Boolean functions for homomorphic-friendly stream ciphers

Claude Carlet[1] and Pierrick Méaux[2],

[1] Department of Informatics, University of Bergen, Norway and LAGA, University of Paris 8, France
claude.carlet@gmail.com
[2] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium
pierrick.meaux@uclouvain.be

Abstract. The proliferation of small embedded devices having growing but still limited computing and data storage facilities, and the related development of cloud services with extensive storage and computing means, raise nowadays new privacy issues because of the outsourcing of data processing. This has led to a need for symmetric cryptosystems suited for hybrid symmetric-FHE encryption protocols, ensuring the practicability of the FHE solution. Recent ciphers meant for such use have been introduced, such as LowMC, Kreyvium, FLIP, and Rasta. The introduction of stream ciphers devoted to symmetric-FHE frameworks such as FLIP and its recent modification has in its turn posed new problems on the Boolean functions to be used in them as filter functions. We recall the state of the art in this matter and present further studies (without proof) [3] .

1 Introduction

The cloud has become nowadays an unavoidable complement to a variety of embedded devices such as mobile phones, smart cards, smart-watches, as these cannot perform all the storage and computing needed by their use. This raises a new privacy concern: it must be impossible to the cloud servers to learn about the data of the users. The first scheme of *fully homomorphic encryption* (*FHE*) realized by Gentry [Gen09] gives a solution to this problem by providing an encryption scheme $\mathbf{C}^H$ preserving both operations of addition and multiplication:

$$\mathbf{C}^H(m+m') = \mathbf{C}^H(m) \boxplus \mathbf{C}^H(m');\ \mathbf{C}^H(mm') = \mathbf{C}^H(m) \boxdot \mathbf{C}^H(m'). \tag{1}$$

Then, combining these two operations allows to evaluate any polynomial over the algebraic structure where m and m' live, allowing to perform any computation if this structure is a finite field, or even if it is a vector space over a finite field, since we know that any function over such structure is (univariate, resp. multivariate) polynomial (see [KLP06]). Let us recall how such scheme can be used if one wants to compute the image of some data by some function, and needs the help of the cloud for that. We first represent the data by elements m_i of a finite field $\mathbb{F}_q$ (or a ring but we shall restrict ourselves to a field), where $i \in I \subseteq \mathbb{N}$; the function, transposed as a function over $\mathbb{F}_q^I$, that we shall denote by $F(m_i, i \in I)$, becomes then a polynomial, according to what we recalled above (or according to the fact that the vector space $\mathbb{F}_q^I$ can be identified with the field $\mathbb{F}_{q^{|I|}}$). If one needs the help of the cloud for the computation, it is sufficient to send $\mathbf{C}^H(m_i)$ for $i \in I$ to the cloud server, which will compute $F(\mathbf{C}^H(m_i), i \in I)$. Thanks to (1), this value will equal $\mathbf{C}^H(F(m_i, i \in I))$ and decryption by the owner of the private decryption key will provide $F(m_i, i \in I)$, and the server will have not learned anything about the m_i nor about $F(m_i, i \in I)$. The computation to perform is transposed as a function F over this field since the homomorphic operations allowed by a FHE scheme are only defined for this field (or ring), and it does not allow to perform other operations using different representation of the data. For example, a FHE scheme for plaintexts from $(\mathbb{F}_{2^n}, +, \times)$ cannot

[3] this paper is an extended version of the invited talk given by Claude Carlet at Algebra, Codes and Cryptology (A2C) 2019

handle the plaintexts as elements from $\mathbb{F}_2^n$, therefore in this case any computation is evaluated based on the univariate representation.
But the theoretical solution we just described is not practical by itself, because the repetitive use of homomorphic encryption (and even in most cases a single use!) requires itself too much computational power and storage capacity than what can offer a device like those listed above. In practice, FHE schemes come from noise-based cryptography such as schemes relying on the learning with errors assumption [Reg05]. Each ciphertext contains an error part, also called noise, hiding the relation with the plaintext and the secret key. When a plaintext is encrypted, the error corresponds to a vector of low norm, and a ciphertext can be decrypted correctly until the amount of noise reaches a fixed bound. When homomorphic operations are performed on ciphertexts, the noise increases. The cornerstone of FHE is a function called bootstrapping, that allows to obtain a ciphertext for m with a low noise from a noisy ciphertext of m. This function corresponds to homomorphically performing the decryption of the FHE scheme and requires an extra key. Bootstrapping is the most costly algorithm of a FHE cryptosystem in terms of computation and storage, and two different strategies are used to get around this bottleneck. The most spread strategy consists in minimizing the number of bootstrappings during an evaluation; the parameters are taken to allow a bounded number of operations (or levels) on fresh ciphertexts before bootstraping. The best performances are obtained by expressing the functions to evaluate in a way minimizing the noise growth. A more recent strategy initiated by [DM15], later referred as gate-bootstrapping, performs a bootstrapping at each operation, amortizing the cost by combining the two functions (the operation and the bootstrapping) at once. Such strategy can lead to good performances when function F can be evaluated as a circuit with a limited number of gates. For both strategies, some functions F imply a low noise growth (and can be qualified as homomorphic-friendly, but the functions qualified this way in the title of this paper are also functions involved in the hybrid symmetric-FHE encryption itself; they need then extra properties, see below); these functions depend on the particular FHE scheme chosen.
The main drawback of FHE constructions is the huge expansion factor, the ratio between the size of the plaintext (in bits) and the size of the corresponding ciphertexts. The expansion factor can be as big as $1.000.000$, and it implies the major constraints for small devices. Indeed, doing computations on small devices with these ciphertexts is challenging, and only a limited number of homomorphic ciphertexts can be handled at the same time. A solution to this problem for the user, traditionally called Alice, is to use a *hybrid symmetric-FHE encryption* protocol:

