Improved Filter Permutators: Combining Symmetric Encryption Design, Boolean Functions, Low Complexity Cryptography, and Homomorphic Encryption, for Private Delegation of Computations

Pierrick Méaux[1], Claude Carlet[2], Anthony Journault[1], and François-Xavier Standaert[1]

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

Abstract. Motivated by the application of delegating computation, we revisit the design of filter permutators as a general approach to build stream ciphers that can be efficiently evaluated in a fully homomorphic manner. We first introduce improved filter permutators that allow better security analyses, instances and implementations than the previously proposed FLIP family of stream ciphers. We also put forward the similarities between these improved constructions and a popular PRG design by Goldreich. Then, we exhibit the relevant cryptographic parameters of two families of Boolean functions, direct sums of monomials and XOR-MAJ functions, which give candidates to instantiate the improved filter permutator paradigm. We develop new Boolean functions techniques to study them, and refine Goldreich's PRG locality bound for this purpose. We give an asymptotic analysis of the noise level of improved filter permutators instances using both kind of functions, and recommend them as good candidates for evaluation with a third-generation FHE scheme. Finally, we propose a methodology to evaluate the performance of such symmetric cipher designs in a FHE setting, which primarily focuses on the noise level of the symmetric ciphertexts (hence on the amount of operations on these ciphertexts that can be homomorphically evaluated). Evaluations performed with HElib show that instances of improved filter permutators using direct sums of monomials as filter outperform all existing ciphers in the literature based on this criteria. We also discuss the (limited) overheads of these instances in terms of latency and throughput.

1 Introduction.

1.1 Initial Goal: Delegating Computations.

Delegating computations has become an important and common habit in our very connected world. From its simplest form, outsourcing data, to advanced forms such as complex computations on multi-user aggregated data, delegating computation is widely spread in our society. One of the reason of this massive use is that, despite the surrounding of more and more devices able to perform computations and communications, the specifications of such machines are going in two opposite directions. On one side, many objects are now called smart, or connected, capable of storing a limiting amount of data and send it regularly to bigger devices. Typical examples of these devices are bracelets, watches, or even pacemakers or toothbrushes. On the other side, some companies are acquiring more and more servers, enabling to provide huge storage capacity or computational power, often sold as Cloud services. The boom on the data produced by each user, together with the increasing need of processing these data cannot be handled by small devices, or even by personal machines. Then, it is a main cause of the generalization of delegation of computation that the recent years are witnessing.

Outsourcing computation seems to be a solved problem, nevertheless combining it with privacy is way trickier. Giving a total control on personal data allows to perform any operation on it, however keeping hidden some parts of the data greatly increases the difficulty of this task. One way of solving it is to use

Fully Homomorphic Encryption (FHE), which first scheme have been exhibited in 2009 by Gentry [Gen09]. The principle of this primitive is to allow to perform any operation on encrypted versions of the data which correspond to operation on the actual data, without knowing its value. With this primitive, an user can encrypt all its data and send it to a Cloud. Then, the cloud can store and perform some operations on the encrypted data, and send a encrypted version of the result to the user. The Cloud does not learn the value of the sent data, neither the result of the computation it performs. The user can decrypt the encrypted message from the Cloud, learning a result she could not compute, from data she did not needed to keep anymore. It answers to her needs of storage and complex computations, keeping the data privacy.

But, all magic comes with a price. Gentry proved the existence of FHE based on reasonable assumptions, and further works developed this result *e.g.* [BV11, BGV12, Bra12, GSW13] but there are some drawbacks for the concrete efficiency. Known constructions are parts of noise-based cryptography, such as Lattice-based cryptography, where the security of the scheme generally reduces to the Learning With Error problem [Reg05], or a variant. Informally, the principle of these encryption schemes consists in hiding the plaintext with an error. Then, performing operations on the ciphertexts applies on the plaintexts and makes increase the error (also called noise). When the error is to big to allow a correct decryption, a technique called boostraping is used to decrease the noise of the ciphertext. This technique relates to homomorphically perform the decryption of the FHE scheme on its own ciphertext, which implies important costs in terms of time and data. This technique is the main bottleneck in FHE evaluation, generally leading to consider contexts where few or no boostrapping are considered for concrete applications. The second drawback, is greatly impacting asymmetric situations such as the case of outsourcing computation. Fully homomorphic encryption has an important expansion factor, which corresponds to the size of a ciphertext required to encrypt one bit. The expansion factor makes the storage of FHE ciphertext costly for the user, and the encryption algorithm a quite costly algorithm. In contrast, FHE decryption has often a lighter cost in data and in time complexity for the user.

1.2 Hybrid Homomorphic Framework.

These different drawbacks lead to neglect FHE in favor of an hybrid approach for delegating computation, often called hybrid FHE. Firstly envisaged in [LNV11], the principle of this primitive is to consider the combination of a Symmetric Encryption (SE) scheme with a FHE. The typical framework between the user and the Cloud is depicted in different works (*e.g.* [LNV11, MJSC16, CCF$^+$16], and can be extended to different frameworks with more actors. The first main difference with the standard FHE protocol is for the user, which encrypts only once homomorphically. The user encrypts the key of the symmetric scheme and send it to the Cloud. Then, all the data is sent symmetrically encrypted to the Cloud, which uses the encrypted symmetric key and the encrypted plaintexts to obtain homomorphic encryption of the data. The Cloud realizes this part by homomorphically evaluating the decryption of the symmetric encryption scheme. Finally, with the obtained homomorphic ciphertexts, the protocol can follow the one of standard FHE.

The evaluation of the decryption algorithm by the Cloud is called transciphering, and it is the most costly part of the framework, and it is the main focus of hybrid FHE. However, in this framework, the functionality interesting the user corresponds to all the processing performed by the Cloud after this point. Then, the efficiency, or satisfiability, of the delegation of computations should depend on the later steps. It translates into two main constraints regarding transciphering. First, the time cost of this step should be small relatively to the computations, it can represent only a negligible part of the total framework. Second, transciphering should involve a very low impact on the noise, in order not to restrict the quantity of operations performed by the Cloud in reasonable time (*i.e.* with 0 or few bootstrappings). More practically, the time and error growth depends on how efficiently the symmetric decryption algorithm can be homomorphically evaluated.

When 0 or few bootstrappings are allowed, it implies that the ciphertext sizes has to be big enough to handle more operations, or more directly bigger errors. The ciphertext sizes affect the time and data cost of all homomorphic operations, during the whole framework. Thus, the management of the noise is the main ingredient determining the efficiency of the outsourcing computation application, leading to often wonder: What about the noise? Hence, a SE scheme is as appropriate for Hybrid FHE as its decryption algorithm produces a small noise, and is quickly homomorphically evaluated. The error evolution through the operations depends on the FHE, and different families (or generations) of FHE gives different metrics of noise. For the schemes known up to now, the basis of the computations considered are additions and products, the second one being the most costly. The name of second generation (2G) is often used for schemes where the noise can be approximated by levels given by the multiplicative depth, such as [BGV12]. The third generation (3G), such as GSW scheme [GSW13], benefits from an asymmetric error growth for multiplication, leading to a quasi-additive noise for multiplicative chains.

In a first time the SE schemes that have been considered for Hybrid FHE were standard symmetric schemes, known to have a relatively small multiplicative depth. In these cases, the performance has been mostly evaluated in terms of speed, considering which scheme can provide homomorphic ciphertexts quickly, based on libraries such as HElib [HS14] working on 2G FHE schemes such as [FV12, BGV12]. Most of the evaluated schemes were block ciphers such as AES [GHS12, CLT14], Simon [LN14] and Prince [DSES14]. The stream cipher Trivium has also been considered due to is slowly increasing multiplicative complexity [CCF+16]. These works focused on the minimal time necessary to produce homomorphic ciphertexts, or ciphertexts allowing a small fixed number of further multiplications. Interested in outsourcing computation, the situation is different, we care on schemes enabling most of the computations after the transciphering, which is possible using more recent SE schemes, the ones designed for advanced cryptographic primitives.

1.3 Symmetric Encryption for Advanced Primitives.

Block cipher designs with reduced multiplicative complexity (*e.g.* number of AND gates per ciphertext bit or AND depth) have recently attracted significant attention in symmetric cryptography research. Such ciphers are motivated by new constraints raised by emerging security applications. For example, limited multiplicative complexity allows preventing side-channel attacks via masking more efficiently [PRC12, GGNS13, GLSV14], can improve the throughput and latency of Multi-Parti Computation (MPC) protocols [ARS+15, GRR+16], and mitigates the noise increase and the ciphertext expansion in FHE schemes [ARS+15, ARS+16, CCF+16, MJSC16, DEG+18]. Concretely, thanks to innovative (and sometimes aggressive) design choices, recent ciphers (*e.g.* LowMC [ARS+15, ARS+16]) can encrypt with as little of four ANDs per bit, or with a multiplicative depth of four (*e.g.* FLIP [MJSC16]). In a recent work by Dobraunig et al. [DEG+18], the authors even go as far as minimizing both metrics jointly for a single cipher.

The use of symmetric encryption for advanced primitives gave birth to many new schemes with the principle of consisting in simpler algorithms to benefit to the advanced primitive. Such advanced primitives are FHE indeed, but also MPC and Zero-Knowledge (ZK) proofs. These new SE schemes are sometimes designed for all these applications such as LowMC [ARS+15, ARS+16] or Rasta [DEG+18] by focusing on the number of ANDs. Some focus on the compatibility with FHE mostly, such as Kreyvium [CCF+16] or FLIP [MJSC16]. An important line of work focuses both on MPC and ZK, going out of the binary extensions considered for most of the SE, such examples are given by MiMC [AGR+16] or the recent Marvellous suite [AD18]. Diverse strategies are used to build these SE schemes with simple, or low cost, encryption and decryption algorithms, but, what about the noise?

The simplicity generally comes from a reduced multiplicative depth and a reduced number of logical gates, mainly a reduction of the number of AND gates. The efficiency relatively to the MPC and ZK application can directly be related to these values. Regarding FHE, the situation is more complex, not only the number of gates imports, the error growth is a complex function depending also on the the order of the computations. The 2G allows to approximate the final error by the multiplicative depth, but this approximation can be falsified when many additions are performed, as witnesses in [CCF$^+$16] for the evaluation of LowMC. The 3G enables to compute long multiplicative chains when low-noise ciphertexts are used [AP14, DM15, CGGI16], whereas multiplying few sums can result in very noisy ciphertexts. These examples show that despite reduced multiplicative level and small number of ANDs could serve as a first direction for hybrid FHE, they are not sufficient for efficient outsourcing delegation.

1.4 Filter Permutators and FLIP.

The approach of [MJSC16] does not focus in decreasing the number of gates or the multiplicative depth but in reducing the noise of the decryption algorithm to the evaluation of one Boolean function only, it is the principle of the Filter Permutator (FP) paradigm. Encryption (or decryption) for this stream cipher paradigm consists, for each keystream bit, in applying a different public wire-cross permutation to the secret key, and then to filter this permuted key by the (unique) filtering function. The permutation being publicly derived (from a Pseudorandom Generator (PRG)) the error growth comes from the evaluation of the function (the addition of the keystream bit with the plaintext or ciphertext has a negligible influence). This feature seems optimal relatively to hybrid FHE, the efficiency of the framework relying on the homomorphic evaluation of one function only. Finding a function with minimal noise growth relatively to a FHE scheme sufficient to provide security is then the main goal. FLIP ciphers [MJSC16] correspond to the filter permutator paradigm instantiated relatively to the 3G, but they also give a record of low noise for the 2G as the chosen functions have a very small multiplicative depth.

The FP paradigm is designed for efficient hybrid FHE, but, how far can it goes in this direction rises many questions. The very unusual design requires to investigate different topics which are generally not jointly studied in cryptology. More precisely, this paradigm is at the crossroads of design of SE scheme, low depth cryptographic primitive, study of Boolean functions, and homomorphic evaluation. We develop the main questions of FP relatively to these topics in the following paragraphs, as in this article we present results in all these directions, in order to get efficient hybrid FHE.

First, the security of Filter Permutators is difficult to assess, in [MJSC16] different usual properties of Boolean functions are considered to estimate the complexities of the potential attacks applying on the paradigm. A generalization of the common criteria of Boolean function used in cryptography is also introduced, in terms of recurrent criteria, to handle the impact of guess and determine attacks, exhibited by Duval et al. [DLR16] in this context. Then, the permutation making the input weight of the filtering function invariant, non standard properties of the function have to be investigated, as determined in [CMR17]. Studying the security from the properties of the Boolean function in this context led to a new branch of works on the so-called restricted criteria [CMR17, Mes17, MZD18, MMM$^+$18]. Thus, finding modifications enabling to increase the security of the FP paradigm, or simplifying its analysis would greatly impact its use in hybrid FHE frameworks.

Then, the concrete instantiations of FP given by FLIP for a bit security of 80 and 128 have a multiplicative depth of only 4, way smaller than standard SE schemes. It rises the question of how low can be the multiplicative depth, or the degree, of a concrete SE scheme. Rewriting the filter permutator paradigm puts forward some similarities with a popular PRG design by Goldreich [Gol00]. It highlights a connection between FP and cryptographic primitives existing in low complexity classes such as local PRGs.

These constructions being the focus of many works as surveyed in [App13], investing the connection can lead to results in both directions.

The Boolean functions used to instantiate the FP paradigm have a larger number of variables than usually considered in cryptography. For the functions used in combiners models, investigating the parameters of function in 20 variables was sufficient, looking for the ones reaching optimal parameters. Later on, more recent stream ciphers often use registers of 128 bits, however FP leads to consider functions in several hundreds of variables. General algorithms have prohibitive complexity to determine the parameters of such functions, furthermore when the parameters of sub-functions are also needed, as expected from the concept of recurrent criteria. It leads to consider functions in many variables but with a simple structure, as the FLIP functions. Such families are still almost not investigated in the area of Boolean functions and could lead to new constructions [HKM17], or new results for Boolean functions and symmetric cryptography.

Finally, the homomorphic evaluation of FPs rises two main questions. First, how to efficiently evaluate the error-growth given by functions in many variables. FLIP functions can be expressed by a low number of sums and products, enabling to easily compute the associated noise, but it gives access to a small number of functions. A good alternative in this direction consists in investigating other representations of the functions that could be compatible with homomorphic evaluation, such as branching program or finite automata evaluations [BV14, CGGI16]. Second, most of previous works are comparing the performances of Hybrid FHE protocol based on the latency or throughput of the transciphering part only. As the motivation of this construction is outsourcing computation, it is interesting to investigate if other comparisons can represent more the applicability of the SE scheme for concrete applications. These time estimations give a first intuition on the SE schemes interesting for hybrid FHE but they suffer some limitations. As example, the time required for transciphering, or the time to perform this operation and still allow some level of multiplications, gives few information on the time necessary for more computations, and even less information to compare different SE schemes in this context. Also, optimizing the timings relatively to one level of noise and one particular library leads to over-tailor the homomorphic choices, neglecting security issues or the application goals. These examples lead to investigate a more representative methodology for comparing SE for hybrid FHE, and consequently to compare the previously used schemes accordingly.

1.5 Contributions.

In this article we present different results relative to the four sub-mentioned topics, jointly investigated, in order to provide more efficient hybrid FHE. We firstly modify the FP paradigm in the Improved Filter Permutator (IFP) paradigm. It is performed by first considering two different sizes of register, one for the key, and a smaller one on which the permutation applies. It implies that the filtering function is applied on a different (publicly chosen) subset of the key bits a each clock cycle. This difference modifies the Hamming weight of the input of f during the encryption, which strongly decreases the potential impact of restricted inputs attacks and makes the paradigm more similar to the design of Goldreich's PRG. Then, a whitening is xored to the input of the function, at each clock cycle. This modification greatly complexifies the task of finding guesses leading to particularly weak sub-functions, and randomizes the input of the filtering function f. With these two modifications, we propose a security analysis simpler than the one known on FPs, and algorithms to evaluate it. This analysis uses well known Boolean criteria such as the algebraic immunity on the filtering function and all its sub-functions. We give two potential instantiations of the IFP paradigm, in terms of FiLIP stream ciphers, both considering their security and homomorphic impact. The $\mathsf{FiLIP}_{\mathsf{DSM}}$ ciphers use the Direct Sums of Monomials (DSM) functions, which are generalization of FLIP functions, we consider it as the main ciphers to analyze in the context of Hybrid FHE. Then, we propose and study

$\mathsf{FiLIP}_{\mathsf{XMAJ}}$ challenges which are inspired by functions used for local PRGs, but than could have flaws for concrete (non asymptotic) instances.

Considering $\mathsf{FiLIP}_{\mathsf{XMAJ}}$ challenges we exhibit the similitudes between Goldreich's PRG and the IFP paradigm. We investigate in detail the family of XOR-MAJ functions which are considered as predicates of local PRGs [AL16]. This study leads to partially answer a question relatively to the stretch of local PRGs, giving a new bound on the minimum locality required for these PRGs. The answer being constructive we exhibit two families reaching this bound for all stretch considered.

