf{deg}(g)$ and $\mathsf{deg}(h)$ in the system. In the case of filtered LFSR, the linear relation given by the LFSR allows to eliminate the monomials of degree larger than $\mathsf{deg}(g)$ using the Berlekamp-Massey algorithm, hence it gives a better attack than the standard algebraic one. The permutations used in the GFP do not provide such linear relations for consecutive key-stream equations, hence the elimination step would be more costly in this context. On a filtered LFSR the security estimation is $\mathcal{O}(D\log^2(D) + ED\log(D) + E^\omega)$ where $D = \sum_{i=1}^{\mathsf{deg}(h)} \binom{N}{i}$ and $E = \sum_{i=1}^{\mathsf{deg}(g)} \binom{N}{i}$. This estimation will be used as an indicator rather than a sharp limit, considering that the complexity of the best attack of the algebraic kind would lie between this (too low) bound and the (too high) one given by the algebraic attack of Courtois-Meier.

Correlation attacks. Two kinds of correlation attacks are considered on IFP, the first one uses the bias of the output (or a sub-part of the output) from uniform to mount a distinguishing attack, the second one consists in approximating the key-stream equations by low-degree equations and in retrieving the key by solving a noisy linear system such as a Learning Parity with Noise (LPN) instance. Two main criteria permit to estimate the impact of such attacks. The first one is the number of key elements to fix before having a non-uniform output, often referred to as resiliency order for Boolean functions. Generalizing the concept to any group $\mathbb{G}$, a resiliency order of k means that the output of a function from $\mathbb{G}^n$ to $\mathbb{G}$ remains uniformly distributed in $\mathbb{G}$ even fixing up to k variables, and there exists an affectation of $k+1$ variables such that the output is not uniform. The second criterion is the quality of the approximation (equivalently the importance of the bias), it measures the Hamming distance between f and a low-degree function. In particular, the distance to affine functions, the nonlinearity, often leads to the best attacks in this context. For l the best affine approximation of f, the nonlinearity $\mathsf{NL}_{\mathbb{G}}$ corresponds to $\mathsf{d_H}(f, l)$ and an adversary can consider the system of equation given by f as a noisy linear system given by l with noise parameter $\mathsf{NL}_{\mathbb{G}}/(|\mathbb{G}|^n)$. Since affine non-constant functions are well distributed (each element appears exactly $|\mathbb{G}|^{n-1}$ times), if the best affine approximation is not a constant function then the nonlinearity gives a bound on the bias from uniform. Consequently, both for the perspective from $\mathbb{G}$ and from $\mathbb{G}'$ we will ensure a non-trivial resiliency order to prevent these attacks, and we will consider the following rationals for the attack based on low-degree approximation. For an approximation of degree d over $\mathbb{G}$:

- D is the number of monomials of degree up to d,
- m is a positive integer lower than or equal to the number of equations available to the adversary,

– $\mathsf{NL}^d_{\mathbb{G}}$ is the Hamming distance between f and its best degree d approximation,
– $\mathsf{P}_{D,m}$ is the probability of having at least D equations over m with no error (Computed from a binomial law with error parameter $\mathsf{NL}^d_{\mathbb{G}}/(|\mathbb{G}|^n)$),

We will ensure that $\binom{m}{D}\mathsf{P}^{-1}_{D,m}D^{\omega} > 2^{\lambda}$, since this complexity bound matches the one of an attack consisting in solving a linearized system with enough noiseless equations.

In particular for Elisabeth-4 , over $\mathbb{Z}_{16}$ we will ensure a nontrivial resiliency and a sufficient distance to degree-1 and degree-2 functions. Over $\mathbb{F}_2$ we will ensure a nontrivial resiliency and a sufficient nonlinearity. We assume that considering higher degree approximations will lead to less efficient attacks since the gain in approximating the function is lost by the increasing number of monomials.

Guess and determine attacks. The cryptanalysis of [21] on an instance of FP showed the strength of guess and determine attacks on this design when too simple filters are used. The principle of the attack is to guess some elements of the key and attack the resulting system with one of the previously described attacks. Guess and determine attacks perform better than the former ones when the cost of guessing elements (try the attack for the $|\mathbb{G}|^{\ell}$ possibilities) is compensated by the simplicity of the remaining system. As for IFP, in the GFP the subset selection and whitening decrease the interest of guess and determine attacks since only a sub-part of the fixed elements of the key area in the selected subset for each equation, and the whitening randomizes which sub-function is obtained. In [35] the complexity of the guess and determine attacks is estimated based on the cryptographic parameter of the weakest function that can be obtained by fixing ℓ variables. In [34] an involved algorithm is used to compute the complexity of the attack targeting sub-functions with parameters lower than a threshold and leverage it by the probability of obtaining such sub-functions by fixing ℓ bits. Since this algorithm is practical only for bit-fixing stable families of functions (see [7]), we will use the simpler approach of [35]. For the different attacks listed above, the attack complexity is derived from the value of a filter's parameter, hence we will consider the attacks obtained with weaker values taking into account the minimal number of guesses needed to reach this value. Hence, the complexity estimation is $|\mathbb{G}|^{\ell}C(k)$ where $C(k)$ is the complexity for a parameter with value k and ℓ the minimal number of guesses to obtain a sub-function with this parameter value.

4.2 Security of Elisabeth-4

As described in Section 3.3 the design of Elisabeth relies on the use of direct sums, repeating the same basis function in a few variables. In this part we show how to derive the security estimations for Elisabeth-4 from the parameters of its basis function, analyzing it over $\mathbb{Z}_{16}$ and over $\mathbb{F}_2$.

Analysis over $\mathbb{Z}_{16}$. Over $\mathbb{Z}_{16}$ the filtering function can be written as:

$$f(x_1, \ldots, x_n) = \mathsf{DS}_{\mathbb{G}}(\mathsf{DS}^{12}_{\mathbb{G}}(h), \mathsf{DS}^{12}_{\mathbb{G}}(x_i)),$$

where h is the function from $\mathbb{G}^4$ to $\mathbb{G}$ given by the combinations of the 8 NLUTs given in Section 3.3 before adding x_5. We experimentally compute the resiliency, $\mathsf{NL}_{\mathbb{G}}$ and $\mathsf{NL}^2_{\mathbb{G}}$ of h and the sub-functions obtained from h by fixing 1 and 2 inputs, and list the results in Table 6. Then, the following proposition (Proposition 1) enables to bound the parameters of a direct sum over $\mathbb{G}$ and it is used to bound the parameters of the 60-variable filter f and its sub-functions obtained by guessing a limited number of inputs. Finally, we summarize the bounds on f parameters and the associated attack complexity in Table 7 following the analysis of Section 4.1. Additionally, we provide a primary analysis on the polynomials and functions over $\mathbb{G}$ which studies how selecting random NLUTs benefits the security over $\mathbb{G}$.