1. Alice sends to the server her public key pk^H associated to the chosen homomorphic encryption protocol and the ciphertext $\mathbf{C}^H(\mathsf{sk}^S)$ corresponding to the homomorphic encryption of her key sk^S associated to a chosen symmetric encryption scheme $\mathbf{C}^S$,
2. she encrypts her data m with $\mathbf{C}^S$, and sends $\mathbf{C}^S(m)$ to the server,
3. the server computes $\mathbf{C}^H(\mathbf{C}^S(m))$ and homomorphically evaluates the decryption of the symmetric scheme on Alice's data; it obtains $\mathbf{C}^H(m)$,
4. the server homomorphically executes polynomial function F on Alice's data, and obtains $\mathbf{C}^H(F(m))$,
5. the server sends $\mathbf{C}^H(F(m))$ to Alice who obtains $F(m)$ by decrypting (whose operation is much less costly than encrypting in FHE).

Such symmetric-FHE framework allows Alice to circumvent the huge costs implied by the expansion factor. In this context she uses homomorphic encryption only in the first step (on a small data, the symmetric key), and then she uses symmetric encryption and homomorphic decryption. Both of these algorithms are operations that can be efficiently performed on limited devices. Consequently, the performance of the whole hybrid framework is mainly determined by the third and fourth steps. Since the homomorphic evaluation of the symmetric decryption algorithm is independent of the applications wanted by Alice, one

natural goal consists in making its over-cost as low as possible. It would make correspond the time cost of the framework to the computations of the fourth step only, linking the performances to the complexity of computations delegated. As explained above, for reasons of efficiency, the choice of the symmetric cipher itself $\mathbf{C}^S$ is central in this matter, since its decryption should keep as low as possible the noise of homomorphic ciphertexts. The error-growth given by the basic homomorphic operations is different for each FHE scheme but some general trends can be exhibited, therefore giving guidelines to understand which symmetric schemes are homomorphic-friendly. For the schemes often qualified as second generation such as [BV11,BGV12,FV12], the sum corresponds to adding the noises whereas the error produced in a product is way more important. Each level in the tree representing the multiplication corresponds to a level of noise ([HS14]), hence the final noise is often well approximated by relating it to the multiplicative depth of the circuit evaluated, or equivalently, $\lceil \log(d)_2 \rceil$ where d is the degree of the (univariate, resp. multivariate) polynomial evaluated (note that there is no ambiguity on the degree since the homomorphic operations are valid over one field or ring only, therefore allowing only one representation). The homomorphic-friendly functions for this generation are the ones with a low multiplicative depth (and a bounded number of additions). For the schemes following the blueprint of [GSW13] such as [KGV14, CGGI16], referred as third generation, the error-growth of a product is asymmetric in the input ciphertexts. This property allows to obtain a small noise for the result of many products, with some limitations. More precisely, products of ciphertexts where always one (of both) has a small noise results in a small noise. This more complex error-growth gives access to other homomorphic-friendly functions, beyond the restrictions of very small degree. Examples of homomorphic-friendly functions for this generation are given by the sums of successive products, or combinations of multiplexers using a fresh variable.
When the decryption function of a symmetric scheme is evaluated as a polynomial in one field rather than combining computations over different representations, such as alternating operations from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ and operations from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^n}$ on a value from the same register, it often leads to a high degree and many terms. For example, the multiplicative depth of AES is often too large, and its additive depth is still more, to efficiently evaluate it homomorphically. Thus, other symmetric encryption schemes have been proposed in the context of symmetric-FHE frameworks: some block ciphers, like *LowMC* [ARS$^+$16], *Rasta* and *Agrasta* [DEG$^+$18], and stream ciphers such as *Kreyvium* [CCF$^+$16]. These solutions have drawbacks: Kreyvium becomes more and more expensive during the encryption since the noise in the produced ciphertexts increases (or the system has to be reboot often). LowMC provides low noise at each round, but the iteration of rounds makes it unadapted, as almost any other block cipher since the lower bound on the round number for security reverberates on the homomorphic evaluation. We can observe this impact by studying how they can work with HElib [HS14] for instance, where the number of homomorphic levels required is always at least the number of rounds. This is however a minor drawback in this generation (HElib implements the FHE of [BGV12]) for Rasta and Agrasta, since they allow a very small number of rounds. These schemes are also well adapted for *multiparty computation*, but not for all FHE, for example their high number of sums are not well suited for the third generation. In this paper, we focus our study on symmetric encryption schemes that could be tailored for any FHE scheme, and more precisely on the functions used in these homomorphic-friendly constructions.

The *FLIP* cipher is an also very efficient encryption scheme, described in [MJSC16], which tries to minimize the noise involved in homomorphic evaluation. More precisely it intents to optimize the parameters mentioned above, targeting the most homomorphic-friendly functions which are sufficient to ensure security (for example minimizing the multiplicative depth). This scheme is based on a new stream cipher model, called the *filter permutator* (see Figure 1). It consists in updating at each clock cycle a key register by a permutation of the coordinates. A pseudorandom number generator (PRNG) pilots the choice of the permutation. The permuted key is then filtered, like in a classical stream cipher, by a Boolean function f whose output provides the *keystream*. Note that the input to f is the whole key register and this is a difference with the classical way of using filter functions. Applying the non-linear filtering function directly on the key bits allows to greatly reduce the noise level in the framework of hybrid symmetric-FHE encryption protocols. More precisely, the noise is given by the evaluation of one function only: the filtering function, rather than by the combination of all the functions used in the decryption algorithm as for other schemes. In theory, there are no big differences between the filter model and the filter permutator: the LFSR is simply replaced by a permutator. Nevertheless, in practice there are huge differences since the filter function has hundreds of input bits instead of about 20.