We investigate two families of Boolean functions for any number of variables, the families of direct sum of monomials and the family of XOR-THR functions. In both cases we determine the exact resiliency, nonlinearity, algebraic immunity and bounds on the fast algebraic immunity and dimension of annihilators of minimal degree, which are all the relevant criteria for IFPs, in in various contexts. For the second family of functions we begin by studying the threshold functions, a family of symmetric functions containing the majority function. Despite the intense study of symmetric functions in the area of Boolean functions used in cryptography we exhibit new properties on these families, potentially answering questions far from our scope. Motivated by the security analysis of IFPs we develop new tools such as the partitioned algebraic normal form, which enables to derive more results on the algebraic immunity, and generally on low degree annihilators that could benefit to other studies.

Finally, we study the error growth of the two instantiations of IFPs for general 3G schemes. We do it considering two possible representation of functions, and show that both families of functions give a very small noise. It leads to very efficient Hybrid FHE relatively to the third generation. We then use the $\mathsf{FiLIP}_{\mathsf{DSM}}$ instances as a basis to compare IFPs with other published ciphers aimed at efficient FHE evaluation. We describe and use a more stable metric than latency and throughput (since it avoids the aforementioned specialization to a given target function). It is also connected to a generally desirable goal (since one may expect that the ability to perform as much homomorphic operations as possible is a useful feature for practical applications). We formalize our comparison methodology, by (*i*) setting the FHE security parameters at a level that is comparable to the SE ones (*i.e.* 80-bit or 128-bit), (*ii*) using ciphertexts of comparable size for all the cipher designs to compare (so basing the comparison on the most expensive cipher in terms of noise), and (*iii*) monitoring the noise (*e.g.* provided by the used library) not only for the ciphertexts but also after one and two levels of additional multiplications on the ciphertexts. Concrete estimations carried out using HElib put forward that the noise of Rasta and FiLIP is orders of magnitudes smaller than the one of LowMC, and that new instances of FiLIP with reduced multiplicative depth allow performing two more levels of multiplications on its ciphertexts than the recommended Rasta designs. We further observe that even non-recommended versions of Rasta (with comparable multiplicative depth) would not compare favorably to FiLIP due to a (much) larger key size. We complement our analyzes with an evaluation of best-case latency and throughput (*i.e.* when ciphertexts can just be decrypted), as performed previously. We believe that it remains an informative alternative metric and clarify that it has to be understood as the best possible performances of a cipher since any concrete application (where symmetric ciphertexts are manipulated homomorphically) will require ciphertext expansion.

1.6 Organisation.

In Section 2 we give the preliminary notions on Boolean function and FHE used in the paper. In Section 3, and Section 4 we introduce the improved filter paradigm and study its security as a symmetric encryption scheme. The following part, Section 5 and Section 6 is devoted to the results on Boolean functions, proving the relevant parameters of the DSM and XOR-THR families, and diverse applications of partitioned algebraic normal form such as the result on local PRGs. In Section 7 we study the error growth of the two families of functions relatively to 3G schemes. Section 8 presents the instantiations of IFPs with DSM functions: the $\mathsf{FiLIP}_{\mathsf{DSM}}$ ciphers, and Section 9 is dedicated to $\mathsf{FiLIP}_{\mathsf{XMAJ}}$ challenges based on XOR-THR function. Finally, in Section 10 we develop the new methodology to compare hybrid FHE, and apply it on the 2G library HElib.

2 Preliminaries.

In addition to classic notation we use the $\log$ to denote the logarithm in basis 2, and $[n]$ to denote the subset of all integers between 1 and n: $\{1, \ldots, n\}$. For readability we use the notation $+$ instead of $\oplus$ to denote the addition in $\mathbb{F}_2$.

2.1 Boolean Functions and Cryptographic Criteria.

Boolean Functions. We introduce here some core notions of Boolean functions in cryptography, restricting our study to the following definition of Boolean function, more restrictive than a vectorial Boolean function.

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

We call pseudo-Boolean function a function with input space $\mathbb{F}_2^n$ but output space different from $\mathbb{F}_2$.

The following representation is commonly used, and its basic properties also.

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 $\mathbb{F}_2[x_1, \ldots, x_n]/(x_1^2+x_1, \ldots, x_n^2+x_n)$):*

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

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

- *The algebraic degree of f equals the global degree $\max_{\{I \mid a_I = 1\}} |I|$ of its ANF.*
- *Any term $\prod_{i \in I} x_i$ in such an ANF is called a monomial and its degree equals $|I|$. A function with only one non-zero coefficient a_I, $|I| > 0$, is called a monomial function.*
- *The function f is affine if and only if its algebraic degree is at most 1, the function is linear if in addition $a_\emptyset = 0$.*

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 dimension of the space of annihilators of minimal degree.

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 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.*

Note that resiliency is an extended notion of balancedness, a balanced function is a k-resilient function with $k \geq 0$. We also define the Hadamard transform, an important tool to study the resiliency of a Boolean function.

Definition 5 (Hadamard Transform). *The Hadamard transform is the linear mapping which maps any pseudo-Boolean function f on $\mathbb{F}_2^n$ (with output space included in $\mathbb{Z}$) to the function $\hat{f}$ defined on $\mathbb{F}_2^n$ as:*

$$\hat{f}(a) = \sum_{x \in \mathbb{F}_2^n} f(x)(-1)^{a \cdot x},$$

where $a \cdot x$ denotes the inner product in $\mathbb{F}_2^n$, and the sum is performed in $\mathbb{Z}$.

The Hadamard transform can be applied to a Boolean function f itself but also to the sign function $f_\chi(x) = (-1)^{f(x)}$, giving the Walsh transform:

Definition 6 (Walsh Transform). *Let $f \in \mathcal{B}_n$ a Boolean function, its Walsh transform W_f at $a \in \mathbb{F}_2^n$ is defined as:*

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

Note that the Walsh transform is strongly connected to the nonlinearity:

Definition 7 (Nonlinearity). *The nonlinearity NL of a Boolean function $f \in \mathcal{B}_n$, where n is a positive integer, is the minimum Hamming distance between f and all the affine functions 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 definition).

The nonlinearity of a Boolean function can also be defined by its Walsh transform:

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

Note that the nonlinearity measures the distance to affine functions. It can be generalized to the notion of higher-order nonlinearity where the distance is taken over all functions of degree less than or equal to a fixed integer.

Definition 8 (Algebraic Immunity and Annihilators). *The algebraic immunity of a Boolean function $f \in \mathcal{B}_n$, denoted as $\mathsf{AI}(f)$, is defined as:*

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

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).

Note that this definition directly leads to the following properties for simple functions:

Corollary 1 (Algebraic Immunity Properties). *Let f be a Boolean function:*

- *The null and the all-one functions are the only functions such that $\mathsf{AI}(f) = 0$.*
- *All monomial (non constant) functions f are such that $\mathsf{AI}(f) = 1$.*
- *For all non constant f it holds: $\mathsf{AI}(f) \leq \mathsf{AN}(f) \leq \mathsf{deg}(f)$.*

Definition 9 (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:*

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

Families of Boolean Functions. In this part we highlight three families of functions: direct sum of monomials, threshold functions, and XOR-Threshold functions. We begin by introducing a secondary construction called direct sum, enabling to construct the first family.

Definition 10 (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:*

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

A family of functions obtained by direct sums can be of particular interest when looking for functions simple to evaluate: functions obtained by direct sums of monomials. Informally it consists of functions where each variable appears at most once in the ANF, and we focus on the ones where each variable appears once and only once.

Definition 11 (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:*

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

Definition 12 (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:

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

When we consider a function F associated to the direct sum vector $\mathbf{m}_F = [m_1, m_2, \ldots, m_k]$, it corresponds to the function with $M = \sum_{i=1}^{k} m_i$ monomials, and $N = \sum_{i=1}^{k} i m_i$ variables. Note that it corresponds to a function without ineffective variable, each variable appears once and only once in the ANF.

A sub-family of direct sum of monomials of particular interest is the family of triangular functions.

Definition 13 (Triangular Functions [MJSC16]). *Let k be a strictly positive integer. The k-th triangular function T_k is a direct sum of monomials of $k(k+1)/2$ variables:*

$$T_k(x_1, \dots, x_{k(k+1)/2}) = \sum_{i=1}^{k} \prod_{j=1}^{i} x_{j+i(i-1)/2}.$$

It can also be defined from its direct sum vector which is the all-1 vector of length k: $\mathbf{m}_{T_k} = [1, 1, \dots, 1]$.

We also define the family of threshold functions, and a sub-family of threshold functions of particular interest is the family of majority functions.

Definition 14 (Threshold Function). *For any positive integers $d \le n+1$ we define the Boolean function $\mathsf{T}_{d,n}$ as:*

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

Definition 15 (Majority Function). *For any positive odd integer n we define the Boolean function MAJ_n as:*

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

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 $\mathsf{MAJ}_n = \mathsf{T}_{\frac{n+1}{2},n}$. These functions can be described more succinctly through the simplified value vector.

Definition 16 (Simplified Value Vector). *Let f be a symmetric function in n variables, we define its simplified value vector:*

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

of length n, where for each $k \in \{0, \dots, n\}$, $w_k = f(x)$ 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 17 (XOR-THR Function). *For any positive integers k, d and n such that $d \le n+1$ we define $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ for all $z = (x_1, \dots, x_k, y_1, \dots, y_n) \in \mathbb{F}_2^{k+n}$ as:*

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

Boolean Functions and Bit-Fixing. In this part, we give the necessary vocabulary relatively to bit-fixing (as defined in [AL16]) on Boolean function, 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 (see Section 4.1).

Definition 18 (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. $I = \{I_1, \ldots, I_\ell\}$ with $I_i < I_{i+1}$ for all $i \in [\ell - 1]$), 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:*

$$f_{I,b}(x') = f(x) \mid \forall i \in [\ell],\ x_{I_i} = b_i,$$

where $x' = (x_i$, for $i \in [n] \backslash I)$.

Definition 19 (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 the variables $1 \leq I_1 < I_2 < \cdots < I_\ell \leq n$,*
- *for all value of guess $(b_1, \ldots, b_\ell) \in \mathbb{F}_2^\ell$,*

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

Remark 1. Both DSM and XOR-THR functions are bit-fixing stable families. More precisely, for a DSM, considering the behavior on its ANF, fixing a variable to 0 cancels a monomial, fixing a variable to 1 reduces the degree of one of the monomials. Then, the property on the ANF coefficients defining a DSM is still complied by the descendant function. Fixing variables recursively does not change this property, and when ℓ is greater than the number of monomials, it is possible to have only the constant coefficient non null, adding the constant functions to the list of descendants.

For the family of XOR-THR functions, first note that fixing variables maintains the direct sum structure. If a variable is fixed to 0 in the XOR part, the descendant has a XOR part with one variable less and the threshold part is the same. If the variable is fixed to 1, the descendant has a XOR part with one variable less and the threshold part is the complement of the initial one, therefore $1 + f'$ is a XOR-THR function. If a variable is fixed in the threshold part, it gives a threshold function. Indeed, for $n > 1$ using Definition 14, fixing a variable to 1 for $\mathsf{T}_{d,n}$ gives the function $\mathsf{T}_{d-1,n-1}$, and fixing a variable to 0 gives the function $\mathsf{T}_{d,n-1}$. Therefore these descendants are also XOR-THR functions. Then, recursively fixing variables until $\ell < n$ gives descendants which are XOR-THR functions or which complement is a XOR-THR functions (note that the constant functions are in this family too).

2.2 Fully Homomorphic Encryption.

We recall here the definition of (fully) homomorphic encryption, a kind of encryption enabling to perform computations on plaintexts only manipulating the ciphertexts, without requiring the ability of decrypting. We introduce the vocabulary relative to homomorphic encryption we will use in this paper. For more details we refer to [Gen09] for FHE, and to [LNV11, MJSC16] for hybrid homomorphic encryption.

Definition 20 (Homomorphic Encryption Scheme). *Let $\mathcal{M}$ be the plaintext space, $\mathcal{C}$ the ciphertext space and λ the security parameter. A homomorphic encryption scheme consists of four probabilistic polynomial-time algorithms:*

- $H.\mathsf{KeyGen}(1^\lambda)$. *Generates a pair* $(\mathsf{pk}^H, \mathsf{sk}^H)$ *the public and secret keys of the scheme.*
- $H.\mathsf{Enc}(m, \mathsf{pk}^H)$. *From the plaintext* $m \in \mathcal{M}$ *and the public key, outputs a ciphertext* $c \in \mathcal{C}$.
- $H.\mathsf{Dec}(c, \mathsf{sk}^H)$. *From the ciphertext* $c \in \mathcal{C}$ *and the secret key, outputs* $m' \in \mathcal{M}$.
- $H.\mathsf{Eval}(f, c_1, \cdots, c_k, \mathsf{pk}^H)$. *With* $c_i = H.\mathsf{Enc}(m_i, \mathsf{pk}^H)$ *for* $1 \leq i \leq k$, *outputs a ciphertext* $c_f \in \mathcal{C}$.

Homomorphic encryption: simple, leveled, somewhat, fully. Different notions of homomorphic encryption exist, depending on the set over which the function f can be taken, that is, on the operations which are possible. For all these kinds of homomorphic encryptions we assume a compactness property: $|\mathcal{C}|$ is finite, and the size of a ciphertext does not depend on the number of homomorphic operations performed to obtain it. When only one kind of operation is permitted the scheme is simply homomorphic, it is called somewhat homomorphic when more than one operation can be performed, at least partially. Leveled homomorphic encryption schemes correspond to f being any polynomial of bounded degree(defining the level) and bounded coefficients. Fully Homomorphic Encryption (FHE) corresponds to f being any function defined over $\mathcal{M}$. Gentry [Gen09] proved that FHE can be constructed by combining a leveled homomorphic encryption scheme with a bootstrapping technique.

As this technique is still a bottleneck for homomorphic evaluation, we consider a framework where no bootstrapping (or at least less bootstrapping) are performed, and then when we refer to FHE or HE it refers more precisely to this context.

Noise or error-growth. Any known FHE scheme is based on noise-based cryptography, so that an homomorphic ciphertext is associated to a part of error (or noise). The more homomorphic operations are performed, the higher is the noise (if no bootstrapping is used), this quantity of noise can be measured in terms of standard deviation of the distribution followed by the error part. The error-growth involved in an homomorphic evaluation is then the evolution of this parameter.

FHE generations. Since Gentry's breakthrough [Gen09], various FHE schemes following this blueprint appeared. We call second generation the schemes where the error of the product is symmetric in the factors, as BGV [BGV12] which is often considered for efficiency comparisons as implemented in the HElib library [HS14]. We call third generation the schemes where the error of the product is asymmetric in the factors, the most recent generation of FHE, initiated with GSW [GSW13].

2.3 Filter Permutators and FLIP Instances.

The Filter Permutator or FP (by analogy with filter generators) is the general design of stream ciphers introduced in [MJSC16], and FLIP is an instance of this design where the filtering function is taken from a sub-family of DSM functions. The main design principle of FPs is to filter a constant key register with a variable (public) bit permutation. More precisely, at each cycle, the key register is (bitwise) permuted with a pseudo-randomly generated permutation, and then a non-linear filtering function is applied to the output of this permuted key register. The general structure of FPs is depicted in Figure 2.3. It is composed of three parts:

- A register where the key is stored,
- a (bit) permutation generator parametrized by a Pseudo Random Number Generator (PRNG) which is initialized with a public IV,
- a filtering function which generates a key-stream.

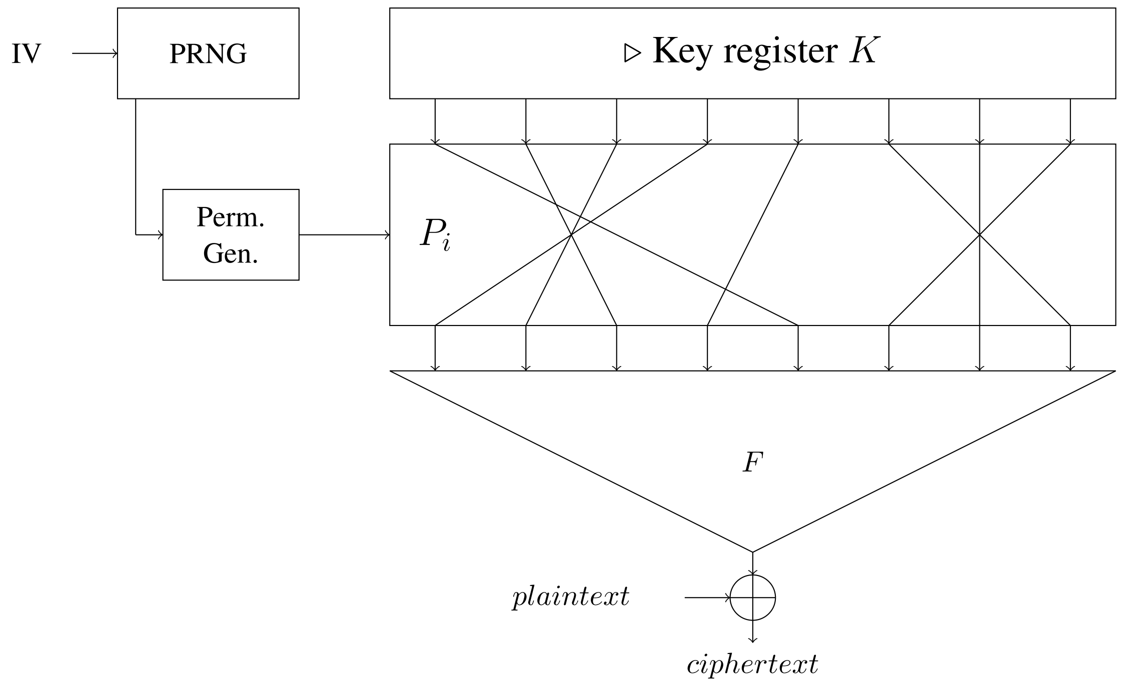

Fig. 1. Filter permutator construction.