Proposition 1 ($\mathbb{G}$-direct sum properties). *Let $h = \mathsf{DS}(f, g)$ be the direct sum of f and g with n and m $\mathbb{G}$-variables respectively. Then $\mathsf{DS}_{\mathbb{G}}(f, g)$ has the following properties:*

1. *Resiliency:* $\mathsf{res}_{\mathbb{G}}(h) \geq \mathsf{res}_{\mathbb{G}}(f) + \mathsf{res}_{\mathbb{G}}(g) + 1$.
2. *Nonlinearity:* $\mathsf{NL}_{\mathbb{G}}(h) \geq \max(|\mathbb{G}|^n \mathsf{NL}_{\mathbb{G}}(g), |\mathbb{G}|^m \mathsf{NL}_{\mathbb{G}}(f))$.
3. *Order-2 nonlinearity:* $\mathsf{NL}^2_{\mathbb{G}}(h) \geq \max(|\mathbb{G}|^n \mathsf{NL}^2_{\mathbb{G}}(g), |\mathbb{G}|^m \mathsf{NL}^2_{\mathbb{G}}(f))$.

Proof. We begin with the result on $\mathsf{res}_{\mathbb{G}}(h)$. For all $t \in \mathbb{N} \cup -1$ such that $t \leq \mathsf{res}_{\mathbb{G}}(f) + \mathsf{res}_{\mathbb{G}}(g) + 1$, all choice of t variables corresponds to t_1 variables of f and t_2 of g such that $t = t_1 + t_2$. If $t_1 \leq \mathsf{res}_{\mathbb{G}}(f)$ the corresponding $(n - t_1)$-variable sub-function f' of f is balanced hence the direct sum $f' + g'$ is also balanced. If $t_1 > \mathsf{res}_{\mathbb{G}}(f)$ then $t_2 \leq \mathsf{res}_{\mathbb{G}}(g)$, therefore the corresponding $(m - t_2)$-variable sub-function g' of g is balanced hence the direct sum $f' + g'$ is also balanced. It allows us to conclude $\mathsf{res}_{\mathbb{G}}(h) \geq t$.

For the nonlinearity, an $n + m$ affine function can be written as:

$$\ell_{a,b} = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j y_j,$$

where the x_i (resp. y_j) denote the variables from the f part (resp. g). Then:

$$d_H(h, \ell_{a,b}) = \sum_{v \in \mathbb{G}^m} d_H(f + g(v), \ell_a + b \cdot v) \geq |\mathbb{G}|^m \mathsf{NL}_{\mathbb{G}}(f).$$

Since f and g play similar roles, the same arguments lead to $d_H(h, \ell_{a,b}) \geq |\mathbb{G}|^n \mathsf{NL}_{\mathbb{G}}(g)$, giving the final result.

Finally, the result for the order-2 nonlinearity can be directly adapted from the proof for the nonlinearity. When the m variables of g are fixed, on one side h equals f plus a constant, and on the other side any $n+m$ degree-at-most-2 function equals a degree-at-most-2 function depending only in the variables of f. Hence, the order-2 nonlinearity can be written as the sum of $|\mathbb{G}|^m$ distances (between f and a degree-at-most-2 function), each one greater than or equal to $\mathsf{NL}^2_{\mathbb{G}}(f)$. Noting that f and g play a similar role allows us to conclude.

The results from Table 7 show that the attacks considered on $\mathbb{G}$, correlation attacks using approximations of degree 1 or 2, are impracticable with the chosen filter.

Number of fixed variables ℓ	0	1	2
Resiliency $\mathsf{res}_{\mathbb{G}}$	-1	-1	-1
Nonlinearity $\mathsf{NL}_{\mathbb{G}}$	≥ 55296	≥ 3456	216
Order-2 nonlinearity $\mathsf{NL}^2_{\mathbb{G}}$	≥ 53248	≥ 3328	208

Table 6: Minimum $\mathbb{G}$-cryptographic parameters for h after fixing up to ℓ inputs. For 0 and 1 guess $\mathsf{NL}_{\mathbb{G}}(h)$ and $\mathsf{NL}^2_{\mathbb{G}}(h)$ are bounded using the worst value obtained with 2 guesses.

Number of guesses ℓ	0	$[1, 12]$	$[13, 23]$	$[24, 35]$
Resiliency	11	$11 - \ell$	-1	-1
$\mathsf{NL}_{\mathbb{G}}/\lvert\mathbb{G}\rvert^{n-\ell}$	≥ 0.84	≥ 0.84	≥ 0.84	0.84
Correlation attack complexity (bits)	$>> 128$	$>> 128$	$>> 128$	$>> 128$
$\mathsf{NL}^2_{\mathbb{G}}/\lvert\mathbb{G}\rvert^{n-\ell}$	≥ 0.81	≥ 0.81	≥ 0.81	0.81
Order-2 Correlation attack complexity (bits)	$>> 128$	$>> 128$	$>> 128$	$>> 128$

Table 7: Elisabeth-4 , minimal parameter bounds up to ℓ fixed inputs and complexity estimations for a key size $N = 256$ (1024 bits). The parameter bounds come from applying Proposition 1 with the values from Table 6 and the complexity estimations come from Section 4.1. We write"$>> 128$" when the complexity estimation overreaches several hundreds.

Polynomials and functions from $\mathbb{G}^4$ to $\mathbb{G}$.