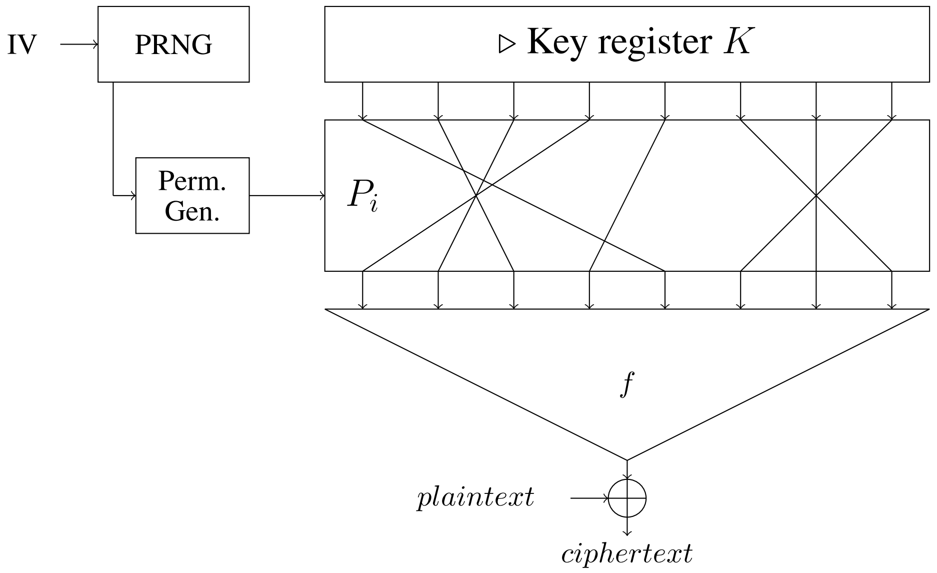

Fig. 1. Filter permutator construction.

In the versions of the cipher proposed in [MJSC16], the function f has $n = n_1 + n_2 + n_3 \geq 500$ variables, where n_2 is even and n_3 equals $\frac{k(k+1)}{2}t$ for some integers k and t. The functions $f(x_0, \ldots, x_{n_1-1}, y_0, \ldots, y_{n_2-1}, z_0, \ldots, z_{n_3-1})$ is defined as:

$$\sum_{i=0}^{n_1-1} x_i + \sum_{i=0}^{n_2/2-1} y_{2i}\, y_{2i+1} + \sum_{j=0}^{t-1} T_k\left(z_{\frac{jk(k+1)}{2}}, z_{\frac{jk(k+1)}{2}+1}, \ldots, z_{\frac{jk(k+1)}{2}+\frac{k(k+1)}{2}-1}\right),$$

where the *triangular function* T_k is defined as:

$$T_k(z_0, \ldots, z_{j-1}) = z_0 + z_1 z_2 + z_3 z_4 z_5 + \cdots + z_{\frac{k(k-1)}{2}} \cdots z_{\frac{k(k+1)}{2}-1}.$$

The filter permutator has been improved in [MCJS19] (see Figure 2). There are two modifications. Firstly, at each clock cycle, the function is applied on a part of the key rather than on the whole key register and, secondly, a public vector (called whitening) is added before the computation of f. The subset of the key register used and the whitening are derived from the PRNG's output at each clock cycle, like the permutation. These modifications have no impact on the noise when the cipher decryption is homomorphically evaluated, the final noise is the one given by the evaluation of f only. On the security side, the resulting register extension allows to obtain the same security with simpler functions, and the whitening allows to temper the attacks using guess-and-determine strategies. The combination of both makes possible to study the security more easily, relating it with the Boolean cryptographic criteria of f, and those of the functions obtained by fixing variables in the input to f.

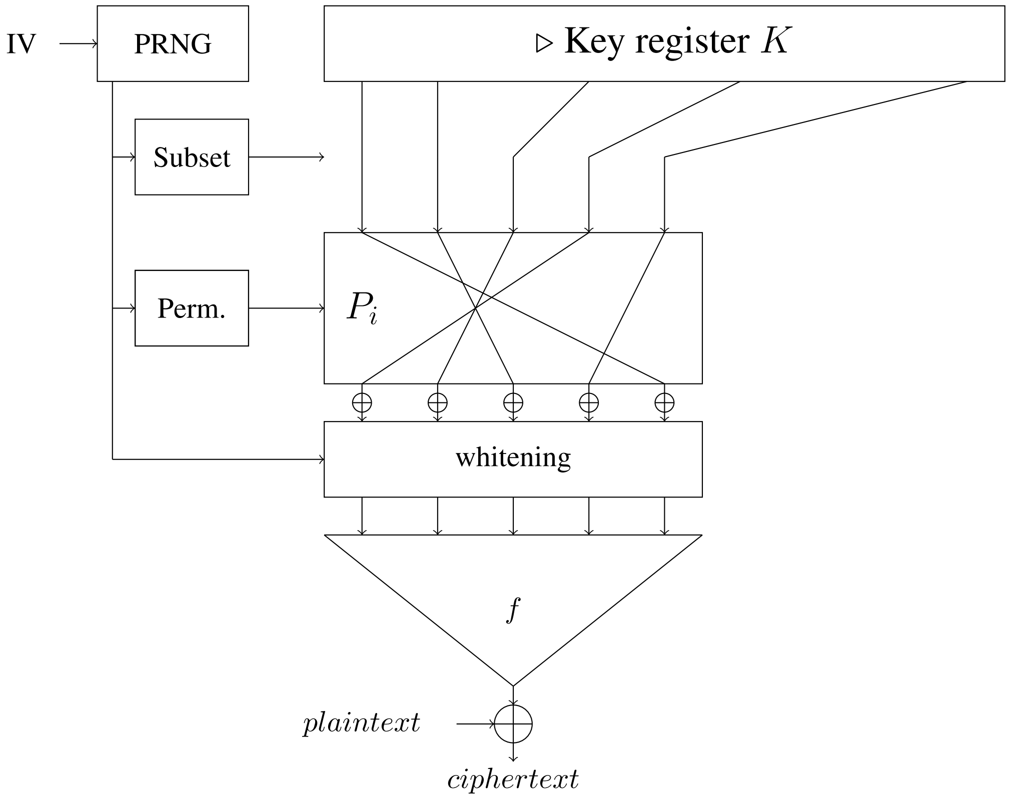

Fig. 2. Improved filter permutator construction.