3 Improved Filter Permutators: a New Design for Better Security and Better Performances.

Two main tweaks are performed on the Filter Permutators blueprint to increase its security and its performances as a SE scheme in the SE-FHE framework. The first goal of these modifications is to generalize the original design, in a way which provides more flexibility to choose the functions used, and the number of variables involved in the computations. The second goal consists in simplifying the security analysis, erasing some particularities of the FP which make the security difficult to evaluate.

3.1 Description.

The design of Improved Filter Permutators (IFPs) deviates from filter permutators blueprint in two ways. First, the size of the key register and the number of variables of the filtering function is not forced to be equal. The IFP key can be longer than the number of inputs of the filtering function (in practice we consider a small factor between both, between 2 and 32). Second, at each clock cycle a whitening of the size of F input's is derived from the PRNG and bitwise XORed with the permuted sub-part of the key.

It gives a new design depicted in Figure 3.1 with the following particularities:

- N is the size of the key register,
- $n \leq N$ is the number of selected bits from the key register at each clock cycle,
- F is the filtering function, a n-variable Boolean function.

For a security parameter λ, to encrypt $m \leq 2^{\lambda}$ bits under a secret key $K \in \mathbb{F}_2^N$ (such that $\mathsf{w_H}(K) = N/2$), the public parameters of the PRNG are chosen and then the following process is executed for each key-stream bit s_i (for $i \in [m]$):

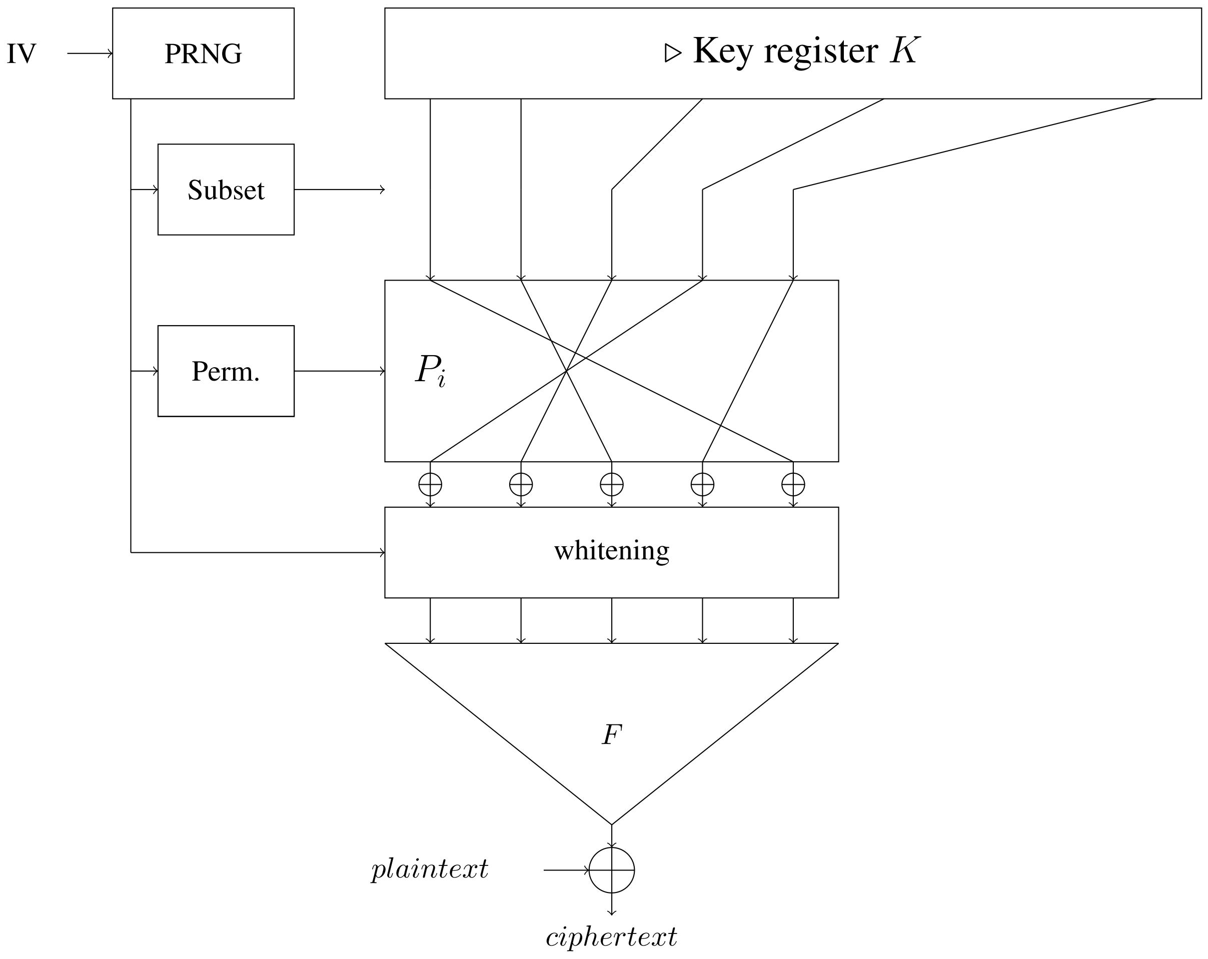

Fig. 2. Improved filter permutator construction.

- The PRNG is updated, its output determines the subset, the permutation, and the whitening at time i,
- the subset S_i is chosen, as a subset of n elements over N,
- the permutation P_i from n to n elements is chosen,
- the whitening w_i from $\mathbb{F}_2^n$ is chosen,
- the key-stream bit s_i is computed as $s_i = F(P_i(S_i(K)) + w_i)$, where $+$ denotes the bitwise XOR.

Note that for each clock cycle i we consider that the PRNG gives enough pseudorandom bits to independently determine the subset ($\log \binom{N}{n}$ bits), the permutation ($\log(n!)$ bits), and the whitening (n bits). Its effect on the performances of IFPs in a hybrid FHE framework is negligible anyway. Note that if the number of pseudorandom bits given by the instance of the PRNG used is limited to b, it enables to compute $\lfloor b/(\log \binom{N}{n} + \log(n!) + n) \rfloor$ bits of ciphertexts only. If this quantity is smaller than m, then another instance of PRNG is used, and so forth until the m bits of ciphertexts are produced (an instantiation of the whole scheme is given in Section 8). Any pseudorandom sequence not adversarially chosen could be used instead of the PRNG's output, the use of the PRNG is only motivated by the storage limitation [MJSC16] of one of the participants in the hybrid FHE framework.

3.2 Impact on Security.

The two modifications from FPs to IFPs, *i.e.* the register extension and the whitening, are generalizing the design, and strictly improving the security. The register extension has two main advantages. First, it enables to increase the security without using more complex functions (allowing then more flexibility in the design). Indeed, keeping invariant the filtering function, increasing N decreases the probability of each key-bit to appear in a key-stream equation, directly increasing the complexity of all attacks known to apply on the Filtering Permutator. Second, the Hamming weight of F's input is not constant anymore. Since $N \geq 2n$, F can be evaluated on any element of $\mathbb{F}_2^n$, it makes the attacks based on restricted input considerations [CMR17] even less efficient.

The main advantage of the whitening is to facilitate the analysis of security against guess-and-determine attacks [DLR16]. When a guess-and-determine strategy is used by the attacker, some key bits (ℓ) are fixed and then the key-stream bits do not correspond to evaluations of F anymore, but to evaluations of descendants of F, which are functions acting on a number of variables between n and $n-\ell$. The complexity of these attacks depends on the properties of the descendants rather than the ones of F. In the security analysis of [MJSC16], the descendant with the worst parameter was considered for each Boolean criterion, giving a lower bound on the complexity of the corresponding attack. By randomizing the choice of the descendant, the whitening enables the security of IFPs to be based on average properties rather than worst-case ones (as the probability of getting a function with the worst parameters is not equal to 1).

Finally, note that increasing the register size makes the construction very similar to Goldreich's PRG [Gol00]. For more details on this PRG, we refer to the initial article of Goldreich and to the survey of Applebaum [App13] relatively to local PRG. In the following we give the necessary explanations to understand the connection between this PRG and IFPs. Goldreich's PRG is an asymptotic construction with interesting conjectured security [App12, App13, AL16], and many implications such as secure computation with constant computational overhead [IKOS08], or indistinguishability obfuscation [LV16, LT17]. We can define Goldreich's PRG in the following way: let n and m be two integers, let $(S_1, \ldots, S_m)$ be a list of m subsets of $[n]$ of size d, and let P be a Boolean function in d variables (often called predicate), we call Goldreich's PRG the functions $G : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ such that for $x \in \mathbb{F}_2^n$, $G(x) = P(S_1(x)), P(S_2(x)), \ldots, P(S_m(x))$. The integer d is called the locality of the PRG and many works have focused on constant d, and on polynomial-stretch local PRG. Local means that d is constant, and polynomial-stretch means that $m = n^s$ where s is the called the stretch, so that these PRG extend a short random seed into a polynomially longer pseudorandom string. These polynomial-stretch local PRG are conjectured secure based on some properties of the subsets and on the function P. Considering the (n, m, d)-hypergraph given by the subsets $(S_1, \ldots, S_m)$, the PRG cannot be secure if the hypergraph is not sufficiently expending (we refer to the survey [App13] for the notions and references). In practice, an overwhelming portion of (n, m, d)-hypergraphs are sufficiently expanding, making the choice of a random (n, m, d)-hypergraph an usual and adequate strategy. Relatively to the function P, the PRG cannot be secure if P is not resilient enough [MST03] or if its algebraic degree, or more generally its algebraic immunity, is not sufficient [AL16], both quantity being related to s. For these constructions, the security is considered asymptotically, relatively to class of polynomial adversaries as linear distinguishers [MST03] or the Lasserre/Parrilo semidefinite programming hierarchy for example. Regarding concrete parameters, very few is known up to now, we are only aware of the recent work [CDM$^+$18], which concretely studies the security of an instance of a super-linear (but less than quadratic) stretch.

3.3 Impact on Homomorphic Evaluation.

The modifications from FPs to IFPs are almost free. The size of the key register does not modify the function F so the homomorphic error-growth given by the evaluation of F is independent of N. The whitening is given by the output of the PRNG, so considered as public, therefore each bit of the whitening is encrypted as a zero-noise homomorphic ciphertext. Adding homomorphically these zero-noise ciphertexts to the input of F does not increase the error-growth, giving a final noise identical to the one obtained with a FP instantiated with the same function. Only the time of the evaluation is modified, but the search in a longer list and the addition of zero-noise ciphertexts has a minor impact compared to the evaluation of the filtering function.

3.4 Key-size Consideration.

A general idea behind FPs and Improved FPs is to have the main part of the encryption process which would have no cost when homomorphically evaluated. This specificity leads to consider longer keys than the traditional λ-bits key for a bit-security of λ. We argue that in the SE-FHE context this specificity has a very low impact. Indeed, even bounding the total key-size to 2^{14} it is still way smaller that the size of only one homomorphic ciphertext. Then, the encryption of each bit depending only on a subpart of fixed length of the key, the total length of the key has no impact for the majority of the hybrid FHE framework. Since the user can store a key of this size, and the server can store this amount of homomorphic ciphertexts, the key size is not a bottleneck in the considered framework. Note that for the schemes with key size of λ bits, more computations are needed for the encryption or decryption, having an important impact on the size of the homomorphic ciphertexts required, impacting the majority of the hybrid FHE framework, and mostly the application part.

4 Security Analysis of the Improved Filter Permutators.

Due to the similarity of (improved) filter permutators to the filter register model, we investigate the attacks known to apply on this model. We consider that no additional weakness arises from the PRNG which is chosen to be forward secure to avoid malleability. The subsets and the whitenings are chosen without any bias and Knuth-shuffle is used to choose the permutations. As a consequence, on this pseudorandom system non adversarially chosen, the attacks applying target the filtering function and they are adaptations from the one applying on filtered registers. The security analysis is similar to the one in [MJSC16], the same kind of attacks are explored but the complexity is computed differently, considering all descendant functions and the probability of obtaining them. We consider the attacks in the single-key setting, in the known ciphertext model, focusing particularly on key-recovery attacks.

4.1 Attacks Applying on Improved Filter Permutators.

We first describe algebraic-like and correlation-like attacks, then we explain how these attacks can be generalized with a guess-and-determine strategy. Finally, we briefly comment other attacks less adapted to IFPs.

Algebraic-like Attacks. We qualify as algebraic-like attacks the kind of attacks consisting in manipulating the system of equations given by the key-stream to build a system of smaller degree, easier to solve. Algebraic attacks [CM03], fast algebraic attacks [Cou03a], or approaches using Grobner bases (such as [Fau99]) are examples of this type of attacks. To determine the security of IFP relatively to this class of attacks we study more particularly the complexity of algebraic attacks and fast algebraic attacks, as their complexity can be estimated from Boolean criteria.

The main idea of algebraic attacks as defined in [CM03] (in a context of filtered LFSR) is to build an over-defined system of equations with the initial state of the LFSR as unknown, and to solve this system with Gaussian elimination. The principle is to find a nonzero function g such that both g and $h = gF$ have low algebraic degree, enabling to get various equations of small degree d. Then, the degree-d algebraic system is solved, by linearization if it is possible, using Grobner basis method or SAT solvers otherwise; linearization is the only method for which evaluating the complexity is easy. In practice, the degree of g is at least $\mathsf{AI}(F)$, and g is chosen to be a non null annihilator of F or $F+1$ of minimal degree. Then the adversary is able to obtain $\mathcal{D}\mathsf{AN}(F)$ (respectively $\mathcal{D}\mathsf{AN}(F+1)$) equations with monomials of degree $\mathsf{AI}(F)$ in the key bits variables, for each equation. After linearization, the adversary obtains a system of equations in $D = \sum_{i=0}^{\mathsf{AI}(F)} \binom{N}{i}$ variables, where N is the number of original indeterminates. Therefore, the time complexity of the algebraic attack is $\mathcal{O}(D^\omega) \approx \mathcal{O}(N^{\omega \mathsf{AI}(F)})$, where ω is the exponent in the complexity of Gaussian elimination (we assume $\omega = \log(7)$ for all our security estimations). The data complexity is $\mathcal{O}(D/\mathcal{D}\mathsf{AN}(F))$.

Fast algebraic attacks [Cou03a] are a variation of the previous attacks. Still considering the relation $gF = h$, their goal is to find and use functions g of low algebraic degree e, possibly smaller than $\mathsf{AI}(f)$, and h of low but possibly larger degree d. Then, the attacker lowers the degree of the resulting equations by an off-line elimination of the monomials of degrees larger than e (several equations being needed to obtain each one with degree at most e). Following [ACG$^+$06], this attack can be decomposed into four steps:

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

Given the FAI of F, ignoring Step 1 which might be trivial for our choice of F, the time complexity of this attack is:

$$O(D\log^2(D) + ED\log(D) + E^\omega) \approx \mathcal{O}(N^{\mathsf{FAI}}).$$

This attack is very efficient on filtered LFSR ciphers as the search of linear relations between equations is simple. For IFPs, as the subset of variables and the permutation chosen at each clock cycle are given by the PRNG, there is no trivial linear relation between one equation and the next ones. It is always possible to simplify some equations using the system, for example forcing collisions on the monomials of higher degree, so other techniques of eliminations could apply. We stress that the time complexity of these techniques would be higher than the one of Berlekamp-Massey, thus we consider the complexity of the fast algebraic attack as an upper bound on the complexity of any attack of the algebraic kind on IFPs. More precisely we consider the time complexity given by D and E in the precedent formula, and we consider a data complexity of D.

Correlation-like Attacks. We qualify as correlation-like attacks the kind of attacks that use the bias of the filtering function relatively to uniform, or relatively to a low degree function. Correlation attacks, Learning Parity with Noise solvers, correlation attacks based on the XL algorithm [Cou03b] are examples of this kind of attacks. To determine the security of IFP relatively to this class of attacks, we study more particularly the complexity of correlation attacks, and show how it complexity can be estimated using Boolean criteria.