As explained in Section 4.1, not all functions from $\mathbb{G}^N$ to $\mathbb{G}$ can be mapped to polynomials in the polynomial ring $K[x_1, \ldots, x_N] = (\mathbb{G}, \cdot)[x_1, \ldots, x_N]$. With the same arguments, not all functions from $\mathbb{G}^4$ to $\mathbb{G}$ admit a polynomial representation in $K[x_1, \ldots, x_4]$, and we show that in particular h does not correspond to a polynomial using a simple necessary condition to be a polynomial. Moreover, we show that most of the functions from this space are far in Hamming distance from the polynomials. More precisely, the probability of a random function to agree with a polynomial on at least half of the inputs is lower than 2^{-128}.

Proposition 2. *Let f a function from $\mathbb{G}^4$ to $\mathbb{G}$. If $f(8, 8, 8, 8) - f(0, 0, 0, 0) \notin \{0, 8\}$ then f is not a polynomial of $(\mathbb{G}, \cdot)[x_1, \ldots, x_4]$.*

Proof. All monomial of $(\mathbb{G}, \cdot)[x_1, \ldots, x_4]$ have the form $x_1^{i_1}\, x_2^{i_2}\, x_3^{i_3}\, x_4^{i_4}$ where i_1 to i_4 are positive integers. In $(0, 0, 0, 0)$ all non-trivial monomial gives 0 and in $(8, 8, 8, 8)$ if one exponent is greater than 1 or more than one exponent is not null a monomial gives 0, the remaining non trivial monomials give 8. Let p be a polynomial of $(\mathbb{G}, \cdot)[x_1, \ldots, x_4]$:

$$p(x_1, x_2, x_3, x_4) = \sum_{i_1, i_2, i_3, i_4 \in \mathbb{N}} a_{i_1, i_2, i_3, i_4} x_1^{i_1} x_2^{i_2} x_3^{i_3} x_4^{i_4},$$

where $a_{i_1, i_2, i_3, i_4} \in \mathbb{G}$. Then, $p(0, 0, 0, 0) = a_{0,0,0,0}$ and $p(8, 8, 8, 8) = a_{0,0,0,0} + 8(a_{1,0,0,0} + a_{0,1,0,0} + a_{0,0,1,0} + a_{0,0,0,1})$ hence $p(8, 8, 8, 8) - p(0, 0, 0, 0) \in \{0, 8\}$, allowing to conclude.

Proposition 3. *Let $\mathcal{G}$ be the space of functions from $\mathbb{G}^4$ to $\mathbb{G}$, the probability that a function f taken uniformly at random in $\mathcal{G}$ agrees with a polynomial of $(\mathbb{G}, \cdot)[x_1, x_2, x_3, x_4]$ on at least half of the inputs is lower than 2^{-128}.*

Proof. For this proof we use the formalization of error correcting codes. Each function in $\mathcal{G}$ can be uniquely represented by its vector of outputs in $\mathbb{G}^{16^4}$, and the polynomials of $(\mathbb{G}, \cdot)[x_1, x_2, x_3, x_4]$ form a linear code of the vector-space $(\mathbb{G}, \cdot)^{16^4}$ with parameter [n, k, d]. First, we determine an upper bound on the number of polynomials with different evaluations (which corresponds to an upper bound on the code dimension k). Then we determine an upper bound on the number of elements of $\mathbb{G}^{16^4}$ at Hamming distance lower than $n/2 + 1$. Finally, we compare the last bound with the number of functions in $\mathcal{G}$ to conclude.

For $x \in \mathbb{G}$ the value of x^4 is 0 if x is even and 1 if x is odd, since x^8 has the same property, all polynomial of $(\mathbb{G}, \cdot)[x]$ has a representative of degree at most 7. Thereafter, for the polynomial ring $(\mathbb{G}, \cdot)[x_1, x_2, x_3, x_4]$ all monomial $x_1^{i_1} x_2^{i_2} x_3^{i_3} x_4^{i_4}$ with exponents $i_1, i_2, i_3, i_4 \in \mathbb{N}$ has a representative with $i_1, i_2, i_3, i_4 \in [0, 7]$. Since it gives at most $8^4 = 2^{12}$ monomials (that is $k \leq 2^{12}$), there are at most $16^{2^{12}} = 2^{2^{14}}$ different polynomial representations.

Then, we use the standard packing/covering bound to determine the maximum number of elements covered by the union of Hamming balls of radius $n/2$ centered around the code elements (here the polynomials). Each ball contains $B = \sum_{i=0}^{n/2} (|\mathbb{G}| - 1)^i \binom{n}{i}$ elements, giving a total number of $2^{2^{14}} B$ elements: the upper bound reached if no ball intersect.

Finally, since there are $(16)^{16^4} = 2^{2^{18}}$ elements in $\mathcal{G}$ we can compute the proportion p of functions coinciding on a least half of there inputs with a polynomial:

$$p \leq \frac{2^{2^{14}} B}{2^{2^{18}}} = \frac{\sum_{i=0}^{2^{15}} 15^i \binom{2^{16}}{i}}{2^{15 \cdot 2^{14}}} \approx 2^{-2^{15.67}} < 2^{-128}.$$

Proposition 3 shows that most of the functions from $\mathbb{G}^4$ to $\mathbb{G}$ cannot be well approximated by a polynomial. The same idea extends to functions from $\mathbb{G}^n$ to $\mathbb{G}$, therefore it makes the attack consisting in solving a noisy polynomial system over $\mathbb{G}$ very unrealistic when the filter is a random function. Since h is not taken at random from $\mathcal{G}$, and neither is f, we cannot directly use this argument. This is why we additionally provide bounds for the nonlinearity and distance from degree-2 polynomials.

Analysis over $\mathbb{F}_2$ To perform the analysis over $\mathbb{F}_2$ we introduce Beth-4 , a variation of Elisabeth-4 . The variation consists in replacing some additions over $\mathbb{Z}_{16}$ by coordinate-wise XOR, it gives a scheme which security is easier to analyze and conjectured at most as secure as Elisabeth-4 .

Formally, Beth-4 is a stream cipher following the GFP paradigm with $\mathbb{G}' = \mathbb{F}_2^4$ where the operation is the addition modulo 2 coordinate-wise. The addition of the whitening and the addition of the key-stream with the plaintext are considered in $\mathbb{G}'$, only the additions inside the 5-to-1 function remain in $\mathbb{G}$. More precisely, the filtering function F from $(\mathbb{F}_2^4)^n$ to $\mathbb{F}_2^4$ is computed as $\mathsf{DS}_{\mathbb{G}'}(H(x_{20i-19}, \cdots, x_{20i}), i \in [4n/5])$ where H is determined by the truth table obtained by considering the canonical embedding of $\mathbb{G}$ over $\mathbb{G}'$ on the inputs and outputs of the ($\mathbb{G}^5$ to $\mathbb{G}$) function g defined in Section 3.3.