The attacks which classically apply on the filter model, or slightly modified ones, can apply on the filter permutator or its improved version. Then, the usual Boolean cryptographic criteria have to be taken into consideration for the choice of f. More precisely, for algebraic attacks or variants, the parameters of algebraic immunity, fast algebraic immunity, and number of annihilators of minimal algebraic degree are important. Standard correlation attacks do not apply on these models, but some variations can; this motivates to address the resiliency and nonlinearity of f. Since, unlike the filter model, the key register in the filter permutator is not updated, guessing some key bits importantly simplifies the system of equations given by the keystream. It makes attacks using guess-and-determine strategies more efficient [DLR16] than regular ones. Bounding the complexity of these attacks necessitates to determine the cryptographic parameters of the functions obtained by fixing various variables in f. These security considerations bring us to focus on families of functions with more variables than usually and whose sub-function parameters can be determined or at least be efficiently bounded.

In this article we give without proof the relevant cryptographic parameters for two families of homomorphic-friendly functions. Functions from these families enable to securely instantiate the filter permutator or the improved filter permutator, and allow a very efficient homomorphic evaluation.

2 Preliminaries.

For readability we use the notation $+$ instead of $\oplus$ to denote addition in $\mathbb{F}_2$. We denote $\{1, \dots, n\}$ by $[n]$.

2.1 Boolean Functions and Cryptographic Criteria.

Boolean Functions. We recall here some core notions on Boolean functions in cryptography, restricting our study to the single-output Boolean functions.

Definition 1 (Boolean Function). *A Boolean function f in n variables is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. The set of all Boolean functions in n variables is denoted by $\mathcal{B}_n$. We call pseudo-Boolean function a function with input space $\mathbb{F}_2^n$ but output space different from $\mathbb{F}_2$ (e.g. $\mathbb{R}$).*

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

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

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

Every Boolean functions has a unique ANF. The degree of this unique ANF is called the algebraic degree of the function and denoted by $\mathsf{deg}(f)$.

Boolean Criteria. In this part, we recall the main cryptographic properties of Boolean functions, mostly taken from [Car10]: balancedness, resiliency, nonlinearity, algebraic immunity, fast algebraic immunity and minimal degree annihilator space's dimension.

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

Definition 4 (Resiliency). *A Boolean function $f \in \mathcal{B}_n$ is called 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) of f and set $\mathsf{res}(f) = -1$ if f is unbalanced.*

Definition 5 (Nonlinearity). *The nonlinearity $\mathsf{NL}(f)$ of a Boolean function $f \in \mathcal{B}_n$, where n is a positive integer, is the minimum Hamming distance between f and all the affine functions 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 $g(x) = a \cdot x + \varepsilon$; $a \in \mathbb{F}_2^n, \varepsilon \in \mathbb{F}_2$ (where $\cdot$ is some inner product in $\mathbb{F}_2^n$; any choice of an inner product will give the same value of $\mathsf{NL}(f)$).

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

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

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

We additively use the notation $\mathsf{AN}(f)$ for the minimum algebraic degree of non null annihilator of f:

$$\mathsf{AN}(f) = \min_{g \neq 0}\{\mathsf{deg}(g) \mid fg = 0\}.$$

We also use the notation $\mathcal{D}\mathsf{AN}(f)$ for the dimension of the vector space made of the annihilators of f of degree $\mathsf{AI}(f)$ and the zero function. Note that, for every function f we have $\mathcal{D}\mathsf{AN}(f) \leq \binom{n}{\mathsf{AI}(f)}$, because two distinct annihilators of algebraic degree $\mathsf{AI}(f)$ cannot have in their ANF the same part of degree $\mathsf{AI}(f)$ (their difference being itself an annihilator).

Definition 7 (Fast Algebraic Immunity [ACG⁺06]). *The fast algebraic immunity of a Boolean function $f \in \mathcal{B}_n$, denoted as $\mathsf{FAI}(f)$, is defined as:*

$$\mathsf{FAI}(f) = \min\{2\mathsf{AI}(f), \min_{1 \leq \mathsf{deg}(g) < \mathsf{AI}(f)} \mathsf{deg}(g) + \mathsf{deg}(fg)\}.$$

Families of Boolean Functions. In the very constrained framework in which we are, where we need to maximize the ratio $\frac{security}{complexity}$ of our functions in a much more drastic way than for classical stream ciphers, we highlight three families of functions: direct sum of monomials, threshold functions, and XOR-Threshold functions. We begin by introducing the secondary construction (*i.e.* construction of functions using already defined functions as building blocks) called direct sum, which will lead to the first of these families. This secondary construction is usually considered as unadapted to the design of cryptographic Boolean functions, because the *decomposability* of the functions it provides may be used in attacks. But in our framework, the number of variables is more than 500 (while in classical stream ciphers it is about 20) and this changes the situation. Moreover, the direct sum will lead us to a quite interesting class of functions (direct sums of monomials), well adapted to our framework, and for which we shall be able to determine the cryptographic parameters of all functions in the class. This is the first time that all functions in a whole class of functions can be evaluated (with the exception of the Maiorana-McFarland class, see [Car10], but the functions in this latter class do not have quite good algebraic immunity). Before, the contributions of the papers were to construct functions achieving provably good characteristics; the corresponding classes contained only one or at most a few functions in each number of variables. Concretely, in most cases, these functions had optimal algebraic immunity (this was necessary because of the rather small number of variables). Here we shall have much more flexibility for the choice of functions within the class.