The principle of correlation attacks is to distinguish the output of IFPs from random. For example if the filtering function is not balanced an attack can consist in averaging the key-stream and observing a bias relatively to $1/2$. If the function is balanced, this strategy does not apply, but instead of doing an average on all the key-stream, the attack can target one part of the key-stream only, depending on a sub-part of the variables for example. As the goal of these attacks is to distinguish the key-stream from random, then for key-recovery attacks we assume that they have at least the complexity of the distinguishing attack. Two points influence the effectiveness of this attack: the possibility to get equations relatively to an unbalanced function, and the bias.

Two criteria enable to study the functions relatively to these points: the resiliency and the nonlinearity. The resiliency of a function gives the number of variables that have to be fixed to make it unbalanced, and can be used for the first point. Then, the nonlinearity gives the distance with the closest affine function, which determines the bias to $1/2$. Note that to detect the bias to $1/2$ the data complexity would be:

$$\mathcal{O}(\delta^{-2}), \text{ with } \delta = \frac{1}{2} - \left(\frac{\mathsf{NL}(F)}{2^n}\right).$$

For Learning Parity with Noise solvers, correlation attacks based on XL, or other attacks of this kind, a similar bias has to be observed. The smaller is δ, the more distant is the algebraic system from a linear one, which decreases the efficiency of these attacks. When combinations of vectors are required to observe a bias, the higher is the resiliency, the higher is the attack complexity. In our following analyses, we adopt a conservative approach to thwart this variety of attacks: we assume that guaranteeing both $\delta^{-2} \geq 2^\lambda$ and a resiliency of $\lambda - 1$ avoids any attack of this kind with time or data complexity of less that 2^λ operations.

Note that in the context of Goldreich's PRG only the resiliency is studied. The underlying principle is, as the output is bounded (polynomial) and as the subsets are well distributed, the probability of repetitively finding subsets of the key-stream bits whose sum gives an unbalanced function is low, with enough resilience. In this context the nonlinearity is not studied, as any bias is considered as giving a polynomial attack.

Guess-and-determine Strategies. As shown in [DLR16] guess-and-determine attacks apply on FPs. Thus, we consider this class of attacks relatively to IFPs. The principle of the guess-and-determine attack consists in guessing ℓ key bits in order to target simpler functions, obtaining a system of equations easier to solve or with a distribution easier to distinguish. In our context it can be less costly for an attacker to consider the 2^ℓ possible systems given by fixing the value of ℓ variables than attacking the initial system of equation given by the key-stream. Hence, both kinds of attacks presented before can be generalized with guess-and-determine. We explain the principle relatively to the algebraic attack: the attacker selects ℓ variables and gives a value of its guesses, it simplifies the algebraic system. Then, the attacker considers all equations such that the descendant function has algebraic immunity at most k, and generates the corresponding degree k algebraic system. Once linearized, the attacker solves the system, if it is not consistent, then another guess is tried. As one of the 2^ℓ values of the guess is the correct one, the attack will succeed. Similarly for the other attacks, once the value of the guess is fixed, the attack is mounted on the new system relatively to a specific value of a parameter (the value of e and d for the fast algebraic attack, the value of δ, or the value of the resiliency).

A bound on the complexity of these attacks can be derived from the complexity of the attack without guess-and-determine. For the time complexity, it corresponds to multiplying by 2^ℓ the complexity of the attack using the parameter of value k on a system with $N - \ell$ variables. For the data complexity, the probability of getting a function with parameter k is important, the whole complexity can then be bounded by the inverse of this probability multiplied by the complexity of the attack using the parameter of value k on a system with $N - \ell$ variables. To determine this probability, it requires to determine the parameters relatively to the Boolean criteria of all descendant functions of F up to $\ell \leq \lambda$ variables. Some descendants may have extreme parameters (called recurrent criteria in [MJSC16]), but very low probability of appearing. Then for attacks with guess-and-determine, it is important to investigate both time and data complexities.

Other Attacks. Besides the previous attacks that will be taken into account quantitatively when selecting concrete instances, we also investigated other cryptanalyses, so we develop some explanations on those which are known to apply on filter permutators [MJSC16].

First, weak key attacks can be considered: if the Hamming weight of the key is extreme the input of F is far away from the uniform distribution. The probability of this weight to be extreme is very low due to the register extension, and as explained before the whitening avoids simple attacks using the unusual distribution of F's inputs. Restricting our instances to keys of Hamming weight $N/2$ handles these attacks.

Second, higher-order correlation attacks [Cou03b] consist in approximating the filtering function by a function of degree $d > 1$ and to solve the approximated algebraic system of degree d with a Grobner basis algorithm such as F4 [Fau99]. The attack could be efficient if the function was very close to a degree d function (which corresponds to a small nonlinearity of order d), and if d was low enough as one part of the attack consists in solving a degree d system. This attack can easily be combined with guess-and-determine techniques, but up to now for the filtering functions we tried, the complexity of this attack is always superior to the one considered for fast algebraic attacks or for correlation-like attacks.

Eventually, restricted input attacks [CMR17] using the behavior of F on a restricted part of its input are handled by the register size and the whitening. Since the input of F is not restricted to a subset of $\mathbb{F}_2^n$, but to the whole set, it seems unrealistic to adapt this attacks in this context. It would require to combine equations to obtain a set of equations corresponding with high probability to a known small subset of $\mathbb{F}_2^n$. Moreover the function should also have some flaws relatively to this particular subset, which we leave as a scope for further investigations.

4.2 Estimating the Attacks Complexity.

Based on the previous parts of this section, relatively to a Boolean function F and the register size N, we can estimate the security of IFPs by computing the parameters of each descendant up to λ variables, and then combining it with the probability or reaching descendants with this parameter when some guesses are made. For sake of clarity, we focus on the example of algebraic attack, on a simplified version where an upper bound of the $\mathcal{D}\mathsf{AN}$ is considered, in this simpler case the unique parameter to determine is the algebraic immunity.

First, we describe the principle of the algorithm used to determine the complexity of an attack relatively to a parameter. Then, to illustrate it, we give the algorithm relatively to the attack based on the algebraic immunity. Finally, we explain different variations of the security estimation, giving less costly algorithms at the cost of under estimating the attack complexities.

General Principle. The principle of the complexity estimator is to compute the profile of F and all its descendants relatively to a criterion. Then, based on this profile, the probability of getting functions with a fixed parameter when L values are guessed can be computed. It enables to derive an upper bound on the complexity of the attack performed on the whole system, considering the number of guesses made by the attacker.

More precisely the principle, applied on a simplified example of algebraic attack, is the following:

1. From F and λ, the profile of the function relatively to algebraic immunity is computed. The profile corresponds to the probability of getting a descendant of F with algebraic immunity less than or equal to k ($0 \leq k \leq \mathsf{AI}(F)$) by fixing ℓ bits of F inputs. The probability is taken over all choices of ℓ over n variables ($0 \leq \ell \leq \lambda$) and over the 2^ℓ possible values taken by these variables.
   To compute the profile, the probability of getting each descendant is computed iteratively, from step 0 to λ. Step 0 corresponds to the function F with probability 1, the profile for 0 guess gives a probability of 0 for $k < \mathsf{AI}(F)$ and 1 for $k \geq \mathsf{AI}(F)$. Then, from step ℓ to step $\ell+1$, for each descendant of step ℓ and its probability, all descendants obtained by fixing one of its variables (to 0 and to 1) are computed, together with their probability. It gives then all descendants of step $\ell+1$, the algebraic immunity of each one is computed, and the profile for ℓ guesses at value k is the sum of the probabilities of all these descendants with algebraic immunity less than or equal to k.
2. From the profile and N, for each L with $0 \leq L \leq \lambda$, and for each possible value k of the algebraic immunity ($0 \leq k \leq \mathsf{AI}(F)$), we compute the time and data complexity of the attack targeting functions with algebraic immunity less than or equal to k:
   - The time complexity is then 2^L multiplied by the time complexity of an algebraic attack with algebraic immunity equal to k on a system in $N-L$ variables.
   - The data complexity depends on the probability of obtaining an equation with such a parameter of algebraic immunity. This probability depends on the profile and on N. It corresponds to:
   $$P = \sum_{\ell=0}^{L} P_{L=\ell} \cdot P_{(\mathsf{AI} \leq k)\,|\ell},$$
   where $P_{L=\ell} = \frac{\binom{L}{\ell}\binom{N-L}{n-\ell}}{\binom{N}{n}}$ is the probability that ℓ over the L guesses of the adversary are in the n input's variables of F. $P_{(\mathsf{AI} \leq k)\,|\ell}$ is the probability that the function has algebraic immunity less than or equal to k conditioned on the number of variables fixed in F to get this function. This probability is what the profile gives. The data complexity is finally P^{-1} multiplied by the data complexity of an algebraic attack with algebraic immunity equal to k on a system in $N-L$ variables.
3. For each pair (L, k), we finally determine the maximum between the time and data complexity, then the minimum over these complexities gives the final complexity of the attack.

The principle of considering the properties of all descendants corresponds to consider average properties of the system given by the keystream. It is justified by the generation of the system from the PRG, which ensures that the subsets, permutations and whitenings are pseudorandomly distributed.

Algorithms for the Algebraic Attack. For completeness, we give the algorithms in pseudo-code relatively to a simplified algebraic attack. More precisely Algorithm 1 corresponds to the first item (determination of the algebraic profile) and Algorithm 2 corresponds to the second and third items (computation of the best attack complexity).

Remark 2. In Algorithm 2 the auxiliary function AIdata(k,N) gives the minimal data complexity of an algebraic attack considering an AI of k and functions in N variables. This function uses the maximal possible value of $\mathcal{D}\mathsf{AN}$, a tighter estimation can be obtained by considering the $\mathcal{D}\mathsf{AN}$ of the descendant functions, or at least a tighter bound. Nevertheless, it requires to compute the profile of the algebraic immunity jointly with the $\mathcal{D}\mathsf{AN}$. Instead of considering the probability of a descendant to have a fixed AI, it consists in separating this probability among the different possible values of the $\mathcal{D}\mathsf{AN}$ of these functions.

We explain more these trade-offs between the precision of the estimators and the complexity of the algorithm in the next part.

```
Input: $F$, $\lambda$
Output: Algebraic immunity profile of $F$ up to $\lambda$.
profileAI ←[]; listFunctions ← [$F$]; listProbaFunctions ← [1];
listAI ← [0 for i in [0, maxAI] ]; //maxAI is a global parameter upper bounding the algebraic
  immunity of any descendant of $F$.
//The algebraic immunity is computed from the function AI(), and the probabilities are updated
listAI[AI($F$)]+=1;
for i in [1, maxAI] do
    listAI[i]+=listAI[i-1];
end
profileAI.append(listAI);
//Computation of the profile up to $\lambda$ guesses.
for $\ell$ in [1, $\lambda$] do
    newListFunctions ←[]; newListProbaFunctions ←[];
    listAI←[0 for i in [0, maxAI] ];
    //Computation of the new functions, probabilities and AI for each element of listFunctions.
    for index in [0, listFunctions.length()-1] do
        parentFunction ← listFunctions[index]; parentProba ← listProbaFunctions[index];
        //All descendant obtained by fixing a variable of parentFunction to 0 or to 1 are considered.
        for i in [1, parentFunction.nbvar()] do
            for b in [0, 1] do
                descendantFunction ← $parentFunction_{\{i\},b}$;
                // using the notation of bit fixing descendant.
                descendantProba← parentProba / (2 parentFunction.nbvar());
                if descendantFunction in newListFunctions then
                    ind←newListFunctions.index(descendantFunction);
                    newListProbaFunctions[ind]+= descendantProba ;
                else
                    newListFunctions.append(descendantFunction);
                      newListProbaFunctions.append(descendantProba);
                end
                listAI[AI(descendantFunction)]+=descendantProba;
            end
        end
    end
    listFunctions ← newListFunctions;
    listProbaFunctions ← newListProbaFunctions;
    for i in [1, maxAI] do
        listAI[i]+=listAI[i-1];
    end
    profileAI.append(listAI);
end
return (profileAI)
```

Algorithm 1: Determination of the algebraic immunity profile.

```
**Input:** The algebraic immunity profile of $F$: profileAI, $F$,$N$, $\lambda$
**Output:** Complexity of the best algebraic attack (ignoring the $\mathcal{D}$AN).
bestCpx=maxCpx //maxCpx is a global parameter bounding the maximal complexity considered
for $L$ in [0, $\lambda$] do
    listAI[i]+=listAI[i-1];
    $k \leftarrow 0$;
    // maxAI is a global parameter upper bounding the algebraic immunity of any descendant. We
      consider the auxiliary functions AItime($k$,$N$) and AIdata($k$,$N$) giving respectively the minimum
      time and data complexity of the algebraic attack with parameter $k$ on a system in $N$ variables.
    while $k \leq$ maxAI do
        //Compute the time complexity
        timeCpx $\leftarrow$ AItime($k$,$N - L$);
        if timeCpx $\geq$ bestCpx then
            $k \leftarrow$ maxAI $+1$;
        else
            //Compute the data complexity
            proba $\leftarrow 0$;
            for $\ell$ in [0,$L$] do
                probaLell $\leftarrow \binom{L}{\ell}\binom{N-L}{F.nbvar()-\ell}/\binom{N}{F.nbvar()-\ell}$;
                proba += probaLell $\cdot$ profileAI[$\ell$][$k$];
            end
            dataCpx $\leftarrow proba^{-1}$ AIdata($k$,$N - L$);
            cpx $\leftarrow$ max(timeCpx,dataCpx);
            if cpx $\leq$ bestCpx then
                bestCpx $\leftarrow$ cpx;
            end
        end
        $k$ += 1;
    end
end
return (bestCpx)
```

Algorithm 2: Computation of the complexity of the best algebraic attack (simplified).

Potential Modifications The advantage of this methodology is to apply on any filtering function F, and any register size N, giving a general framework to determine the security of IFPs instances. This general algorithm being exhaustive, it has a high time and storage complexity. Indeed, note that the number of descendants of a function is exponential. The algorithm can be modified in order to be more efficiently evaluated, but sometimes at the cost of underestimating the cost of the attacks:

– A first modification, which does not underestimates the cost of the attacks, consists in finding the descendants which are equivalent. That is, the ones which have exactly the same parameters for each criterion and that give the same descendants with identical probabilities. When such equivalent descendants are found, which can be handled through the representation of the function, the number of descendant at step ℓ can be less than the initial bound of $2^\ell\binom{n}{\ell}$.
– A second modification, underestimating the cost of the attacks, consists in replacing each value of the parameter (which can take in some cases numerous values) by the nearest one among those which are more favorable to the attacker in a shorter list, and summing the probabilities corresponding to each such approximation.
– A third modification, also underestimating the cost of the attacks, can be achieved by not considering all descendants but only descendants which have worse parameters. It is possible when, for each number of guesses considered, for each criterion, the profile of a function is worse than the profile of another one. Then the probability of the function with better profiles can be added to the probability of the function with worse profiles. In other words, a (stronger) function can be neglected and its probability added to another one, if the probability of its descendants to reach a particular weak value of parameter is always inferior than the corresponding probability for the descendants of the other (weaker) function.

5 Parameters of Direct Sums of Monomials and XOR-Threshold Functions.

In this section and in the following we prove the relevant parameters relatively to Boolean cryptographic criteria of two family of functions: DSM and XOR-THRfunctions. The results relative to balancedness, resiliency and nonlinearity and the first part of the algebraic properties are proven in this section. The second part of the algebraic properties require new techniques based on the Partitioned Algebraic Normal Form coefficients, then we give them in the next section where the new tool is developed.

In sub-section 5.1 we give the parameters and proofs related to the resiliency, nonlinearity, algebraic immunity and fast algebraic immunity of DSM functions. In sub-section 5.2 we give the parameters and proofs of threshold functions. Combining it with results on direct sums (Lemma 1) gives the resilience and nonlinearity of XOR-THR.

5.1 Direct Sum of Monomials.

First we recall some properties on direct sums (*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:* $\mathsf{FAI}(F) \geq \max(\mathsf{FAI}(f), \mathsf{FAI}(g))$.

Resiliency and Nonlinearity The precedent 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* $= [m_1, \ldots, m_k]$, *its resiliency is:*

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

Proof. A monomial function of degree greater than 1 has resiliency -1 as such function is unbalanced, a monomial function of degree 1 has resiliency 0. Then, applying the first item of Lemma 1 recursively (adding one by one the monomial functions) gives the result. $\square$

The nonlinearity can be recursively computed from Lemma 1, but we give a simpler expression based on the direct sum vector.