Since the design of Beth-4 uses direct sums and 20-variable Boolean functions its security can be analyzed easily following the approach on IFP [34]. Seen over $\mathbb{G}' = \mathbb{F}_2^4$, the addition in $\mathbb{G}$ corresponds to the addition (in $\mathbb{F}_2^4$) more nonlinear combinations of the coordinate functions due to caries. Therefore, the algebraic system given by Elisabeth-4 is more complex over $\mathbb{F}_2$ than the one of Beth-4 , and we assume that Elisabeth-4 is at least as secure as Beth-4 for the different attacks we consider. Accordingly, in the following part, we determine the parameters of the 20-variable component functions of the basis function of Beth-4 . Then, we bound the significant cryptographic parameters of the $4n$-variable component Boolean functions. Finally, we give the complexity estimations for the security of Beth-4 , we consider these values as lower bounds for the complexity estimations on the security of Elisabeth-4 .

Over $\mathbb{F}_2^4$ the filtering function can be written as $F(x_1, \ldots, x_{4n}) = \mathsf{DS}_{\mathbb{G}'}^{12}(H)$, where H is the function from $\mathbb{F}_2^{20}$ to $\mathbb{F}_2^4$ given by the NLUTs and addition in $\mathbb{G}$ illustrated in Section 3.3. We experimentally compute the parameters of H and give them in Table 8. Then, the following lemmas allow to determine the cryptographic parameters of the function F, with or without guessed bits. The structure of direct sum of F component functions is preserved even guessing bits, hence Lemma 1 on the cryptographic properties of direct sums enables to bound the parameters (degree, nonlinearity, resiliency) of the obtained functions from the parameters of H. Lemma 2 gives a lower bound on the AI of a direct sum based on the degree of its sub-parts, it allows us to bound the AI of the obtained functions from the degree of the components of H (with or without guessing bits). Finally, the bounds on F's parameters are given in Table 9, together with the complexity estimations following the analysis of Section 4.1.

ℓ	0	1	2	3
deg	12	10	9	8
res	0	-1	-1	-1
NL	509440	249216	120768	57920

Table 8: Minimum Boolean cryptographic parameters over H's components after fixing up to ℓ binary inputs.

Lemma 1 (Boolean direct sum properties (e.g. [35] Lemma 3)). *Let $h = \mathsf{DS}(f, g)$ be the direct sum of f and g n and m-variable Boolean functions respectively. Then $\mathsf{DS}(f, g)$ has the following cryptographic properties:*

1. *Degree:* $\mathsf{deg}(h) = \max(\mathsf{deg}(f), \mathsf{deg}(g))$.
2. *Algebraic immunity:* $\max(\mathsf{AI}(f), \mathsf{AI}(g)) \leq \mathsf{AI}(h) \leq \mathsf{AI}(f) + \mathsf{AI}(g)$.
3. *Resiliency:* $\mathsf{res}(h) = \mathsf{res}(f) + \mathsf{res}(g) + 1$.
4. *Nonlinearity:* $\mathsf{NL}(h) = 2^m\mathsf{NL}(f) + 2^n\mathsf{NL}(g) - 2\mathsf{NL}(f)\mathsf{NL}(g)$.

Lemma 2 ([33] Lemma 6). *Let $t \in \mathbb{N}^*$, and $f_1, \ldots, f_t$ be t Boolean functions, if for $r \in [t]$ there exists r different indexes $i_1, \cdots, i_r$ of $[t]$ such that $\forall j \in [r], \mathsf{deg}(f_{i_j}) \geq j$ then $\mathsf{AI}(\mathsf{DS}(f_1, \ldots, f_t)) \geq r$.*

Number of guesses ℓ	0	$[1, 10]$	11	$[12, 23]$	$[24, 35]$
Resiliency	11	$11 - \ell$	0	-1	-1
$\mathsf{NL}/2^{4n-\ell}$	0.49	≥ 0.48	≥ 0.48	≥ 0.47	≥ 0.46
Correlation attack complexity (bits)	$>> 128$	$>> 128$	$>> 128$	$>> 128$	$>> 128$
Algebraic immunity	12	12	11	10	9
AA complexity (bits) $4N = 1024$	229	230	222	205	198
AA complexity (bits) $4N = 4096$	288	289	276	254	242
FAA complexity (bits) $4N = 1024$	110	111	114	108	113
FAA complexity (bits) $4N = 4096$	136	137	139	131	134

Table 9: Beth-4 , minimal parameter bounds up to ℓ fixed binary inputs and complexity estimations (in bits) for key sizes $N = 256$ (1024 bits) and $N = 1024$ (4096 bits). The complexity estimations come from Section 4.1, when ℓ lies in an interval the complexity given is the minimal one for this interval of guesses. AA refers to the algebraic attack and FAA to the fast algebraic attack. The complexity of the fast algebraic attack is computed taking $\mathsf{deg}(g) = 1$ and $\mathsf{deg}(h) = \mathsf{AI}(f) + 1$.

The results on the security of Beth-4 in Table 9 show that the correlation attacks are impracticable with the chosen filter, which goes in the same direction as the analysis over $\mathbb{G}$. Regarding the attacks on the algebraic kind, the estimations for the fast algebraic attacks (FAA) are slightly below the threshold of 128 bits of security when we consider a key size corresponding to 1024 bits. The last row of the table shows that taking a bigger key (equivalent to 4096 bits) is sufficient to reach the threshold of 128 bits for this attack, and for the cipher instance since the FAA gives by far the lower attack complexity. Since as explained in Section 4.1 the complexity of FAA is underestimated on GFP, and the level of security of Elisabeth-4 is assumed higher than the one of Beth-4 , we consider that the key size of 1024 bits is an adequate choice for a security level of 128 bits, and an interesting starting point for further cryptanalyses.