Definition 8 (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:*

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

The direct sum has been generalized into the indirect sum (see [Car10]) which provides more complex functions, better adapted to classical stream ciphers. It seems that using the indirect sum does not allow to have simple enough functions for our framework nor to determine exactly the cryptographic parameters of all the functions in a class. We focus more precisely on direct sums of monomials, which consist of functions where each variable appears at most once in the ANF.

Definition 9 (Direct Sum of Monomials). *Let f be a non constant Boolean function of n variables, we call f a Direct Sum of Monomials (or DSM) if the following holds for its ANF:*

$$\forall (I, J) \text{ such that } a_I = a_J = 1,\, I \cap J \in \{\emptyset, I \cup J\}.$$

Definition 10 (Direct Sum Vector [MJSC16]). *Let f be a DSM, we define its direct sum vector:*

$$\mathbf{m}_f = [m_1, m_2, \ldots, m_k],$$

of length $k = \mathsf{deg}(f)$, where m_i is the number of monomials of degree i, $i > 0$, of f:

$$m_i = |\{a_I = 1, \text{ such that } |I| = i\}|.$$

The function f associated to the direct sum vector $\mathbf{m}_f = [m_1, m_2, \ldots, m_k]$ has $M = \sum_{i=1}^{k} m_i$ monomials in its ANF and has $N \geq \sum_{i=1}^{k} im_i$ variables.

As we wrote, we shall be able to determine all parameters of all functions in this class.

We also define the family of threshold functions, which is a super-class of that of majority functions.

Definition 11 (Threshold Function). *For any positive integers $d \leq n+1$ we define the Boolean function $\mathsf{T}_{d,n}$ as:*

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

Definition 12 (Majority Function). *For any positive odd integer n we define the Boolean function MAJ_n as:*

$$\forall x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n, \quad \mathsf{MAJ}_n(x) = \begin{cases} 0 & \text{if } \mathsf{w_H}(x) \leq \lfloor \frac{n}{2} \rfloor, \\ 1 & \text{otherwise.} \end{cases}$$

Note that threshold functions are symmetric functions (changing the order of the input bits does not change the output), which have been the focus of many studies *e.g.* [Car04, CV05, DMS06, QLF07, SM07, QFLW09]. Note also that $\mathsf{MAJ}_n = \mathsf{T}_{\lceil \frac{n+1}{2} \rceil, n}$. These functions can be described more succinctly through the simplified value vector.

Definition 13 (Simplified Value Vector). *Let f be a symmetric function in n variables, we define its simplified value vector:*

$$\mathbf{s} {=} [w_0, w_1, \ldots, w_n]$$

of length n, where for each $k \in \{0, \ldots, n\}$, $w_k = f(x)$ where $\mathsf{w_H}(x) = k$, i.e. w_k is the value of f on all inputs of Hamming weight k.

Note that for a threshold function, we have $w_k = 0$ for $k < d$ and 1 otherwise, so the simplified value vector of a threshold function $\mathsf{T}_{d,n}$ is the $n+1$-length vector of d consecutive 0's and $n+1-d$ consecutive 1's.

We will also be interested in functions obtained by a direct sum of a linear direct sum of monomials and a threshold function, called XOR-THR (or XOR-MAJ when the threshold function happens to be a majority function).

Definition 14 (XOR-THR Function). *For any positive integers k, d and n such that $d \leq n+1$ we define $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ for all $z = (x_1, \ldots, x_k, y_1, \ldots, y_n) \in \mathbb{F}_2^{k+n}$ as:*

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

Boolean Functions and Bit-Fixing. In this part, we give the necessary vocabulary relatively to bit-fixing (as defined e.g. in [AL16]) on Boolean functions, the action consisting in fixing the value of some variables of a Boolean function and then considering the resulting Boolean function. These notions are important when guess-and-determine attacks are investigated.

Definition 15 (Bit-fixing Descendant). *Let f be a Boolean function in n variables (x_i, for $i \in [n]$), let ℓ be an integer such that $0 \leq \ell < n$, let $I \subset [n]$ be of size ℓ (i.e. $I = \{i_1, \dots, i_\ell\}$ with $i_j < i_{j+1}$ for all $j \in [\ell - 1]$), and let $b \in \mathbb{F}_2^\ell$, we denote as $f_{I,b}$ the ℓ-bit fixing descendant of f on subset I with binary vector b the Boolean function in $n - \ell$ variables:*

$$f_{I,b}(x') = f(x) \mid \forall j \in [\ell],\ x_{i_j} = b_j,$$

where $x' = (x_i)_{i \in [n] \setminus I}$.

Definition 16 (Bit-fixing Stability). *Let $\mathcal{F}$ be a family of Boolean functions, $\mathcal{F}$ is called bit-fixing stable, or stable relatively to guess and determine, if for all functions $f \in \mathcal{F}$ such that f is a n-variable function with $n > 1$, the following holds:*

- *for all number of variables ℓ such that $0 \leq \ell < n$,*
- *for all choice of positions $1 \leq i_1 < i_2 < \dots < i_\ell \leq n$,*
- *for all value of guess $(b_1, \dots, b_\ell) \in \mathbb{F}_2^\ell$,*

at least one of these properties is fulfilled: $f_{I,b} \in \mathcal{F}$, or $f_{I,b} + 1 \in \mathcal{F}$, or $\mathsf{deg}(f_{I,b}) = 0$.