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* $\mathbf{m}_f = [m_1, \ldots, m_k]$, *its nonlinearity is:*

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

Proof. First we use the expression of the nonlinearity using the Walsh transform recalled in Definition 7:

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

Then, recall that the Walsh transform of a direct sum is the product of the Walsh transforms of its components:

Let consider the Walsh transform of F, defined for all $u \in \mathbb{F}_2^{n+m}$. For each element u we use the following partition: we denote a the first n bits and b the last m bits, then a and b are uniquely defined. Denoting u as (a, b), enables to derive the following equalities:

$$\begin{aligned} W_F(u) &= \sum_{x \in \mathbb{F}_2^{n+m}} (-1)^{F(x)+u\cdot x} = \sum_{(y,z) \in \mathbb{F}_2^n \times \mathbb{F}_2^m} (-1)^{h(y)+g(z)+a\cdot y+b\cdot z} \\ &= \sum_{y \in \mathbb{F}_2^n} \left(\sum_{z \in \mathbb{F}_2^m} (-1)^{h(y)+a\cdot y+g(z)+b\cdot z} \right) = \sum_{y \in \mathbb{F}_2^n} (-1)^{h(y)+a\cdot y} \left(W_g(b)\right) \\ &= W_h(a)\ W_g(b). \end{aligned}$$

Therefore $\max_{a \in \mathbb{F}_2^n} |W_f(a)|$ can be determined by computing the maximum of the Walsh transform of all monomial functions indicated by the direct sum vector. For a degree 1 function (and also 0) h in m variables $\max_{a \in \mathbb{F}_2^m} |W_h(a)| = 2^m$. For a degree d (with $d > 1$) monomial function g, $\max_{a \in \mathbb{F}_2^d} |W_g(a)| = 2^d - 2$. Multiplying the maximum of the Walsh transform of all the monomial functions composing f gives the final result.

$\square$

Algebraic Immunity More material is necessary in order to determine the exact algebraic immunity of a direct sum of monomials. To get this result we use two results of others works, the algebraic immunity of a triangular function ([MJSC16] Lemma 6), and a property on the algebraic immunity of linked functions ([CMR17]).

Lemma 4 (Algebraic Immunity of Triangular Functions (adapted from [MJSC16], Lemma 6)). *Let k be a non null positive integer and let T_k be the k-th triangular function, then $\mathsf{AI}(T_k) = k$.*

Lemma 5 ([CMR17] Proposition 11). *Let $f(x_1, x_2, x_3, \dots, x_n)$ be a Boolean function in n variables such that there exist two variables (x_1 and x_2 without loss of generality) satisfying:*

$$\forall x \in \mathbb{F}_2^{n-2}\ f(0,0,x) = f(0,1,x) = f(1,0,x)$$

Let $F(X, x_3, \dots, x_n)$ be the Boolean function in $n-1$ variables defined by :

$$\forall x \in \mathbb{F}_2^{n-2}\ F(1,x) = f(1,1,x) \text{ and } F(0,x) = f(0,0,x)$$

If $\mathsf{AI}(f) \leq d$ then $\mathsf{AI}(F) \leq d$.

Using these two lemmata we can determine the exact algebraic immunity of any 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:*

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

Proof. First, we prove the inequality:

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

We know that the algebraic immunity of a direct sum of functions is bounded above by the sum of the algebraic immunities of the functions, that the algebraic immunity of any function is bounded above by its algebraic degree, and that the algebraic immunity of a monomial equals 1 (see Corollary 1). We fix d, and express f as a direct sum of two functions f_1 and f_2, with direct sum vectors:

$$\mathbf{m}_{f_1} = [m_1, \dots, m_d], \text{ and } \mathbf{m}_{f_2} = [0, \dots, 0, m_{d+1}, \dots, m_k].$$

From $\mathbf{m}_{f_1}$, we have $\mathsf{deg}(f_1) \leq d$ and we deduce the inequality.
Note that this inequality shows, for the case of direct sums of monomials, that the algebraic immunity is upper bounded both by the number of monomials (case $d = 0$) and the degree of f (case $d = k$).

Then, we prove the inequality in the other sense:

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

Let us denote by e the integer between 0 and k giving the minimal value of the sum, we take e as the smallest element if multiple integers lead to the minimal value.

So, for all integers j such that $1 \leq j \leq e$:

$$e - j + \sum_{i=e-j+1}^{k} m_i \geq e + \sum_{i=e+1}^{k} m_i,$$

which is equivalent to:

$$\sum_{i=e-j+1}^{e} m_i \geq j.$$

This inequality holds for all j such that $1 \leq j \leq e$; for $j = 1$ it guarantees that m_e is non null, for $j = 2$ it gives that $m_e + m_{e-1} \geq 2$, and so forth until $m_1 + \cdots + m_e \geq e$. Therefore it guarantees that we can repetitively apply Lemma 5 on f to obtain a function with direct sum vector having all m_i non null for $1 \leq i \leq e$, which already gives $\mathsf{AI}(f) \geq e$ (as this function is the direct sum of T_e of algebraic immunity e by Lemma 4 and another function). A constructive way can consist in contracting all but one monomials of each degree less than or equal to e, beginning by the degree-e monomials. It would lead to a function F with direct sum vector:

$$\mathbf{m}_F = \left[\left(\sum_{i=1}^{e} m_i \right) - e + 1, \ldots, 1, m_{e+1}, \ldots, m_k \right].$$

Now, let us consider the monomials of degree higher than e; for all integers j such that $1 \leq j \leq k - e$:

$$e + j + \sum_{i=e+j+1}^{k} m_i \geq e + \sum_{i=e+1}^{k} m_i,$$

which is equivalent to:

$$j \geq \sum_{i=e+1}^{e+j} m_i.$$

This inequality holds for all j such that $1 \leq j \leq k - e$; for $j = 1$ it guarantees that $m_{e+1} \leq 1$, for $j = 2$ it gives that $m_{e+1} + m_{e+2} \leq 2$, so one and so forth until $m_{e+1} + \cdots + m_k \leq k - e$. Therefore it guarantees that there are no more monomials than positions between e and any position of degree higher than e. So we can repetitively apply Lemma 5 on f to obtain a function with direct sum vector containing $\sum_{i=e+1}^{k} m_i$ consecutive 1 from m_{e+1} and zeros for higher positions (the zeros are then deleted for a correct representation of the direct sum vector). A constructive way can consist in contracting each monomial to the first empty position of the vector, from m_{e+1} to m_k. It would lead to a function F with direct sum vector of length $e + \sum_{i=e+1}^{k} m_i$:

$$\mathbf{m}_F = [m_1, \ldots, m_e, 1, \ldots, 1].$$

Together with the reasoning on the lower positions of the vector, this result shows that the AI of f can be linked through the repetitive use of Lemma 5 to the AI of a function F with direct sum vector of length $e + \sum_{i=e+1}^{k} m_i$:

$$\mathbf{m}_F = [m_1, \ldots, m_e, 1, \ldots, 1], \text{ such that } m_i > 0 \; \forall i \in [1, e].$$

Note that this function can be expressed as the direct sum of T_k and another function, then using Lemma 1 and Lemma 4 $\mathsf{AI}(F) \geq k$. As $\mathsf{AI}(F) = e + \sum_{i=e+1}^{k} m_i$, Lemma 5 gives that $\mathsf{AI}(f) > e - 1 + \sum_{i=e+1}^{k} m_i$, proving the second inequality, and finishing the proof.

□

Note that to estimate accurately the time complexity of the algebraic attack (mounted on f), it is better to additively know the number of annihilators of f or $f + 1$ of degree $\mathsf{AI}(f)$. As determining this number requires additional concepts on algebraic immunity and direct sums, we defer its study to Section 6.

Fast Algebraic Immunity Concerning the fast algebraic immunity criterion, its definition leads to the bound $\mathsf{FAI}(f) \geq \mathsf{AI}(f) + 1$ for any f. In the case of direct sum of monomials, we can show that the FAI is reaching this bound for some functions.

Lemma 6 (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 $\mathbf{m}_f = [m_1, \ldots, m_k]$ such that $\mathsf{AI}(f) = \mathsf{deg}(f)$, and $\mathsf{AI}(f) > 1$, its fast algebraic immunity is:*

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

Proof. We first consider the case where $m_k = 1$. As $\mathsf{AI}(f) = \mathsf{deg}(f) = k$ by Theorem 1, we have $m_{k-1} \geq 1$. Let us denote by x_1 one of the variables of the monomial of degree k, then we consider the degree of the product of f and $1 + x_1$, this function has degree k (as the only monomial of degree k of f is canceled, and the m_{k-1} monomials of degree $k - 1$ do not contain x_1). Then, by definition of the FAI (see Definition 9), and since $\mathsf{AI}(f) = k$, this gives:

$$\mathsf{FAI}(f) \leq \min(2\mathsf{AI}(f), \max(\mathsf{AI}(f) + 1, 3)).$$

As the Lemma is restricted to the case $\mathsf{AI}(f) > 1$, this gives $\mathsf{FAI}(f) \leq \mathsf{AI}(f) + 1$, enabling to conclude $\mathsf{FAI}(f) = \mathsf{AI}(f) + 1$.

We consider the case $m_k > 1$. Multiplying f by a linear function g we study the degree of fg. As f is a DSM, denoting without loss of generality by x_1 a variable present in the ANF of g, x_1 appears in either zero or one of the higher degree monomials of f. If it does not appear then fg produces m_k monomials of degree $k + 1$ containing x_1, as $k > 1$, all these monomials are different, so $\mathsf{deg}(fg) = k + 1$. If x_1 appears in a degree k monomial, the same reasoning applies for the $m_k - 1$ others, leading to $\mathsf{deg}(fg) = k + 1$. This gives:

$$\mathsf{FAI}(f) \leq \min(2\mathsf{AI}(f), \max(\mathsf{AI}(f) + 2, 3)).$$

And therefore $\mathsf{FAI}(f) \leq \mathsf{AI} + 2$. Then, as for all non null function g such that $\mathsf{deg}(g) < \mathsf{AI}(f)$ we have $\mathsf{deg}(fg) \geq \mathsf{AI}(f)$ (property of the AI), any nonlinear function g leads to consider a maximum greater than or equal to $\mathsf{AI}(f) + 2$ leading to an equal or higher upper bound. It enables to conclude: $\mathsf{FAI}(f) = \mathsf{AI}(f) + 2$.

□

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.

5.2 Threshold Functions.

In order to obtain the parameters of XOR-THR functions we first need to determine the one of threshold functions, and then to use the properties of direct sum constructions.

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

We begin by giving a basic property between $\mathsf{T}_{d,n}$ and $\mathsf{T}_{n-d+1,n}$, which will simplify the number of cases to consider in some proofs.

Property 1. Let $n \in \mathbb{N}^*$ and $d \in [0, n+1]$, for all $x \in \mathbb{F}_2^n$ let $1+x$ denote the element $(1+x_1, \dots, 1+x_n) \in \mathbb{F}_2^n$, then the following relation holds for $\mathsf{T}_{d,n}$ and $\mathsf{T}_{n-d+1,n}$:

$$\forall x \in \mathbb{F}_2^n, \quad 1 + \mathsf{T}_{d,n}(1+x) = \mathsf{T}_{n-d+1,n}(x).$$

Proof. We use the simplified value vector formalization (see Definition 16) to show this result. The simplified value vector of $\mathsf{T}_{d,n}$ is a vector of length $n+1$ such that $w_k = 0$ for $k \in [0, d-1]$ and $w_k = 1$ for $k \in [d, n]$. For all element $x \in \mathbb{F}_2^n$, we have $\mathsf{w_H}(x+1) = \mathsf{w_H}(1) - \mathsf{w_H}(x) = n - \mathsf{w_H}(x)$. So denoting w'_k the coefficients of the simplified value vector of $\mathsf{T}_{d,n}(1+x)$ we get: $w'_k = w_{n-k}$ for all $k \in [0, n]$. It gives a vector symmetric to the first simplified value vector, *i.e.* with the elements from 0 to $n-d$ being 1 and from $n-d+1$ to n being 0.

For all $x \in \mathbb{F}_2^n$, $1+\mathsf{T}_{d,n}(1+x) = \overline{\mathsf{T}_{d,n}(1+x)}$, its complement to 1. Then, denoting $w"_k$ the coefficients of the simplified value vector of $1 + \mathsf{T}_{d,n}(1+x)$ we get: $w"_k = \overline{w_{n-k}}$ for all $k \in [0, n]$. It gives a vector which is the complement to the all 1 vector to the precedent simplified value vector, *i.e.* with the elements from 0 to $n-d$ being 0 and from $n-d+1$ to n being 1. This simplified value vector corresponds to the definition of $\mathsf{T}_{n-d+1,n}$.

□

Resiliency Now, focusing on the main criteria of threshold functions, we first give the resiliency of such functions.

Lemma 7 (Resiliency of Threshold Functions). *Let f be the threshold function $\mathsf{T}_{d,n}$,*

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

Proof. We first show that only the functions such that $n = 2d-1$ can be balanced, using the simplified value vector. The Hamming weight of a symmetric function is equal to $\sum_{i=0}^{n} w_i \binom{n}{i}$, for a threshold function the d first coefficients are equal to 0 and the $n+1-d$ others are equal to 1, then:

$$\mathsf{w_H}(\mathsf{T}_{d,n}) = \sum_{i=d}^{n} \binom{n}{i} = \sum_{i=0}^{n-d} \binom{n}{i}.$$

This sum needs to be equal to 2^{n-1} to be balanced, which imposes $d = (n+1)/2$. In all other cases $\mathsf{T}_{d,n}$ is not balanced so with resiliency order -1.

We finish by proving that these balanced function (which are the majority functions) are not 1-resilient. As the family of XOR-MAJ functions is bit-fixing stable, fixing one variable of a majority function gives a threshold function in $n-1$ variables, which cannot be balanced based on the previous part. Then majority functions are 0 resilient only (note that rigorously we cannot consider bit fixing on MAJ_1, which cannot be more than 0 resilient due to its number of variables).

$\square$

Nonlinearity We follow by studying the nonlinearity of threshold functions. It would be possible to adapt the approach presented in [DMS06] for calculating the nonlinearity of the majority functions $\mathsf{T}_{\frac{n+1}{2},n}$ (n odd) and $\mathsf{T}_{\frac{n}{2}+1,n}$ (n even). It would consist in expressing the Walsh transform by means of Krawtchouk polynomials and using relations on these polynomials to obtain the maximal absolute value. But the resulting proof, that we wrote, needs to consider several particular cases, and is 5 page long (preliminaries on Krawtchouk polynomials excluded). There is a better way to obtain the nonlinearity by using a very efficient representation of Boolean functions called the numerical normal form (see *e.g.* [Car10]).

Definition 21 (Numerical Normal Form). *For every n-variable Boolean function f, we call* Numerical Normal Form *(NNF) of f the unique polynomial $N_f(x)=\sum_{I\subseteq[n]}\lambda_I x^I \in \mathbb{Z}[x_1,\dots,x_n]/(x_1^2-x_1,\dots,x_n^2-x_n)$, where x^I stands for $\prod_{i\in I} x_i$, such that $f(x)=N_f(x)$ for every $x\in\mathbb{F}_2^n$.*

Note that the ANF (see Definition 2) is simply the modulo 2 version of the NNF, which coefficients give some information more directly. First we recall some properties of the NNF, then using them we prove a lemma linking the Walsh transform of threshold function to already studied functions, and we conclude by giving the exact nonlinearity of all threshold function.

Proposition 1 (Properties of NNF and Walsh transform, adapted from [Car10]).

- *Let f be any Boolean function in n variables and any $I\subseteq[n]$:*

$$\lambda_I = (-1)^{|I|}\sum_{x\in\mathbb{F}_2^n;\ \mathsf{supp}(x)\subseteq I}(-1)^{w_H(x)}f(x),$$

  where $\mathsf{supp}(x)$ denotes the support $\{i\in[n]\mid x_i\neq 0\}$ of vector x, this sum being calculated in $\mathbb{Z}$.
- *Let f be the indicator function $1_{E_{n,r}}$ of the set $E_{n,r}$ of all vectors of Hamming weight r and length n, then for any $I\subseteq[n]$:*

$$\lambda_I = (-1)^{|I|}\sum_{\substack{x\in\mathbb{F}_2^n;\ w_H(x)=r\\ \mathsf{supp}(x)\subseteq I}}(-1)^{w_H(x)} = (-1)^{|I|-r}\binom{|I|}{r}.$$

- *Let f be a Boolean function in n variables, if $u\neq 0$ then:*

$$W_f(u) = 2(-1)^{\mathsf{w_H}(u)+1}\sum_{I\subseteq[n];\ \mathsf{supp}(u)\subseteq I}2^{n-|I|}\lambda_I.$$

Lemma 8. *The maximum absolute value of the Walsh transform of $1_{E_{n,d}}$ equals the maximum absolute value of the Walsh transform of $\mathsf{T}_{d+1,n+1}$ at nonzero entries.*

Proof. Using the first and second items of Proposition 1, since $\mathsf{T}_{d,n}=\sum_{k=d}^{n}1_{E_{n,k}}$ (this sum being calculated in $\mathbb{Z}$), the coefficients of x^I in the NNF of $\mathsf{T}_{d,n}(x)$ equals:

$$(-1)^{|I|}\sum_{k=d}^{|I|}(-1)^k\binom{|I|}{k} = (-1)^{|I|-1}\sum_{k=0}^{d-1}(-1)^k\binom{|I|}{k} = (-1)^{|I|-d}\binom{|I|-1}{d-1},$$

where the first equality comes from $\sum_{k=0}^{|I|}(-1)^k\binom{|I|}{k} = 0$, and the last equality is obtained by induction on d using Pascal's identity.