5 Case study

Homomorphic encryption is useful to prevent privacy issues and transciphering will typically be used on personal data in a real-world scenario. Nowadays, algorithms handling personal data, and already running server-side, are mainly neural networks. Therefore, we next present a neural network evaluation with HHE. The homomorphic evaluation of neural networks (i.e., the inference part) using TFHE has already been studied in [11]. To the best of our knowledge, our following investigations are the first to evaluate the combination of such a complex FHE computation with a transciphering, which will allow us to discuss the interest of HHE in more general terms.

We decided to evaluate a neural network over the standard dataset Fashion MNIST (FMNIST)[13]. The neural network performs classification on 28×28 pixels grayscale clothes pictures. This dataset has been created to provide a more challenging problem

[13] https://github.com/zalandoresearch/fashion-mnist

than MNIST, which is a handwritten text recognition dataset. We note that this dataset itself does not pose privacy concerns and an even more relevant usecase would be to classify medical data streams (e.g. monitored by a smartwatch). Yet, it is a standard and well-documented one and serves as an interesting first step for benchmarking. The challenges raised by more complex data sets are discussed in the conclusions.

5.1 Network design under FHE constraints

To make the homomorphic evaluation of the network as fast as possible as well as taking into account the fact that Elisabeth-4 outputs 4-bit messages, the model has to be designed with some constraints in mind.

TFHE for Machine Learning. As presented in section 2.1, TFHE is efficient for performing modular additions, multiplications with public scalars, and evaluation of homomorphic NLUTs. While most functions are not negacyclic, it is actually possible to evaluate them through a PBS thanks to a small trick: any arbitrary LUT of length N can be transformed into a negacyclic one of length $2N$ by appending its negation to it. When evaluating this lookup table, the user is only interested in the value written in the first half of the table, that is the slots 0 to $N-1 = 2^{d-1}-1$. These are the slots for which the most significant bit is set to 0, so to ensure that the value retrieved from the table is in the first half, one must guarantee that the MSB of the message is 0. This is done by adding a bit of padding in the MSB of the plaintext. Since by design Elisabeth-4 does not require this bit of padding, it has been decided to encrypt it along with the message, leaving 3 effective bits for the actual message. Note that a recent work from Chillotti *et al.* [12] suggests that bits of padding inside a PBS can be avoided and could become obsolete in the future. The ability to compute arbitrary functions, especially non-linear ones, through a PBS gives the possibility to efficiently evaluate the activation function of a neural network. Every other operation in between two activation functions is linear and can be efficiently computed with TFHE. The reader can also refer to [11] for more details on the homomorphic evaluation of neural networks.

Neural Network. In order to classify FMNIST pictures, the Convolutional Neural Network (CNN) [26] relies on a convolutional layer to extract features, then on linear layers in order to perform the classification. Since the homomorphic inference is performed on 3-bit data (instead of originally 8 bits), the CNN has been trained on unencrypted quantized data using Weight Decay [26] to prevent overfitting. The reader will find below a picture of a grayscale picture from the FMNIST dataset, which is degraded from 8 bits to 3 bits of quantization in the right part.

We use a sigmoid function on our data at the entrance of the neural network. This (non-standard) operation is performed for 2 reasons:

- The sigmoid bounds the norm of the data we manipulate. Having smaller-norm data reduce the noise growth during the homomorphic operation.
- We are impelled to perform a PBS between Elisabeth-4 and the CNN, for the reasons developed in the previous subsection 5.1 (mainly in order to add a bit of padding). While doing it, we can evaluate the sigmoid without any additional cost.

(a) Original data (8 bits shades) (b) Quantized data (3 bits shades)

Fig. 4: 784-pixels Fashion MNIST pictures

In order to perform homomorphic inference, the CNN expects a bootstrapped LWE ciphertext encrypted under the input key s_{in}. This implies that Elisabeth is used with the single KeySwitch parameters in order to save a KeySwitch at the exit of Elisabeth-4 's evaluation. The scalar weights inside the linear layers are given in clear so that these layers are evaluated efficiently with TFHE. Still, for noise management reasons, these weights ought to remain small (inside a magnitude of 20 for the FMNIST implementation). The CNN does not output normalized probability nor use a softmax function, as those operations would be costly when performed homomorphically.

As detailed in the next subsection, not using these tools does not have a big impact on the resulting accuracy, which remains high when we evaluate the adapted-for-homomorphic CNN with unencrypted data.

5.2 Performance

In order to reduce the amount of data sent to the server, we chose to use a single PBS key and a single $\mathrm{KeySwitch}$ key for the whole circuit, rather than different keys for the transciphering and the homomorphic inference. The parameters chosen with these constraints are given in table 10. In clear, the trained network gave 84.37% accuracy over 10,000 randomized inputs. When tested over the same inputs, encrypted this time, the network kept a pretty similar accuracy of 84.18%. The loss of accuracy in an homomorphized network is studied in [11].

One homomorphic inference took 427.23 seconds on average over 100 samples, versus 5.74 seconds per homomorphic inference without transciphering. This means that the transciphering took 537.62 ms per 4-bit message (with $28^2 = 784$ messages). This increase in transciphering time is explained by the constraints put by the neural network on the PBS and $\mathrm{KeySwitch}$ keys it shares with Elisabeth-4 . This is expected since Elisabeth-4 was designed so that its parameters would be less restrictive than most usecases, in order to make sure that transciphering would neither require its own set of public keys nor slow down the rest of the computation. Note that we chose here to minimize bandwidth consumption, but computation time could be further improved by sending Elisabeth's optimal PBS and $\mathrm{KeySwitch}$ keys along with the one used by the neural network. That way, Elisabeth could run in optimal time, reducing the transciphering time to $28 \times 28 \times 0.091143 = 71.46$ seconds. The initial data overload

on the bandwidth would thus be higher and longer to compensate, but the inference time would be greatly reduced. More details on this are given in Figure 6. Also note that the transciphering algorithm[14] can be performed offline even before the client starts sending their inputs, thus saving time during the online phase.

Evaluating 1000 inferences of this model with Elisabeth-4 requires sending 400 MB over the network, against $5,337$ GB if LWE ciphertexts were sent directly without compression, and $6,272$ MB if only LWE bodies and their corresponding seed were sent. This represents a saving of at least 93.62% of bandwidth. Figure 5 compares the bandwidth saved by Elisabeth-4 and FiLIP over seeded LWE ciphertexts. It both confirms the interest of the HHE framework in general, and the significant gains that Elisabeth-4 provides over FiLIP for our usecase.