Remark 1. Both DSM and XOR-THR functions are bit-fixing stable families (as well as the set of threshold functions). Therefore, if the cryptographic parameters of any functions of one of these families are determined then the complexity of all attacks using guess-and-determine strategies on any filtering function of this family can be derived.

3 Parameters of Direct Sums of Monomials.

In this section we give the relevant parameters relatively to Boolean cryptographic criteria of DSM functions. The DSM are a generalization of triangular and FLIP functions [MJSC16]. Their very sparse ANF is the reason why they are homomorphic-friendly. Note that such function in n variables can be computed with at most $n - 1$ additions and multiplications. Regarding the satisfaction of the constraints of the second generation of FHE schemes, secure functions can have a multiplicative depth as low as 2 or 3 ([MCJS19]). Focusing on the third generation, each monomial of the ANF can be evaluated as a serial multiplication of freshy encrypted ciphertexts, then the whole function is evaluated as a sum of multiplicative chains, giving an error-growth quasi-additive in n ([MCJS19]).

On the cryptographic point of view, the DSM can be obtained by recursively applying the direct sum construction, which is convenient to determine the resiliency and nonlinearity, but not to study the exact behavior of the algebraic properties such as the algebraic immunity and the dimension of annihilators (non null) of minimal degree.

First we recall some properties on direct sums (see *e.g.* [MJSC16]).

Lemma 1 (Direct Sum Properties ([MJSC16] Lemma 3)). *Let F be the direct sum of f and g with n and m variables respectively. Then F has the following cryptographic properties:*

1. *Resiliency:* $\mathsf{res}(F) = \mathsf{res}(f) + \mathsf{res}(g) + 1$.

2. *Non Linearity:* $\mathsf{NL}(F) = 2^m\mathsf{NL}(f) + 2^n\mathsf{NL}(g) - 2\mathsf{NL}(f)\mathsf{NL}(g)$.
3. *Algebraic Immunity:* $\max(\mathsf{AI}(f), \mathsf{AI}(g)) \leq \mathsf{AI}(F) \leq \mathsf{AI}(f) + \mathsf{AI}(g)$.
4. *Fast Algebraic Immunity:* $\mathsf{FAI}(F) \geq \max(\mathsf{FAI}(f), \mathsf{FAI}(g))$.

The previous lemma is sufficient to determine the resiliency and the nonlinearity of any direct sums of monomials.

Lemma 2 (Resiliency of Direct Sum of Monomials). *Let $f \in \mathbb{F}_2^n$ be a Boolean function obtained by direct sums of monomials with associated direct sum vector $= [m_1, \dots, m_k]$, its resiliency is:*

$$\mathsf{res}(f) = m_1 - 1$$

Lemma 3 (Nonlinearity of Direct Sum of Monomials). *Let $f \in \mathbb{F}_2^n$ be a Boolean function obtained by direct sums of monomials with associated direct sum vector $\mathbf{m}_f = [m_1, \dots, m_k]$, its nonlinearity is:*

$$\mathsf{NL}(f) = 2^{n-1} - \frac{1}{2}\left(2^{(n-\sum_{i=2}^{k} im_i)} \prod_{i=2}^{k} \left(2^i - 2\right)^{m_i}\right)$$

Now, we give the algebraic immunity, a lower bound on the FAI and an upper bound on th $\mathcal{D}\mathsf{AN}$ of a direct sum of monomials.

Theorem 1 (Algebraic Immunity of Direct Sums of Monomials). *Let $f \in \mathbb{F}_2^n$ be a Boolean function obtained by direct sums of monomials with associated direct sum vector $\mathbf{m}_f = [m_1, \dots, m_k]$, its algebraic immunity is:*

$$\mathsf{AI}(f) = \min_{0 \leq d \leq k} \left(d + \sum_{i=d+1}^{k} m_i\right).$$

Lemma 4 (Fast Algebraic Immunity of Direct Sums of Monomials). *Let $f \in \mathbb{F}_2^n$ be a Boolean function obtained by the direct sum of monomials with associated direct sum vector $\mathbf{m}_f = [m_1, \dots, m_k]$ such that $\mathsf{AI}(f) = \mathsf{deg}(f)$, and $\mathsf{AI}(f) > 1$, its fast algebraic immunity is:*

$$\mathsf{FAI}(f) = \begin{cases} \mathsf{AI}(f) + 1 & \textit{if } m_k = 1, \\ \mathsf{AI}(f) + 2 & \textit{otherwise.} \end{cases}$$

Note that this lemma does not consider the case $\mathsf{AI}(f) = 1$ (of linear functions or monomial functions), for this case the fast algebraic immunity is not very relevant as the algebraic attack already targets a linear system.

Theorem 2. *Let f be a DSM with associated direct sum vector $\mathbf{m}_f = [m_1, \dots, m_k]$. Let us consider the set $\mathsf{S_d}(f)$ such that:*

$$\mathsf{S_d}(f) = \begin{cases} \{0 \leq d \leq k \mid d + \sum_{i>d} m_i = \mathsf{AI}(f)\} & \textit{if } m_1 \neq 1, \\ \{0 < d \leq k \mid d + \sum_{i>d} m_i = \mathsf{AI}(f)\} & \textit{if } m_1 = 1. \end{cases}$$

Then, we have the following relation:

$$\mathcal{D}\mathsf{AN}(f) \leq \sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{k+1} i^{m_i}.$$

Note that when $\mathsf{AN}(f) = \mathsf{AI}(f)$ the bound is reached.

This formula gives a tight upper bound on the dimension of the annihilators of a DSM.