According to Lucas' theorem [MS78, page 404], the coefficient of x^I in the ANF of $\mathsf{T}_{d,n}$ equals 1 (*i.e.* λ_I is odd) if and only if the binary expansion of $d-1$ is covered by that of $|I|-1$, and the algebraic degree of $\mathsf{T}_{d,n}$ equals then $k+1$ where k is the largest number smaller than n whose binary expansion covers that of $d-1$, that is, where $k-d+1$ is the largest number smaller than $n-d+1$ whose binary expansion is disjoint from that of $d-1$.

Moreover, using the third item of Proposition 1, we deduce that the Walsh transform of the threshold function satisfies (for $u \neq 0$):

$$W_{\mathsf{T}_{d,n}}(u) = 2(-1)^{w_H(u)+1} \sum_{\substack{I\subseteq[n]\\ \mathsf{supp}(u)\subseteq I}} 2^{n-|I|}(-1)^{|I|-d}\binom{|I|-1}{d-1}.$$

Using the second and third item of Proposition 1, we obtain for $1_{E_{n,r}}$:

$$W_{1_{E_{n,r}}}(u) = 2(-1)^{\mathsf{w_H}(u)+1} \sum_{I\subseteq[n];\ \mathsf{supp}(u)\subseteq I} 2^{n-|I|}(-1)^{|I|-r}\binom{|I|}{r}.$$

Therefore, the Walsh transform of function $1_{E_{n,d}}$ at $u \in \mathbb{F}_2^n$ equals, for every u, the opposite of the Walsh transform of function $\mathsf{T}_{d+1,n+1}$ at $(u,1)$ (the concatenation of u and the length one vector (1)). Since these two functions are symmetric, this concludes the proof.

□

We can finally give the exact nonlinearity of all threshold functions through the following theorem:

Theorem 2 (Nonlinearity of Threshold Functions). *Let n be a non null positive integer, the threshold function $\mathsf{T}_{d,n}$ has the following nonlinearity:*

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

Proof. The first case $d = (n+1)/2$ is possible only for odd n, it corresponds to the majority function, and is proved in [DMS06]. For the case $d > (n+1)/2$, we have for every $u \neq 0$:

$$|W_{1_{E_{n-1,d-1}}}(u)| = 2\,|\sum_{x\in E_{n-1,d-1}}(-1)^{u\cdot x}| \leq 2\,\mathsf{w_H}(1_{E_{n-1,d-1}}) = 2\binom{n-1}{d-1},$$

where the first inequality comes from the basic properties of Hadamard Fourier and Walsh transforms. For the null vector:

$$|W_{\mathsf{T}_{d,n}}(0)| = 2^n - 2\sum_{i=d}^{n}\binom{n}{i} = \sum_{i=n-d+1}^{d-1}\binom{n}{i}.$$

Hence, using Pascal's identity, $|W_{\mathsf{T}_{d,n}}|$ takes its maximum at the 0 input and this completes the proof in this case.

Finally, in the case $d < (n+1)/2$, we can use Property 1: $\forall x \in \mathbb{F}_2^n, \quad 1 + \mathsf{T}_{d,n}(x+1) = \mathsf{T}_{n-d+1,n}(x)$. As the application $x \mapsto x+1$ is an affine isomorphism of $\mathbb{F}_2^n$, and that adding the constant function 1 does not change the nonlinearity, the nonlinearity criterion being affine invariant it gives: $\mathsf{NL}(\mathsf{T}_{d,n}) = \mathsf{NL}(\mathsf{T}_{n-d+1,n})$, with $n - d + 1 > \frac{n+1}{2}$, and we are brought back to the previous case. This completes the whole proof. □

5.3 Algebraic Immunity, $\mathcal{D}\mathsf{AN}$, and Fast Algebraic Immunity.

We then investigate the algebraic immunity of threshold functions. Relatively to Boolean function 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]). 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. We decide to use the following proof strategy: we consider as known the algebraic immunity of the majority function in an odd number of variables, $\mathsf{AI}(\mathsf{T}_{(n+1)/2,n}) = \frac{n+1}{2}$, and we use the connections between various threshold functions to get the algebraic immunity of all functions of the family. A method for determining the algebraic immunity of threshold functions consists in using the following result on the algebraic immunity of restrictions of functions.

Lemma 9. *Fixing $\ell \in [0, n]$ variables of an n-variable Boolean function decreases it algebraic immunity by at most ℓ.*

Proof. If the restriction of a function f obtained by fixing x_i to a_i for any $i \in I \subseteq \{1, \ldots, n\}$ has a nonzero annihilator g of some algebraic degree r, then f has for nonzero annihilator the function equal to g when $x_i = a_i, \forall i \in I$ and equal to 0 otherwise, whose ANF equals $g(x)(1 + \prod_{i \in I}(x_i + a_i + 1))$, and whose algebraic degree equals then $r + \ell$, where $\ell = |I|$. This can be applied to f and to $f + 1$, and this proves that the algebraic immunity of the restriction is at least $AI(f) - \ell$. □

Remark 3. Note that the result of Lemma 9 also corresponds to Proposition 1 in [DGM04] written differently, and its principle is used in proofs dealing with recurrent algebraic immunity ([MJSC16]) or other bit-fixing relations (see [DGM05] Proposition 1 and [AL16]).

This result can be used in conjunction with the fact that, if $d \leq \frac{n+1}{2}$, then $\mathsf{T}_{d,n}$ can then be obtained by fixing $n - 2d + 1$ input bits to 1 in the $(2n - 2d + 1)$-variable majority function (indeed we sen in Remark 1) that fixing a variable to 1 in $\mathsf{T}_{d,n}$ gives $\mathsf{T}_{d-1,n-1}$). But we can also give a direct proof of the algebraic immunity of threshold functions (and also determine their annihilators of minimum algebraic degrees):

Lemma 10 (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:*

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

Proof. Applying the transformation $x \mapsto x + 1_n$, where 1_n is the all-1 vector of length n, changes $\mathsf{T}_{d,n}$ into the indicator of the set of vectors of Hamming weight at most $n - d$; the relations relating the expressions of the coefficients of the ANF $\sum_{I \subseteq \{1,\ldots,n\}} a_I x^I$ by means of the values of the function, namely, $a_I = \sum_{\mathsf{supp}(x) \subseteq I} f(x)$ and $f(x) = \sum_{I \subseteq \mathsf{supp}(x)} a_I$, show that the annihilators of this indicator are all the linear combinations over $\mathbb{F}_2$ of the monomials of degrees at least $n - d + 1$; hence, the annihilators of $\mathsf{T}_{d,n}$ are obtained from these latter linear combinations by the transformation $x \mapsto x + 1_n$. They can have every algebraic degree at least $n-d+1$. And the annihilators of $1+\mathsf{T}_{d,n}$ are similarly the linear combinations over $\mathbb{F}_2$ of the monomials of degrees at least d. They can have every algebraic degree at least d. Hence $\mathsf{AI}(\mathsf{T}_{d,n}) = \min(d, n-d+1)$. $\square$

We finally investigate the $\mathcal{D}\mathsf{AN}$ of threshold functions, and derive from it a bound on the fast algebraic immunity. Note that the $\mathcal{D}\mathsf{AN}$ can be easily deduced from the proof of the precedent lemma, we give the following step by step proof for completeness.

Lemma 11 (**$\mathcal{D}\mathsf{AN}$ of Threshold Functions**). *Let n be a non null positive integer, the threshold function $\mathsf{T}_{d,n}$ has the following $\mathcal{D}\mathsf{AN}$:*

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

Proof. First, we investigate $\mathcal{D}\mathsf{AN}(\mathsf{T}_{d,n})$ for $d \geq \frac{n+1}{2}$. According to Property 1 and using that the dimension of the annihilators of a fixed degree is invariant under an affine transformation, this is equivalent to considering the annihilators of degree d of $1+\mathsf{T}_{d,n}$ with $d \leq \frac{n+1}{2}$. Then we show that $\mathsf{T}_{d,n}$ with $d < \frac{n+1}{2}$ has no annihilator of degree d.

When $d \leq \frac{n+1}{2}$, the function $1+\mathsf{T}_{d,n}$ (of algebraic immunity d, according to Lemma 10) has simplified value vector $[1,\dots,1,0,\dots,0]$ where the first 0 corresponds to the Hamming weight d. All monomial functions of degree d are then annihilators since they vanish on the support of $1+\mathsf{T}_{d,n}$. This gives $\binom{n}{d}$ independent annihilators of degree $\mathsf{AI}(1+\mathsf{T}_{d,n})$ (which is the maximum value as we saw). It gives then $\mathcal{D}\mathsf{AN}(1+\mathsf{T}_{d,n}) = \binom{n}{d}$, and therefore noting d' as $n-d+1$, $\mathcal{D}\mathsf{AN}(\mathsf{T}_{d',n}) = \mathcal{D}\mathsf{AN}(\mathsf{T}_{n-d+1,n}) = \binom{n}{d} = \binom{n}{d'-1}$.

For the function $\mathsf{T}_{d,n}$ itself with $d < \frac{n+1}{2}$ we prove that $\mathcal{D}\mathsf{AN}(\mathsf{T}_{d,n}) = 0$ by showing that the unique function of degree less than $n-d+1$ annihilating $\mathsf{T}_{d,n}$ is the null function. To do so, let us consider the function f defined as:

$$\forall x \in \mathbb{F}_2^n, \quad f(x) = \mathsf{T}_{d,n}(x_1+1, x_2+1, \dots, x_n+1).$$

f is affine equivalent to $\mathsf{T}_{d,n}$, and $\mathcal{D}\mathsf{AN}(f) = \mathcal{D}\mathsf{AN}(\mathsf{T}_{d,n})$. Let g be an annihilator of f of degree less than $n-d+1$, the ANF (see Definition 2) of g is then given by:

$$g(x) = \sum_{\substack{I \subseteq [n] \\ |I| \leq n-d}} a_I x^I.$$

By construction, f has value 1 on all input of Hamming weight less than or equal to $n-d$ (as explained in the proof of Property 1 the simplified value vector of f is the symmetric of the one of $\mathsf{T}_{d,n}$, so with $n-d+1$ values 1 and then d values 0). It implies for g:

$$\forall x \in \mathbb{F}_2^n \mid \mathsf{w_H}(x) \leq n-d, \quad g(x) = 0.$$

Then, using the binary Mobius transform (*e.g.* [Car10] Proposition 1):

$$a_I = \sum_{\substack{x \in \mathbb{F}_2^n \\ \mathsf{supp}(x) \subseteq I}} g(x) = \sum_{\substack{x \in \mathbb{F}_2^n \\ \forall i \in [1,n], x_i = 0 \text{ if } i \notin I}} g(x).$$

So, for all I such that $|I| \leq n - d$, $a_I = 0$ as the sum is running on null terms only. Then, $g(x) = 0$ for all x, giving that f has no non null annihilator of degree less than $n - d + 1$ and the result applies to $\mathsf{T}_{d,n}$. Finally $\mathsf{AN}(\mathsf{T}_{d,n}) \geq n - d + 1$ (in fact it is an equality when $d \neq 0$ as f can be canceled by any monomial of degree $n - d + 1$) so $\mathcal{D}\mathsf{AN}(\mathsf{T}_{d,n}) = 0$, concluding the proof.

□

Corollary 2 (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:*

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

Proof. Note that, for every Boolean function f, we have:

$$\mathsf{FAI}(f) \geq \min(2\mathsf{AI}(f), 1 + \mathsf{AN}(f + 1)).$$

This bound comes from the definition of $\mathsf{FAI}(f)$ (see Definition 9), focusing more particularly on the degree of fg for all g such that $1 \leq \mathsf{deg}(g) < \mathsf{AI}(f)$). Let us denote $h = fg$, multiplying by $1 + f$ it gives: $(1 + f)h = 0$. Considering f non constant (otherwise $\mathsf{FAI}(f) = 0$ as $\mathsf{AI}(f) = 0$), as g is taken such that $\deg(g) < \mathsf{AI}(f)$, h is non null, therefore h is a non null annihilator of $1 + f$, and by definition $\deg(h) \geq \mathsf{AN}(f + 1)$.

As the fast algebraic immunity is invariant when we add 1 to f, using the same arguments we get $\mathsf{FAI}(f) = \mathsf{FAI}(f + 1) \geq \min(2\mathsf{AI}(f), 1 + \mathsf{AN}(f))$, finally giving:

$$\mathsf{FAI}(f) \geq \min(2\mathsf{AI}(f), 1 + \max(\mathsf{AN}(f), \mathsf{AN}(f + 1))).$$

Plugging the values of $\mathsf{AI}(\mathsf{T}_{d,n})$ and $\mathsf{AN}(1 + \mathsf{T}_{d,n})$ from Lemma 10 and the proof of Lemma 11 in the precedent formula gives the result.

□

Remark 4. 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 $m \geq 2$.

5.4 Parameters of XOR-THR Functions.

The particular structure of XOR-THR functions, direct sum of a linear function and a threshold function make their parameter easy to determine from the one of these two components. The resiliency and nonlinearity can be directly determined by combining Lemma 1 with the parameters of the threshold functions for these criteria (Lemma 7 and Theorem 2). For the exact algebraic immunity, the dimension of vector space of non null annihilators of minimum degree and the bound on the fast algebraic immunity, we need more advanced tools developed in Section 6.

Lemma 12 (Resiliency of XOR-THR Functions). *Let f be the XOR-THR function $\mathsf{XOR}_k + \mathsf{T}_{d,n}$, then:*

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

Proof. The resiliency of XOR_k equals $k - 1$, then combining the first item of Lemma 1 and Lemma 7 gives the result. □

Lemma 13 (Nonlinearity of XOR-THR Functions). *Let f be the XOR-THR function $\mathsf{XOR}_k + \mathsf{T}_{d,n}$, then:*

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

Proof. The nonlinearity of XOR_k is null, then combining the second item of Lemma 1 and Theorem 2 gives the result. □

6 Partitioned Algebraic Normal Form Coefficients and Applications.

In this section we introduce the partitioned normal form coefficients, and we use this tool to prove different results relatively to the algebraic properties of direct sums. In Subsection 6.1 we begin by defining the partitioned algebraic normal form, and we exhibit conditions for a direct sum construction to exceed the maximum algebraic immunity of its 2 components. Then, we use it in Subsection 6.2 to determine the exact algebraic immunity of all XOR-THR functions and a lower bound on the fast algebraic immunity. The result on the algebraic immunity is revisited in Subsection 6.3 to improve the locality bound of local PRGs [AL16]. Another application of the partitioned normal form coefficients is given in Subsection 6.4, enabling to determine the $\mathcal{D}$AN of XOR-THR functions. Finally, in Subsection 6.5 we determine and prove the $\mathcal{D}$AN of DSM functions. It finishes to prove all relevant cryptographic criteria on the two families of function we consider in this article.

6.1 Partitioned Algebraic Normal Form and Algebraic Immunity of Direct Sums.

We develop here some techniques to estimate better the algebraic immunity or the dimension of the space of annihilators of minimal degree in the case of direct sums. First recall from Lemma 1 that the algebraic immunity of a direct sum is always between the maximum of the algebraic immunities of its components and their sum. The upper bound is reduced in [BP05] to:

$$\min\left(\max\left[\mathsf{deg}(f_1), \mathsf{deg}(f_2)\right], \mathsf{AI}(f_1) + \mathsf{AI}(f_2)\right).$$

This upper bound is obtained by considering specific annihilators of f_1 and f_2: $1 + f_1 + f_2$ for the maximum on the degree, and the product of a function defining the algebraic immunity of f_1 by a function defining the algebraic immunity of f_2 for the sum. In the following, we determine sufficient conditions to refine these bounds, beginning with conditions under which the lower bound cannot be achieved with equality.

The ANF (see Definition 2) can be a useful tool to study the algebraic immunity of a function. Here, in the particular case of direct sum, we will modify the form of this representation. Instead of considering binary coefficients related to subsets of all variables, we represent a function in N variables as a function in m variables with $m \leq N$, but with coefficients which are functions in the $N - m$ other variables. We partition then the set of all variables into two sets of sizes m and (say) $n = N - m$. This modification of the representation by ANF makes some concepts easier to study for functions obtained by direct sum (adapting the variable partition to the direct sum).

Definition 22 (Partitioned Algebraic Normal Form). *We call (n,m)-Partitioned Algebraic Normal Form of a Boolean function f its $(n+m)$-variable polynomial representation over $\mathbb{F}_2$ (i.e. belonging to $\mathbb{F}_2[x_1,\dots,x_n,y_1,\dots,y_m]/(x_1^2+x_1,\dots,y_m^2+y_m)$):*

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

where $a_I\in\mathbb{F}_2[x_1,\dots,x_n]/(x_1^2+x_1,\dots,x_n^2+x_n)$.