TFHE Parameters								
n	k	N	$\log(\sigma_{\text{LWE}})$	$\log(\sigma_{\text{GLWE}})$	PBS		KeySwitch	
					$\log(B)$	ℓ	$\log(B)$	ℓ
754	1	2048	-17.8745	-52.0036	7	6	2	8

Table 10: Set of FHE parameters used for Elisabeth-4 combined with a neural network.

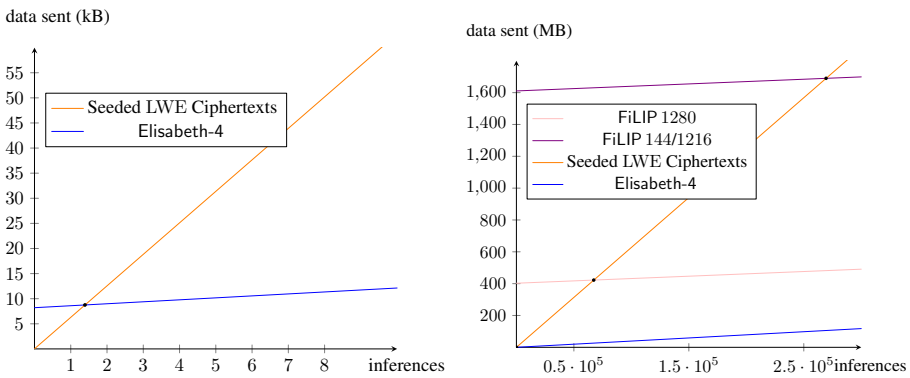


Fig. 5: Data sent per inference with different methods of compression. Elisabeth-4 has a better usage of bandwidth if more than two inferences are performed. On the other side, FiLIP needs 67366 or 269424 inferences (depending on the version) to become cheaper in bandwidth usage than seeded ciphertexts.

Our implementation is available at:

https://github.com/princess-elisabeth/elisabeth_usecase.

[14] Up to the point of summing with the ciphertext.

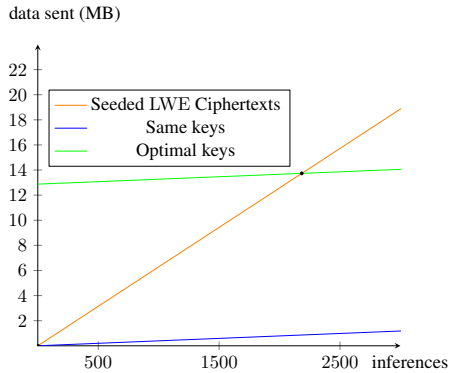


Fig. 6: Using Elisabeth-4 with its optimal parameters implies an overhead on bandwidth usage since another PBS and KeySwitch have to be sent to the server. That being the case, this overhead is compensated compared to seed-compression in only 2185 inferences. Note also that these inferences would take roughly 47 hours to compute, whereas less than 400 inferences could be computed in the same amount of time using the same keys for both Elisabeth-4 and the neural network.

6 Conclusion

Our results show that HHE can highly benefit from the optimization of a symmetric encryption scheme depending on the operations to be performed homomorphically. We introduced the family of stream ciphers Elisabeth for this purpose and instantiated it with 4-bit components, motivated by current constraints of the TFHE Concrete library. While the stand-alone homomorphic evaluation of Elisabeth-4 only brings similar performances to state-of-the-art competitors, its combination with the homomorphic classification of grayscale images from the Fashion MNIST dataset leads to significant improvements. In particular, the transciphering does not impact the accuracy of the inference (compared to the direct homomorphic processing of fresh ciphertexts) and it enables large gains in terms of bandwidth.

These results lead to several challenging open problems. First, they are based on an admittedly simple usecase, where the quantization of the images in 3 bits does not significantly affect its accuracy. It is expected that more challenging case studies (e.g. medical data, where fine-grain details can be essential for the classification) may require either more granular data (which would be calling for larger NLUTs) or more complex models (which may change the cost balance between the transciphering and the homomorphic inference). Considering larger instances of Elisabeth is an interesting direction regarding the first concern. Optimizing machine learning for homomorphic computations and leveraging transformations that would reduce the load of homomorphic computations to perform are interesting directions regarding the second concern. For example, the application of a Principal Component Analysis (PCA) [39], could be used to reduce the number of dimensions of the Fashion MNIST images from 28×28 down to 10, however with a penalty on the accuracy of approximately 10%. PCA there-

fore enables to choose a compromise between the amount of data sent and the accuracy. Other types of features selection would certainly deserve further investigation.

Another possible improvement client-side would be the encryption time. Even though Elisabeth-4 encryption algorithm is reasonably fast, the generation of its permutations and whitening values could be optimized and would represent an interesting development.

Finally, studying how to apply other FHE-friendly symmetric encryption and FHE schemes to practical-relevant case studies will be important, in order to better assess which combination works best in which context. Besides, and independent of the interest of the HHE framework, improving the loss of accuracy that homomorphic implementations may imply, as observed with TFHE and the Concrete library for example, is a critical optimization goal to further incentivize the application of FHE in contexts where privacy requires it. We believe having a first (and more) benchmark(s) is an essential seed to drive research towards achieving these important goals.

Acknowledgments. François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). Pierrick Méaux was supported by the ERC Advanced grant CLOUDMAP (num. 787390). This work has been funded in part by the European Union through the ERC consolidator grant SWORD (num. 724725), and by the PEPR Cyber France 2030 programme (ANR-22-PECY-0003). We thank Arthur Meyre for his help with the neural network, Samuel Tap for the parameters and Pascal Paillier, Damien Stehlé and Alain Passelègue for interesting discussions.

References

1. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
2. M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, Apr. 2015.
3. M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. *IACR Cryptology ePrint Archive*, 2016:687, 2016.
4. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, Jan. 2012.
5. A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In T. Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*. Springer, Heidelberg, Mar. 2016.
6. C. Carlet. *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press, 2021.
7. C. Carlet and P. Méaux. A complete study of two classes of boolean functions: Direct sums of monomials and threshold functions. *IEEE Trans. Inf. Theory*, 68(5):3404–3425, 2022.
8. C. Carlet, P. Méaux, and Y. Rotella. Boolean functions with restricted input and their robustness; application to the FLIP cipher. *IACR Trans. Symmetric Cryptol.*, 2017(3), 2017.
9. I. Chilloti, M. Joye, D. Ligier, J.-B. Orfila, and S. Tap. Concrete: Concrete operates on ciphertexts rapidly by extending tfhe. *8th Workshop on Encrypted Computing and Applied Homomorphic Cryptography (WAHC 2020)*, 2020.

10. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: fast fully homomorphic encryption over the torus. *J. Cryptol.*, 33(1):34–91, 2020.
11. I. Chillotti, M. Joye, and P. Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In *Cyber Security Cryptography and Machine Learning (CSCML 2021)*, volume 12716 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2021.
12. I. Chillotti, D. Ligier, J.-B. Orfila, and S. Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. In *ASIACRYPT*, 2021.
13. J. Cho, J. Ha, S. Kim, B. Lee, J. Lee, J. Lee, D. Moon, and H. Yoon. Transciphering framework for approximate homomorphic encryption (full version). *IACR Cryptol. ePrint Arch.*, page 1335, 2020.
14. J. Cho, J. Ha, S. Kim, B. Lee, J. Lee, J. Lee, D. Moon, and H. Yoon. Transciphering framework for approximate homomorphic encryption. In *ASIACRYPT*. Springer-Verlag, 2021.
15. B. Cogliati and T. Tanguy. Multi-user security bound for filter permutators in the random oracle model. *Designs, Codes and Cryptography*, 09 2018.
16. J.-S. Coron, T. Lepoint, and M. Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 311–328. Springer, Heidelberg, Mar. 2014.
17. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In D. Boneh, editor, *CRYPTO 2003*, pages 176–194, 2003.
18. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *EUROCRYPT 2003*, 2003.
19. C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger. Rasta: A cipher with low anddepth and few ands per bit. In *CRYPTO 2018*, pages 662–692, 2018.
20. C. Dobraunig, L. Grassi, L. Helminger, C. Rechberger, M. Schofnegger, and R. Walch. Pasta: A case for hybrid homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 731, 2021.
21. S. Duval, V. Lallemand, and Y. Rotella. Cryptanalysis of the FLIP family of stream ciphers. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, pages 457–475, 2016.
22. J.-C. Faugère. A new efficient algorithm for computing groebner bases. *Journal of Pure and Applied Algebra*, 139:61–88, june 1999.
23. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, Aug. 2012.
24. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, 2013.
25. O. Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.
26. I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.
27. J. Ha, S. Kim, W. Choi, J. Lee, D. Moon, H. Yoon, and J. Cho. Masta: An he-friendly cipher using modular arithmetic. *IEEE Access*, 8:194741–194751, 2020.
28. J. Ha, S. Kim, B. Lee, J. Lee, and M. Son. Rubato: Noisy ciphers for approximate homomorphic encryption. In *EUROCRYPT*. Springer-Verlag, 2022.
29. P. Hebborn and G. Leander. Dasta - alternative linear layer for rasta. *IACR Trans. Symmetric Cryptol.*, 2020(3):46–86, 2020.
30. C. Hoffmann, P. Méaux, and T. Ricosset. Transciphering, using filip and TFHE for an efficient delegation of computation. In *INDOCRYPT*, volume 12578 of *Lecture Notes in Computer Science*, pages 39–61. Springer, 2020.

31. A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.
32. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May 2010.
33. P. Méaux. On the algebraic immunity of direct sum constructions. *Discret. Appl. Math.*, 320:223–234, 2022.
34. P. Méaux, C. Carlet, A. Journault, and F. Standaert. Improved filter permutators for efficient FHE: better instances and implementations. In F. Hao, S. Ruj, and S. S. Gupta, editors, *Progress in Cryptology - INDOCRYPT*, 2019.
35. P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 311–343. Springer, Heidelberg, May 2016.
36. M. Naehrig, K. E. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *CCSW*, pages 113–124. ACM, 2011.
37. N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman. Sok: Security and privacy in machine learning. In *EuroS&P*, pages 399–414. IEEE, 2018.
38. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.
39. S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.
40. H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

Supplementary material

A Details on the Programmable Bootstrapping

In this part, we detail how works the PBS to apply an NLUT and bootstrap a LWE ciphertext at the same time. First, we recall the definition of a cryptographic multiplexer, and then we explain how the computation of a length-$2N$ NLUT is incorporated inside the bootstrapping.

Definition 11 **(**CMux**).** *A Cryptographic Multiplexer (*CMux *in short) is an operator that, given two* GLWE *ciphertexts* $\boldsymbol{c}_0$ *and* $\boldsymbol{c}_1$ *of* μ_0 *and* μ_1 *respectively and a* GGSW *encryption* $\mathbf{B}$ *of a bit* b*, outputs a reencryption of* μ_b*. This can be done in a single external product by computing* $\mathbf{B} \boxdot (\boldsymbol{c}_1 - \boldsymbol{c}_0) + \boldsymbol{c}_0$.

Given $\mathscr{L}(X) = \sum_{i=0}^{N} l_i X^i \in \mathbb{Z}_{q,N}[X]$, one can see that the constant term of $X^{-(i+N)}\mathscr{L}(X) = -X^{-i}\mathscr{L}(X)$ is $-l_i$, meaning that a negacyclic look-up table of length $2N$ can be represented as a polynomial of $\mathbb{Z}_{q,N}[X]$: accessing the i-th value of the look-up table is equivalent to multiplying the polynomial by X^{-i} then keeping the constant term.