4 Parameters of Threshold, and XOR-THR Functions.

In this section we exhibit the parameters of threshold functions and xor-threshold functions. The threshold functions are a generalization of majority functions, which are known to reach the optimal algebraic immunity. Despite a very dense ANF, other representations can lead to low noise ciphertexts, as shown in [MCJS19] inspired by branching programs. Using multiplexers, the noise for third generation schemes is quasi additive in the number n of variables of the threshold function. Combining it in direct sum with a xor function allows to compensate the resiliency, and also to connect it with the predicates considered secure to instantiate local PRGs ([Gol00], [App13]). Therefore, the xor-threshold functions are appealing as filtering functions, and the threshold functions are an intermediate step to exhibit the cryptographic parameters of xor-threshold functions. Regarding homomorphic evaluation, the noise mostly depends on the threshold part, therefore for the third generation the noise is quasi additive in the number of variables. Focusing on the second generation of FHE, secure instantiations from xor-threshold functions can be chosen with a multiplicative depth between 3 and 7 ([MCJS19]).

On the cryptographic point of view, the parameters of majority functions are well investigated, but lesser is known on the properties of threshold functions in general, such as their nonlinearity. The direct sum with a xor function allows to derive the resiliency and nonlinearity from the parameters of the threshold function. Nevertheless, the exact immunity and dimension of annihilators (non null) of minimal degree require more than the general results of direct sum constructions.

4.1 Threshold Functions

Threshold functions are symmetric functions, which have been much studied relatively to cryptographic significant criteria (*e.g.* [CV05]). The existence of optimal symmetric functions relatively to a specific criterion has been widely investigated, but the class of symmetric functions is too wide for their parameters to be studied globally; here we focus on the exact parameters of the subfamily of threshold function.

We first give the resiliency and nonlinearity of such functions.

Lemma 5 (Resiliency of Threshold Functions). *Let f be the threshold function $\mathsf{T}_{d,n}$,*

$$\mathsf{res}(\mathsf{T}_{d,n}) = \begin{cases} 0 & \text{if } n = 2d-1, \\ -1 & \text{otherwise.} \end{cases}$$

Theorem 3 (Nonlinearity of Threshold Functions). *Let n be a non null positive integer, the threshold function $\mathsf{T}_{d,n}$ has the following nonlinearity:*

$$NL(\mathsf{T}_{d,n}) = \begin{cases} 2^{n-1} - \binom{n-1}{(n-1)/2} & \text{if } d = \frac{n+1}{2}, \\ \sum_{k=d}^{n} \binom{n}{k} = \mathsf{w_H}(\mathsf{T}_{d,n}) & \text{if } d > \frac{n+1}{2}, \\ \sum_{k=0}^{d-1} \binom{n}{k} = 2^n - \mathsf{w_H}(\mathsf{T}_{d,n}) & \text{if } d < \frac{n+1}{2}. \end{cases}$$

We then investigate the algebraic immunity of threshold functions. As Boolean functions used for cryptography, the majority functions have been introduced as functions reaching the optimal algebraic immunity (case of $\mathsf{T}_{(n+1)/2,n}$ and $\mathsf{T}_{n/2+1,n}$ as proven in [BP05,DMS06]), but their nonlinearity is not good. As far as we know, the exact algebraic immunity have not been investigated for all threshold functions, but it can be determined in various ways as for the majority functions.

Lemma 6 (Algebraic Immunity of Threshold Functions). *Let n be a non null positive integer, the threshold function $\mathsf{T}_{d,n}$ has the following algebraic immunity:*

$$\mathsf{AI}(\mathsf{T}_{d,n}) = \min(d, n-d+1).$$

Lemma 7 ($\mathcal{D}\mathsf{AN}$ of Threshold Functions). *Let n be a non null positive integer, the threshold function $\mathsf{T}_{d,n}$ has the following $\mathcal{D}\mathsf{AN}$:*

$$\mathcal{D}\mathsf{AN}(\mathsf{T}_{d,n}) = \begin{cases} 0 & \textit{if } d < \frac{n+1}{2}, \\ \binom{n}{d-1} & \textit{if } d \geq \frac{n+1}{2}. \end{cases}$$

Corollary 1 (Lower Bound on the Fast Algebraic immunity of Threshold Functions). *Let n be a non null positive integer, the fast algebraic immunity of the threshold function $\mathsf{T}_{d,n}$ follows the following bound:*

$$\mathsf{FAI}(\mathsf{T}_{d,n}) \geq \begin{cases} \min(2d, n-d+2) & \textit{if } d \leq \frac{n+1}{2}, \\ \min(2(n-d+1), d+1) & \textit{if } d > \frac{n+1}{2}. \end{cases}$$

Remark 2. Note that this bound can be reached, as proven in [TLD16] for the majority functions $\mathsf{T}_{2^{m-1},2^m}$ and $\mathsf{T}_{2^{m-1}+1,2^m+1}$ for all integers $m \geq 2$.

4.2 Parameters of XOR-THR Functions.

The particular structure of XOR-THR functions make the resiliency and nonlinearity parameters easy to determine from the ones of these two components.

Lemma 8 (Resiliency of XOR-THR Functions). *Let f be the XOR-THR function $\mathsf{XOR}_k + \mathsf{T}_{d,n}$, then:*

$$\mathsf{res}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \begin{cases} k & \textit{if } n = 2d-1, \\ k-1 & \textit{otherwise.} \end{cases}$$

Lemma 9 (Nonlinearity of XOR-THR Functions). *Let f be the XOR-THR function $\mathsf{XOR}_k + \mathsf{T}_{d,n}$, then:*

$$\mathsf{NL}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \begin{cases} 2^{n+k-1} - 2^k\binom{n-1}{(n-1)/2} & \textit{if } d = \frac{n+1}{2}, \\ 2^k \displaystyle\sum_{i=d}^{n} \binom{n}{i} & \textit{if } d > \frac{n+1}{2}, \\ 2^k \displaystyle\sum_{i=0}^{d-1} \binom{n}{i} & \textit{if } d < \frac{n+1}{2}. \end{cases}$$

Then we focus on the exact algebraic immunity of these functions, a lower bound on the fast algebraic immunity, and the exact $\mathcal{D}\mathsf{AN}$.