We call partitioned-(n,m)-ANF coefficients the coefficients a_I.

Note that when the partition into the n and m parts is clear, we shall refer to the (n,m)-partitioned ANF coefficients as PANF coefficients. Note also that the uniqueness of the ANF representation guarantees the uniqueness of the (n,m) partitioned algebraic form. The standard ANF corresponds to the $(0,n)$-partitioned ANF. In the following we give a characterization of the annihilators of a direct sum based on the PANF coefficients.

Lemma 14. *Let f be a Boolean function in the variables $x_1,\dots x_n$ and g be a Boolean function in the variables $y_1,\dots y_m$. Let F denote the direct sum of f and g, let ε be $\in\{0,1\}$, and let h be a function in $x_1,\dots,x_n,y_1,\dots,y_m$ with the (n,m)-partitioned algebraic normal form:*

$$h(x,y)=\sum_{I\subseteq[m]} h_I y^I.$$

We denote accordingly as g_I the (standard) ANF coefficients of g.

If h is an annihilator of $F+\varepsilon$ then the following relation holds on its PANF coefficients:

$$\forall I\subseteq[m],\quad h_I(f+\varepsilon)+\sum_{\substack{J\subseteq[m],K\subseteq[m]\\ J\cup K=I}} h_J g_K=0.$$

Proof. We first consider the case of h being an annihilator of F:

$$\begin{aligned}
Fh=0 &\Leftrightarrow \sum_{I\subseteq[m]} h_I y^I(f+g)=0,\\
&\Leftrightarrow \left(\sum_{I\subseteq[m]} h_I f y^I\right)+\left(\sum_{I\subseteq[m]} h_I y^I\cdot\sum_{J\subseteq[m]} g_J y^J\right)=0,\\
&\Leftrightarrow \sum_{I\subseteq[m]} y^I\left(h_I f+\sum_{\substack{J\subseteq[m],K\subseteq[m]\\ J\cup K=I}} h_J g_K\right)=0,\\
&\Leftrightarrow \forall I\subseteq[m],\quad h_I f+\sum_{\substack{J\subseteq[m],K\subseteq[m]\\ J\cup K=I}} h_J g_K=0.
\end{aligned}$$

Then, for the case of f being an annihilator of $1+F$, *i.e.* $\varepsilon=1$, we apply the same reasoning to $f+1$. $\square$

Now we can show a necessary condition to obtain $\mathsf{AI}(F)>\max(\mathsf{AI}(f),\mathsf{AI}(g))$, using the difference between the smallest degree non null annihilator (see Definition 8) of a function and its complement.

Lemma 15. *Let F be the direct sum of two Boolean functions f and g in respectively n and m variables such that* $\mathsf{AI}(f) \geq \mathsf{AI}(g)$.

$$\textit{If } \deg(g) > 0, \textit{ and } \mathsf{AN}(f) \neq \mathsf{AN}(f+1) \textit{ then } \mathsf{AI}(F) > \mathsf{AI}(f).$$

Proof. First we assume that h is a non null annihilator of F, and we use Lemma 14. Recall that in this lemma, the coefficients g_I, which correspond to the standard ANF coefficients of g, are binary values. Then, we use the following equality:

$$\sum_{\substack{J\subseteq[m],K\subseteq[m]\\ J\cup K=I}} h_J g_K = \sum_{J\subseteq I} h_J \sum_{K\subseteq J} g_{K\cup(I\setminus J)}.$$

So, the equation corresponding to each I is equivalent to the following one:

$$h_I\left(f + \sum_{J\subseteq I} g_J\right) = \sum_{J\subsetneq I} h_J \sum_{K\subseteq J} g_{K\cup(I\setminus J)}. \tag{1}$$

For instance, considering the set $I = \emptyset$, we get the equation:

$$h_\emptyset(f + g_\emptyset) = 0.$$

As we assumed that h is a non null annihilator of F, at least one of the h_I is not null. Let I_0 be such that, for all $J \subsetneq I_0$ $h_J = 0$, and $h_{I_0} \neq 0$, then Equation (1) relatively to I_0 gives that the function h_{I_0} is a non null annihilator of f or $f+1$, giving $\mathsf{AI}(f) \leq \mathsf{deg}(h_{I_0})$ and therefore $|I_0| + \mathsf{AI}(f) \leq \mathsf{deg}(h)$. As $\mathsf{AI}(f) \geq \mathsf{AI}(g)$, using Lemma 1 the unique possibility to get $\mathsf{AI}(F) = \mathsf{AI}(f)$ implies that $I_0 = \emptyset$, *i.e.* $h_\emptyset$ is not null.

Then we consider two cases, either $h_\emptyset$ is the unique non-zero coefficient of h, either there is a least another non-null coefficient. In the former case, for all $I \neq \emptyset$, Equation (1) gives: $0 = h_\emptyset g_I$, which is impossible as g is not constant. In the latter case, let I_1 be such that for all non-empty $J \subsetneq I_1$, we have $h_J = 0$, and $h_{I_1} \neq 0$. Relatively to I_1, Equation (1) gives then:

$$h_{I_1}\left(f + \sum_{J\subseteq I_1} g_J\right) = h_\emptyset g_{I_1}. \tag{2}$$

Note that, since $h_\emptyset$ is non null, for all non-empty $I \subsetneq I_1$, Equation (1) gives: $0 = h_\emptyset g_I$, forcing g_I to be 0. Then from Equation (2) we get:

$$h_{I_1}\left(f + g_\emptyset + g_{I_1}\right) = h_\emptyset g_{I_1}. \tag{3}$$

If $g_{I_1} = 0$ then h_{I_1} is a non null annihilator of $f + g_\emptyset$, leading to $\mathsf{deg}(h) \geq |I_1| + \mathsf{AI}(f) > \mathsf{AI}(f)$. Else, $g_{I_1} = 1$. Multiplying Equation (3) by $(f + g_\emptyset + 1)$ implies then:

$$(h_{I_1} + h_\emptyset)\,(f + g_\emptyset + 1) = 0.$$

So, $h_{I_1} + h_\emptyset$ is an annihilator of $f + g_\emptyset + 1$, and we already know that $h_\emptyset$ is a non-null annihilator of $f + g_\emptyset$. If $\mathsf{deg}(h_{I_1} + h_\emptyset) \neq \mathsf{deg}(h_\emptyset)$ then $\mathsf{deg}(h_{I_1}) \geq \mathsf{deg}(h_\emptyset) \geq \mathsf{AI}(f)$ implying $\mathsf{deg}(h) > \mathsf{AI}(f)$. Otherwise, we have $\mathsf{deg}(h_{I_1} + h_\emptyset) = \mathsf{deg}(h_\emptyset)$, but recall that $h_{I_1} + h_\emptyset$ is then a non-null annihilator of $f + g_\emptyset + 1$, and $h_\emptyset$ is a non-null annihilator of $f + g_\emptyset$; if $h_{I_1} + h_\emptyset$ is not of minimum algebraic degree among all non-null annihilators of $f + g_\emptyset + 1$ or if $h_\emptyset$ is not of minimum algebraic degree among all non-null annihilators of $f + g_\emptyset$, then $\mathsf{deg}(h) > \mathsf{AI}(f)$; and since we assumed $\mathsf{AN}(f) \neq \mathsf{AN}(f + 1)$, they cannot both have minimal degree, leading to $\mathsf{deg}(h) > \mathsf{AI}(f)$. The same reasoning applies to annihilators of $F + 1$, replacing g by $g + 1$, as we mad no assumption on the value of $g_\emptyset$ in the proof.

□

In the following we show how to use these lemmata to determine the remaining parameters of XOR-THR Functions and DSM, and to improve the bound on the locality of local PRGs.

6.2 Algebraic Immunity of XOR-THR Functions.

Lemma 15 enables to prove the exact algebraic immunity of some families of functions, among them is the family of XOR-THR functions. We give the algebraic immunity of these functions in two lemmata, in order to emphasis when the linear part enables to increase the algebraic immunity of the direct sum.

Lemma 16 (XOR-THR Functions with Improved Algebraic Immunity). *Let* $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ *be a XOR-THR function with* $k > 0$ *and* $d \notin \{0, \frac{n+1}{2}, n + 1\}$ *then:*

$$\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \min(d + 1, n - d + 2).$$

Proof. First, using Lemma 1, decomposing $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ as the direct sum of XOR_k and $\mathsf{T}_{d,n}$ we get $\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) \leq \mathsf{AI}(\mathsf{T}_{d,n}) + 1$. Then, we show that this decomposition complies the requirements of Lemma 15 and achives then this upper bound with equality. We take XOR_k as the function g, which does not have null algebraic degree, as $k > 0$. Then, we take the threshold function as the function f, which algebraic immunity is supposed to be more than the one of g, *i.e.* $\mathsf{AI}(\mathsf{T}_{d,n}) \geq 1$, which using Lemma 10 gives $\min(d, n - d + 1) \geq 1$ which is the case as $d \notin \{0, n + 1\}$. The condition $\mathsf{AN}(\mathsf{T}_{d,n}) \neq \mathsf{AN}(1 + \mathsf{T}_{d,n})$ is verified since $d \neq n - d + 1$ by the proof of Lemma 11, which corresponds to $d \neq \frac{n+1}{2}$. Finally applying Lemma 15 gives $\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) > \mathsf{AI}(\mathsf{T}_{d,n})$, which enables to conclude with the first bound and Lemma 10:

$$\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = 1 + \mathsf{AI}(\mathsf{T}_{d,n}) = \min(d + 1, n - d + 2).$$

□

Note that some values of k, d and n are not tackled by Lemma 16. For the extreme cases $k = 0$, $d = 0$ or $d = n + 1$, at least one of the two components is constant, therefore the algebraic immunity of the whole function is determined by the other one (using Lemma 10 if $k = 0$, or Theorem 1). For the case $d = \frac{n+1}{2}$, corresponding to the majority function, the direct sum does not increase the algebraic immunity as we see in the following lemma.

Lemma 17 (XOR-THR Functions without Improved Algebraic Immunity). *Let* $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ *be a XOR-THR function with* $k > 0$ *and* $d = \frac{n+1}{2}$ *then:*

$$\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \frac{n + 1}{2}.$$

Proof. We exhibit a function g of degree $\frac{n+1}{2}$ annihilating $\mathsf{XOR}_k + \mathsf{T}_{d,n}$, with the following shape:

$$g = g_X \cdot \mathsf{XOR}_k + g_T, \text{ where } \mathsf{deg}(g_X) < \frac{n + 1}{2}, \text{ and } \mathsf{deg}(g_T) = \frac{n + 1}{2},$$

where g_X and g_T only depend on the variables of $\mathsf{T}_{d,n}$.
First note that using Property 1:

$$\forall x \in \mathbb{F}_2^n, \quad \mathsf{T}_{\frac{n+1}{2},n}(x+1) = 1 + \mathsf{T}_{\frac{n+1}{2},n}(x). \tag{4}$$

The elementary symmetric function of degree $\frac{n+1}{2}$ in n variables $\sigma_{\frac{n+1}{2}}$ annihilates $1+\mathsf{T}_{\frac{n+1}{2},n}$ and is non null. Then $\mathsf{T}_{\frac{n+1}{2},n}(x)$ is annihilated by $\sigma_{\frac{n+1}{2}}(x+1)$, which can be written (developing its ANF) as $\sigma_{\frac{n+1}{2}}(x) + \sigma'(x)$ where $\sigma'(x)$ is a function of degree strictly less than $\frac{n+1}{2}$. Let us take $g_X = \sigma'$ and $g_T = \sigma_{\frac{n+1}{2}} + \sigma'$, then by construction, $\mathsf{deg}(g) = \frac{n+1}{2}$. We show that g annihilates $\mathsf{XOR}_k + \mathsf{T}_{\frac{n+1}{2},n}$. According to Equation 4:

$$\mathsf{XOR}_k + \mathsf{T}_{\frac{n+1}{2},n} \cdot g = g_X \cdot (\mathsf{XOR}_k)^2 + g_T \cdot \mathsf{XOR}_k + \mathsf{T}_{\frac{n+1}{2},n} \cdot g_X \cdot \mathsf{XOR}_k + g_T \cdot \mathsf{T}_{\frac{n+1}{2},n},$$

as $(\mathsf{XOR}_k)^2 = \mathsf{XOR}_k$, and $g_T = \sigma_{\frac{n+1}{2}} + \sigma'$ is an annihilator of $\mathsf{T}_{\frac{n+1}{2},n}$, all remaining terms are products of XOR_k, giving:

$$\mathsf{XOR}_k + \mathsf{T}_{\frac{n+1}{2},n} \cdot g = \mathsf{XOR}_k \cdot \left(g_X + g_T + \mathsf{T}_{\frac{n+1}{2},n} \cdot g_X\right).$$

By definition, we have:

$$g_X + g_T + \mathsf{T}_{\frac{n+1}{2},n} \cdot g_X = \sigma' + \sigma_{\frac{n+1}{2}} + \sigma' + \mathsf{T}_{\frac{n+1}{2},n} \cdot \sigma' = \sigma_{\frac{n+1}{2}} + \mathsf{T}_{\frac{n+1}{2},n} \cdot \sigma'.$$

Then, as $\sigma_{\frac{n+1}{2}} + \sigma'$ is an annihilator of $\mathsf{T}_{\frac{n+1}{2},n}$, we use that $\mathsf{T}_{\frac{n+1}{2},n} \cdot \sigma' = \mathsf{T}_{\frac{n+1}{2},n} \cdot \sigma_{\frac{n+1}{2}}$, finally giving:

$$\mathsf{XOR}_k + \mathsf{T}_{\frac{n+1}{2},n} \cdot g = \mathsf{XOR}_k \cdot \left(\sigma_{\frac{n+1}{2}}\left(1 + \mathsf{T}_{\frac{n+1}{2},n}\right)\right) = 0,$$

where the last equality comes from the fact that $\sigma_{\frac{n+1}{2}}$ is an annihilator of $1 + \mathsf{T}_{\frac{n+1}{2},n}$. As $\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{\frac{n+1}{2},n}) \geq \frac{n+1}{2}$ using Lemma 1 and Lemma 10, it enables to conclude: $\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{\frac{n+1}{2},n}) = \frac{n+1}{2}$.

□

We can summarize these two lemmata and the previous comment to give the algebraic immunity of all XOR-THR functions:

Corollary 3 (Algebraic Immunity of XOR Threshold Functions). *Let f be the $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ function:*

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

We use these results to prove exhibit a lower bound on the fast algebraic immunity of XOR-THR functions.

Lemma 18 (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 $k > 0$

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

Proof. As seen in the proof of Corollary 2, for any function f:

$$\mathsf{FAI}(f) \geq \min(2\mathsf{AI}(f), 1 + \max(\mathsf{AN}(f), \mathsf{AN}(f+1))).$$

When $k = 0$, $\mathsf{AI}(f) = \min(d, n - d + 1$, and among $\mathsf{AN}(f)$ and $\mathsf{AN}(f+1)$, one take the value d and the other $n - d + 1$, which concludes the first case.

When $k > 1$, $\mathsf{AN}(f) = \mathsf{AN}(f+1)$ then, for $d = (n+1)/2$, $2\mathsf{AI}(f) = n+1$ and $\mathsf{AN}(f) = (n+1)/2$, giving the bound of $(n+3)/2$. For $d \neq (n+1)/2$, $\mathsf{AI}(f) = \mathsf{AN}(f) = 1 + \min(d, n-d+1)$, giving the bound of $2 + \min(d, n-d+1)$.

□

6.3 Improving the Locality of Local PRG.

Lemma 16 and Lemma 17 enable to partially answer the question of Applebaum and Lovett at STOC 2016 ([AL16, AL18]), relatively to the locality of Goldreich's PRGs.

Goldreich's PRG is the main blueprint for local PRG. It has been largely investigated in the theory of complexity community. The function often considered since [MST03] for any stretch s such that $1 < s < 1.5$ is the XOR-THR function in 5 variables $\mathsf{XOR}_3 + \mathsf{T}_{2,2}$. More recently, some generalizations for any stretch have been proposed [AL16, AL18], in term of XOR-MAJ functions, giving that for a stretch s:

- No attack is known relatively to the functions $\mathsf{XOR}_k + \mathsf{T}_{d,2d}$ or $\mathsf{XOR}_k + \mathsf{T}_{d,2d-1}$ since $k \geq 2s$ and $d \geq s$.
- It is proven that no attacks from the linear class neither the algebraic class (both as defined in [AL16] rather than in a symmetric cryptographic context) applies to the functions $\mathsf{XOR}_k + \mathsf{T}_{d,2d}$ or $\mathsf{XOR}_k + \mathsf{T}_{d,2d-1}$ since $k \geq 5s$ and $d \geq 18s$.