Let $\boldsymbol{c}_L$ be a GLWE encryption of such a polynomial[15] and $\boldsymbol{c}$ be a LWE encryption of a message μ. The goal of bootstrapping is to secretly select a value of L based on the value of μ. Since there are $2N$ slots in L and since μ can take q different values, one first needs to rescale μ from $[0, q]$ to $[0, 2N]$, an operation known as Modulus Switching. This is simply done by multiplying each coefficient of $\boldsymbol{c}$ by $2N/q$, then by rounding to the closest integer. Let call $(\bar{a}_1, \ldots, \bar{a}_n, \bar{b})$ the scaled coefficients obtained, and $\bar{\mu}$ the value of μ rescaled. One now has to select the $\bar{\mu}$-th slot of the NLUT. This can be done approximately by multiplying the polynomial $\mathscr{L}$ by $X^{-\bar{b}+\sum \bar{a}_i s_i} = X^{-\bar{\mu}-\bar{\varepsilon}}$, where the $\bar{a}_i$ and $\bar{b}$ are publicly known. Thus, computing $\mathsf{ACC} \leftarrow \boldsymbol{c}_L \cdot X^{-\bar{b}}$ can be done immediatly. Multiplying by $X^{\sum \bar{a}_i s_i}$ is done iteratively thanks to a series of CMuxes: by using GGSW encryption $\mathbf{S}_i$ of the bits of the LWE secret key s_1 to s_n, one computes $\mathsf{ACC} \leftarrow \mathrm{CMux}(\mathbf{S}_i, X^{\bar{a}_i}\,\mathsf{ACC}, \mathsf{ACC}) = \mathsf{ACC} \cdot X^{\bar{a}_i s_i}$. This yields an encryption of a polynomial which constant coefficient is $L[\bar{\mu}^*]$. Since it is not possible to directly compute the rounding of $\bar{\mu}^* = \bar{\mu} + \bar{\varepsilon}$ homomorphically to recover $L[\bar{\mu}]$, the only alternative is to introduce redundancy in L, so that $L[\bar{\mu}^*] = L[\bar{\mu}]$ for small enough values of $\bar{\varepsilon}$. The actual number of values that the lookup table can hold thus depends on both the degree of the polynomial and the maximal size $\bar{\varepsilon}$ can take: the bigger the polynomial the more redundancy can be introduced, the smaller ε and the lesser redundancy is needed. Now, given a GLWE encryption $(\boldsymbol{a}, b)$ of $\sum \mu_i X^i$ under the secret key $\mathfrak{s} = (\sum s_{1,i} X^i, \ldots, \sum s_{k,i} X^i)$ with $\boldsymbol{a} = (\sum a_{1,i} X^i, \ldots, \sum a_{k,i} X^i)$ and $b = \sum b_i X^i$, one can build an LWE encryption of μ_0 under the secret key $(s_{1,0}, \ldots, s_{1,N-1}, \ldots, s_{k,0}, \ldots, s_{k,N-1})$ as:

$$(a_{1,0}, -a_{1,N-1}, \ldots, -a_{1,1}, \ldots, a_{k,0}, -a_{k,N-1}, \ldots, -a_{k,1}, b_0)$$

This operation is called sample extraction and does not increase the noise in the ciphertext. A complete bootstrap cycle then consists of these three operations: modulus

[15] Note that it is always possible to build a trivial encryption of $\mathscr{L}$ by appending it to a vector of zeros.

switching, blind rotation of the negacyclic look-up table, and sample extraction. The output noise of the bootstrapped ciphertext is independent of the input ciphertext and depends only on the number of CMuxes that has been performed, which in turn depends on the length of the LWE key. The noise caused by each CMux depends on the degree N of the polynomials as well as on the basis and number of levels of the GGSW used.

B Elisabeth-4 specifications

This appendix describes the details of Elisabeth-4 's implementation.

B.1 NLUTs table

We specify the Negacyclic Look-Up Tables used for Elisabeth-4 implementation. Remember that the second half of each NLUT's value is entirely determined by the first.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	3	2	6	12	10	0	1	11	13	14	10	4	6	0	15	5
S_2	4	11	4	4	4	15	9	12	12	5	12	12	12	1	7	4
S_3	11	10	12	2	2	11	13	14	5	6	4	14	14	5	3	2
S_4	5	9	13	2	11	10	12	5	11	7	3	14	5	6	4	11
S_5	3	0	11	8	13	14	13	11	13	0	5	8	3	2	3	5
S_6	8	13	12	12	3	15	12	7	8	3	4	4	13	1	4	9
S_7	4	2	9	13	10	12	10	7	12	14	7	3	6	4	6	9
S_8	10	2	5	5	3	13	15	1	6	14	11	11	13	3	1	15

B.2 Elisabeth-4 algorithms

In this specification, notations from the article are used. For the reader's comfort, let us remind here a few of them:

- $\mathbb{G} = \mathbb{Z}_{16}$.
- S_i denotes the NLUTs.
- Addition $(+)$ between two vectors of $\mathbb{G}^k$ denotes the adddition coefficient by coefficient and substraction $(-)$ denotes the inverse of the addition.

We define the Elisabeth-4 encryption scheme as its key generation, encryption and decryption algorithm. Both encryption and decryption use the keystream algorithm.

Algorithm 1: Elisabeth-4 .KeyGen()

```
output: Secret key sk ∈ G^256
begin
    sk ←$ G^256
    return sk
```

Algorithm 2: Elisabeth-4 .KeyStream(sk)

```
input  : Secret key sk ∈ G^256
         Integer k ∈ N
output: Stream stream ∈ G^k
begin
    for i in range(k) do
        // aborted Knuth shuffling
        for j in range(60) do
            r = random_int(j, 256)
                // Use a forward-secure PRG as described in [34]
            w = random_int(0, 16)          // Random element of G
            swap(s[j], s[r])
            s[j] = (s[j] + w)%16                          // Whitening
        Acc = 0
        for b in range(12) do
            for j in range(5) do
                x_j = s_{5b+j}
            for j in range(4) do
                y_j = S_j [x_j + x_{(j+1)%4}]
            r = 0
            for j in range(4) do
                r += S_{4+j} [x_j + y_{(j+1)%4} + y_{(j+2)%4}]
            r += x_4
            Acc += r
        stream_i = Acc
    return stream
```

Algorithm 3: Elisabeth-4 .Enc(m, sk)

```
input  : Message m ∈ G^k
         Secrek key sk ∈ G^256
output: Ciphertext c ∈ G^k
begin
    s = Elisabeth-4 .KeyStream(sk)
    c = m + s
    return c
```

Algorithm 4: Elisabeth-4 .Dec(c, sk)

input : Ciphertext $c \in \mathbb{G}^k$
Secrek key $sk \in \mathbb{G}^{256}$
output: Message $m \in \mathbb{G}^k$
begin
  $s =$ Elisabeth-4 .KeyStream(sk)
  $m = c - s$
  return m