Lemma 10 (Algebraic Immunity of XOR Threshold Functions). *Let f be the $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ function:*

$$\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \begin{cases} \frac{n+1}{2} & \textit{if } d = \frac{n+1}{2}, \\ \min\{k,1\} + \min\{d, n-d+1\} & \textit{otherwise.} \end{cases}$$

Lemma 11 (Fast Algebraic Immunity of a XOR-THR Function). *Let f be the $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ function: if $k = 0$, then:*

$$\mathsf{FAI}(\mathsf{XOR}_0 + \mathsf{T}_{d,n}) \geq \min(2\min(d, n-d+1), 1 + \max(d, n-d+1)).$$

If $k > 0$

$$\mathsf{FAI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) \geq \begin{cases} \frac{n+3}{2} & \text{if } d = \frac{n+1}{2}, \\ 2 + \min(d, n-d+1) & \text{otherwise.} \end{cases}$$

Lemma 12 ($\mathcal{D}\mathsf{AN}$ of XOR-THR Functions). *Let $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ be a XOR-THR function such that $k > 0$, $n \in \mathbb{N}$, and $1 \leq d \leq n$, then:*

$$\mathcal{D}\mathsf{AN}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \begin{cases} \binom{n}{d} & \text{if } d < \frac{n}{2}, \\ \binom{n+1}{d+1} & \text{if } d = \frac{n}{2}, \\ \binom{n}{d} & \text{if } d = \frac{n+1}{2}, \\ \binom{n+1}{d} & \text{if } d = \frac{n}{2} + 1, \\ \binom{n}{d-1} & \text{otherwise.} \end{cases}$$

Furthermore $\mathcal{D}\mathsf{AN}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \mathcal{D}\mathsf{AN}(1 + \mathsf{XOR}_k + \mathsf{T}_{d,n})$.

References

ACG$^+$06. Frederik Armknecht, Claude Carlet, Philippe Gaborit, Simon Künzli, Willi Meier, and Olivier Ruatta. Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*. Springer, Heidelberg, May / June 2006.

AL16. Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*. ACM Press, June 2016.

App13. Benny Applebaum. Cryptographic hardness of random local functions-survey. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, page 599. Springer, Heidelberg, March 2013.

ARS$^+$16. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. *IACR Cryptology ePrint Archive*, 2016:687, 2016.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

BP05. An Braeken and Bart Preneel. On the algebraic immunity of symmetric boolean functions. In *Progress in Cryptology - INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005, Proceedings*, pages 35–48, 2005.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

Car04. Claude Carlet. On the degree, nonlinearity, algebraic thickness, and nonnormality of boolean functions, with developments on symmetric functions. *IEEE Trans. Information Theory*, pages 2178–2185, 2004.

Car10. Claude Carlet. *Boolean Functions for Cryptography and Error-Correcting Codes*, page 257397. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2010.

CCF$^+$16. Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*. Springer, Heidelberg, March 2016.

CGGI16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2016.

CV05. Anne Canteaut and Marion Videau. Symmetric boolean functions. *IEEE Trans. Information Theory*, pages 2791–2811, 2005.

DEG$^+$18. Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A cipher with low anddepth and few ands per bit. In *CRYPTO 2018*, pages 662–692, 2018.

DLR16. Sébastien Duval, Virginie Lallemand, and Yann Rotella. Cryptanalysis of the FLIP family of stream ciphers. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 457–475. Springer, Heidelberg, August 2016.

DM15. Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015.

DMS06. Deepak Kumar Dalai, Subhamoy Maitra, and Sumanta Sarkar. Basic theory in construction of boolean functions with maximum possible annihilator immunity. *Designs, Codes and Cryptography*, 2006.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `http://eprint.iacr.org/2012/144`.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

Gol00. Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

HS14. Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, LNCS, pages 554–571. Springer, Heidelberg, August 2014.

KGV14. Alhassan Khedr, Glenn Gulak, and Vinod Vaikuntanathan. SHIELD: Scalable homomorphic implementation of encrypted data-classifiers. Cryptology ePrint Archive, Report 2014/838, 2014.

KLP06. T. Kasami, Shu Lin, and W. Peterson. Polynomial codes. *IEEE Trans. Inf. Theor.*, 14(6):807–814, September 2006.

MCJS19. Pierrick Méaux, Claude Carlet, Anthony Journault, and Franois-Xavier Standaert. Improved filter permutators: Combining symmetric encryption design, boolean functions, low complexity cryptography, and homomorphic encryption, for private delegation of computations. Cryptology ePrint Archive, Report 2019/483, 2019. `https://eprint.iacr.org/2019/483`.

MJSC16. Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 311–343. Springer, Heidelberg, May 2016.

QFLW09. Longjiang Qu, Keqin Feng, Feng Liu, and Lei Wang. Constructing symmetric boolean functions with maximum algebraic immunity. *IEEE Trans. Information Theory*, pages 2406–2412, 2009.

QLF07. Longjiang Qu, Chao Li, and Keqin Feng. A note on symmetric boolean functions with maximum algebraic immunity in odd number of variables. *IEEE Transactions on Information Theory*, 53, 2007.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

SM07. Palash Sarkar and Subhamoy Maitra. Balancedness and correlation immunity of symmetric boolean functions. *Discrete Mathematics*, pages 2351 – 2358, 2007.

TLD16. Deng Tang, Rong Luo, and Xiaoni Du. The exact fast algebraic immunity of two subclasses of the majority function. *IEICE Transactions*, pages 2084–2088, 2016.