Rephrasing with the Boolean function vocabulary, the authors of [AL16, AL18] ask what is the minimum n necessary to have a Boolean function f in $\mathcal{B}_n$ such that $\mathsf{AI}(f) = e$ and $\mathsf{res}(f) = k$, and they show that $n \leq 2e + k$. This minimum n corresponds then to the minimum locality for a given stretch. In the following theorem we improve this bound, showing that $n \leq 2e + k - 1$. The proof is constructive, we give two families of functions satisfying this bound (for all n big enough). The new tool developed for the algebraic immunity of XOR-THR functions enable to partially answer, giving the following theorem:

Theorem 3 (Upper Bound on Predicate's Locality). *Let $k \in \mathbb{N}$, and $e \in \mathbb{N}^*$, there exists Boolean function f in n variables with $n \leq k + 2e - 1$ such that* $\mathsf{AI}(f) = e$, *and* $\mathsf{res}(f) = k$.

Moreover, the families of functions $\mathsf{XOR}_k + \mathsf{T}_{e,2e-1}$ *and* $\mathsf{XOR}_{k+1} + \mathsf{T}_{e-1,2e-2}$ *are examples of such functions.*

Proof. As $\mathsf{XOR}_k + \mathsf{T}_{e,2e-1}$ and $\mathsf{XOR}_{k+1} + \mathsf{T}_{e-1,2e-2}$ are functions in $k + 2e - 1$ variables, the existence proof is direct when the algebraic immunity and the resiliency of such functions are proven.

We begin with the functions of the form $\mathsf{XOR}_k + \mathsf{T}_{e,2e-1}$. The threshold part has a resiliency of 0 using Lemma 7, and $k - 1$ for the part XOR_k then $\mathsf{res}(\mathsf{XOR}_k + \mathsf{T}_{e,2e-1}) = k$ using Lemma 1. Then, we are in the case described by Lemma 17 so $\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{e,2e-1}) = e$.

Using the same lemmata for the functions of the form $\mathsf{XOR}_{k+1} + \mathsf{T}_{e-1,2e-2}$ we obtain $\mathsf{res}(\mathsf{XOR}_{k+1} + \mathsf{T}_{e-1,2e-2}) = k$ as $\mathsf{T}_{e-1,2e-2}$ is unbalanced and $\mathsf{AI}(\mathsf{XOR}_{k+1} + \mathsf{T}_{e-1,2e-2}) = e - 1 + 1 = e$. □

Remark 5. In [AL16, AL18], the function reaching the bound of $2e + k$ is referred as the XOR of XOR_k variables and the majority predicate on $2e$ bits, which can be defined in different ways. To fulfill the constraints on the resiliency the majority part has to be balanced so it differs from the function $\mathsf{T}_{e,2e}$ we consider.

6.4 Determining the $\mathcal{D}\mathsf{AN}$ of XOR-THR Functions.

Using Lemma 14 to Lemma 17, and a proposition appearing in [CDM+18] we can get the $\mathcal{D}\mathsf{AN}$ of XOR-THR functions and their complementary. We first rephrase this proposition relatively to our notation, and then we give the $\mathcal{D}\mathsf{AN}$ of XOR-THR functions.

Proposition 2 ([CDM+18] Rephrased). *Let f be the direct sum of XOR_k with $k > 0$ and g, then $\mathcal{D}\mathsf{AN}(f) = \mathcal{D}\mathsf{AN}(f+1)$.*

Lemma 19 **($\mathcal{D}\mathsf{AN}$ 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 $\mathcal{D}\mathsf{AN}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \mathcal{D}\mathsf{AN}(1 + \mathsf{XOR}_k + \mathsf{T}_{d,n})$.

Proof. First, we show that considering $\mathcal{D}\mathsf{AN}(1+\mathsf{XOR}_k+\mathsf{T}_{d,n})$ with $d \leq (n+1)/2$ is sufficient to determine the $\mathcal{D}\mathsf{AN}$ of functions tackled by the lemma. Note that using Proposition 2 we directly get $\mathcal{D}\mathsf{AN}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \mathcal{D}\mathsf{AN}(1 + \mathsf{XOR}_k + \mathsf{T}_{d,n})$. Then, using Property 1, the affine transformation defined as:

$$\forall (x,y) \in \mathbb{F}_2^{k+n}, \quad \mathsf{XOR}_h(x) + \mathsf{T}_{d,n}(y) = \mathsf{XOR}_h(x) + \mathsf{T}_{d,n}(y+1),$$

maps $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ to $1 + \mathsf{XOR}_k + \mathsf{T}_{n-d+1,n}$. As the dimension of the annihilator of fixed degree is affine invariant $\mathcal{D}\mathsf{AN}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \mathcal{D}\mathsf{AN}(1 + \mathsf{XOR}_k + \mathsf{T}_{n-d+1,n})$. Regrouping these two remarks:

$$\begin{aligned} \mathcal{D}\mathsf{AN}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) &= \mathcal{D}\mathsf{AN}(1 + \mathsf{XOR}_k + \mathsf{T}_{d,n}) = \mathcal{D}\mathsf{AN}(\mathsf{XOR}_k + \mathsf{T}_{n-d+1,n}) \\ &= \mathcal{D}\mathsf{AN}(1 + \mathsf{XOR}_k + \mathsf{T}_{n-d+1,n}), \end{aligned}$$

which enable to determine the $\mathcal{D}\mathsf{AN}$ only considering $d \leq (n+1)/2$, more particularly focusing on $\mathcal{D}\mathsf{AN}(1+\mathsf{XOR}_k + \mathsf{T}_{d,n})$.

We proceed then in three steps: first we study the PANF coefficients of a direct sum with g being a XOR function. Then, we focus on the XOR-THR functions where the threshold function is the majority function (*i.e.* $d = (n+1)/2$), case of Lemma 17). Finally we determine the $\mathcal{D}\mathsf{AN}$ of the other XOR-THR functions for which $d < (n+1)/2$ (case of Lemma 16), also answering to the case $d > (n+1)/2$.

Direct sum with a XOR function and PANF coefficients:

First, we focus on the behavior of the PANF coefficients of an annihilator $h(x,y)$ of any direct sum $f(x) + g(y)$ where g is a XOR function of $m \neq 0$ variables. Applying Lemma 14 gives the following equation on the coefficients (proof given in the proof of Lemma 15):

$$h_I\left(f + \sum_{J \subseteq I} g_J\right) = \sum_{J \subsetneq I} h_J \sum_{K \subseteq J} g_{K \cup (I \setminus J)}.$$

As the g_I coefficients correspond to the standard ANF coefficients of g, g being XOR_m it gives:

$$g_I = 1 \text{ if } |I| = 1, \quad \text{and } g_I = 0 \text{ otherwise.}$$

It leads to the following equations:

$$\forall I \subseteq [m], \quad h_I\,(f + (|I| \mod 2)) = \sum_{\substack{J \subsetneq I \\ |I\setminus J|=1}} h_J. \tag{5}$$

These equations are way more simple than in the general case, in the following we determine the potential value of the h_I coefficients for the non null annihilators of minimal degree of the XOR-THR functions (or complementary). We rewrite the conditions on the PANF coefficient given by Equation (5) as a system, identifying different cases based ont the cardinal of the subset I:

$$\begin{cases} \text{if }, |I| = 0, \text{ then } I = \emptyset, & h_\emptyset f = 0, \\ \text{if }, |I| = 1, \text{ then } I = \{i\}, & h_{\{i\}}(f+1) = h_\emptyset, \\ \text{if }, |I| = 2, \text{ then } I = \{i,j\}, & h_I f = h_{\{i\}} + h_{\{j\}}, \\ \text{if }, |I| > 2, & h_I\,(f + (|I| \mod 2)) = \sum_{\substack{J \subsetneq I \\ |I\setminus J|=1}} h_J. \end{cases}.$$

Case $d = (n+1)/2$**:**

In this case, from Lemma 17, we know that $\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = \frac{n+1}{2} = d$. Consequently it gives the following bound on the PANF coefficients: $\mathsf{deg}(h_I) \leq d - |I|$. We consider the system of PANF coefficients described in the precedent part with $f = \mathsf{T}_{d,n}$.

Then, based on Lemma 11 we know that for $d = (n+1)/2$ both $\mathsf{T}_{d,n}$ and its complementary have not non null annihilators of degree less than d. So, the first equation ($|I| = 0$) implies that $h_\emptyset$ is an annihilator of $\mathsf{T}_{d,n}$ of degree d or the null function. As we saw in the proof of Lemma 15, $h_\emptyset \neq 0$ otherwise $\mathsf{AI}(f+g) > \mathsf{AI}(f)$ which is not the case here, hence $\mathsf{deg}(h_\emptyset) = d$.

The second equation ($|I| = 1$) implies:

$$\forall i \in [m], \quad \begin{cases} (h_{\{i\}} + h_\emptyset)(f+1) = 0, \\ h_\emptyset f \qquad\qquad\quad = 0. \end{cases}.$$

Considering the first equation with any pair (i,j), with $i \in [m]$, $j \in [m]$ gives $(h_i + h_j)(f+1) = 0$, and as $\mathsf{deg}(h_I) \leq d - |I|$, it means that $h_i = h_j$. Hence, all PANF coefficients related to a subset of cardinal 1 are equal, and for all $I \subseteq [m]$ such that $|I| \geq 2$, $h_I = 0$ (by induction on the cardinal of the subset, taking $|I| = 2$ as initialization step).

Then, any non null annihilator of $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ in this case has the form $h_\emptyset + h_1(\mathsf{XOR}_k)$. We finish this part by showing that when $h_\emptyset$ is chosen, it fixes h_1, enabling to conclude on the dimension of vector space of non null annihilators of minimum degree. Let us write $h_\emptyset$ as $h_{0,<d} + h_{0,d}$, where we separate the function $h_\emptyset$ (in ANF representation) in two parts, one with all monomials of degree less than d and the other part of degree exactly d.

As $h_\emptyset f = 0$, $\mathsf{deg}(h_\emptyset) = d$, and $\mathcal{D}\mathsf{AN}(\mathsf{T}_{d,n}) = \binom{n}{(n-1)/2}$ by Lemma 11 we consider as $h_\emptyset$ any non null element of this vector space. Then, we consider the remaining equation: $(h_1 + h_\emptyset)(1 + \mathsf{T}_{d,n} = 0)$. As the product of any function with only monomials of degree d or more with $1 + \mathsf{T}_{d,n}$ gives 0, we obtain $h_1 = h_{0,<d}$. Therefore, any non null annihilator of minimal degree of $\mathsf{XOR}_k + \mathsf{T}_{(n+1)/2,n}$ can be written as:

$$h_\emptyset + h_{0,<d} \cdot (\mathsf{XOR}_k), \text{ where } h_\emptyset(\mathsf{T}_{d,n}) = 0, \text{ and } h_\emptyset \neq 0.$$

Finally, as the null function is always an annihilator, we get:

$$\mathcal{D}\mathsf{AN}\left(\mathsf{XOR}_k + \mathsf{T}_{\frac{n+1}{2},n}\right) = \mathcal{D}\mathsf{AN}\left(\mathsf{T}_{\frac{n+1}{2},n}\right) = \binom{n}{\frac{n+1}{2}}.$$

Case $d < (n+1)/2$**:**

In this case (note that we do not consider the case $d = 0$), from Lemma 16, we know that $\mathsf{AI}(\mathsf{XOR}_k + \mathsf{T}_{d,n}) = d+1$. Consequently it gives the following bound on the PANF coefficients: $\deg(h_I) \leq d+1-|I|$. We consider the system of PANF coefficients described in the first part with $f = 1 + \mathsf{T}_{d,n}$.

The first equation ($|I| = 0$) gives that $h_\emptyset = 0$ or $d \leq \deg(h_\emptyset) \leq d+1$. Similarly to the precedent part, the second equation ($|I| = 1$) implies:

$$\forall i \in [m], \quad (h_{\{i\}} + h_\emptyset)(f+1) = 0. \tag{6}$$

As $\deg(h_\emptyset + h_{\{i\}}) \leq d+1$, two cases appear depending on d, indeed, from (the proof of) Lemma 19 $(1+f) = \mathsf{T}_{d,n}$ has not non null annihilator of degree less than $n-d+1$, giving a different situation for $d+1 = n-d+1$ *i.e.* $d = \frac{n}{2}$ and for $d+1 \neq n-d+1$.

We begin with the case $d \neq \frac{n}{2}$. Equation (6) gives that $\forall i \in [m]$ we have $h_{\{i\}} = h_\emptyset$, and by induction $h_I = 0$ for all I such that $|I| \geq 2$. As $\deg(h_i) \leq d$, $h_\emptyset$ is a degree d annihilator of $1 + \mathsf{T}_{d,n}$ or the null function. All annihilators of $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ are then of the form $h_\emptyset + h_\emptyset \cdot (\mathsf{XOR}_k)$, and using Lemma 11 we can conclude in this case:

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

Finally, we consider the case $d = \frac{n}{2}$. In this case, Equation (6) gives that $\forall i \in [m]$, the function $h_{\{i\}} + h_\emptyset$ of degree at most $d+1$ annihilates $\mathsf{T}_{d,n}$, and as $d = \frac{n}{2}$ from Lemma 11 we know that $\mathsf{AN}(\mathsf{T}_{d,n}) = d+1$. So either $h_\emptyset = h_{\{i\}}$ as in the precedent case, either $h_{\{i\}} + h_\emptyset$ is a degree $d+1$ annihilator of $\mathsf{T}_{d,n}$. Then, we determine the other PANF coefficients,. From the system, the equation for the case $|I| = 2$ implies that for any $i \in [m]$ and $j \in [m] \backslash i$, we get $(h_i + h_j)\mathsf{T}_{d,n} = 0$ whereas $\deg(h_{\{i\}} + h_{\{j\}}) \leq d$. Thereafter, $h_i = h_j$ for any i and j both taken in $[m]$, and by induction all h_I such that $|I| \geq 2$ are null. Then, we choose $h_\emptyset$ in a particular vectorial space and show that only one value for h_i is possible, and we express it relatively to $h_\emptyset$, giving the expression of all annihilator of $\mathsf{XOR}_k + \mathsf{T}_{\frac{n}{2}d,n}$ of degree less than or equal to $d+1$ when $d = \frac{n}{2}$.

As $\mathsf{AN}(1+\mathsf{T}_{d,n}) = d$, and as for all functions with all monomials (ANF representation) of degree greater than or equal to d the product with $1 + \mathsf{T}_{d,n}$ gives the null function, the vectorial space of annihilators of $1 + \mathsf{T}_{d,n}$ of degree less than or equal to $d+1$ has dimension $\binom{n}{d} + \binom{n}{d+1}$. Note that it corresponds to the null function and the functions with monomials of degree d and $d+1$ only, we denote this vectorial space $\mathcal{S}$. Let $h_\emptyset \in \mathcal{S}$, we consider the affine transformation $x \mapsto x+1$, transforming $\mathsf{T}_{d,n}(x)$ to $\mathsf{T}_{d,n}(x+1)$ for all $x \in \mathbb{F}_2^n$. Using Property 1, we know that $\mathsf{T}_{d,n}(1+x) = 1 + \mathsf{T}_{n-d+1,n}(x)$, then we can equivalently write (for any $i \in [m]$):

$$\begin{aligned}
&\forall x \in \mathbb{F}_2^n, \quad (h_\emptyset + h_{\{i\}})(x)\mathsf{T}_{d,n}(x) = 0, \\
\iff &\forall x \in \mathbb{F}_2^n, \quad (h_\emptyset + h_{\{i\}})(1+x)\mathsf{T}_{d,n}(1+x) = 0, \\
\iff &\forall x \in \mathbb{F}_2^n, \quad (\widetilde{h_\emptyset} + \widetilde{h_1})(x)(1 + \mathsf{T}_{d+1,n})(x) = 0,
\end{aligned}$$

where $\widetilde{h_\emptyset}$ denotes the function such that $\forall x \in \mathbb{F}_2^n, \quad \widetilde{h_\emptyset}(x) = h_\emptyset(1+x)$, and similarly for $\widetilde{h_i}$.

As the affine transformation $x \mapsto x+1$ keeps the degree, we can write $\widetilde{h_\emptyset}$ as $\widetilde{h_{0,<d+1}} + \widetilde{h_{0,d+1}}$ (similarly to what we did on $h_\emptyset$ in the case $d = (n+1)/2$)). For all functions with all monomials (in ANF representation) of degree greater than or equal to $d+1$ the product with $1 + \mathsf{T}_{d+1,n}$ gives the null function, then:

$$(\widetilde{h_{0,<d+1}} + \widetilde{h_i})(1 + \mathsf{T}_{d+1,n}) = 0.$$

As $\deg(\widetilde{h_{0,<d+1}} + \widetilde{h_i}) \leq d$, it implies that $\widetilde{h_{0,<d+1}} = \widetilde{h_i}$, and as the transformation $x \mapsto x+1$ is a bijection, h_i is fixed by the choice of $h_\emptyset$. In conclusion, it gives that $h_\emptyset + h'_{0,<d+1} \cdot (\mathsf{XOR}_k)$, where $\widetilde{h'_{0,<d+1}}$ is the

result of $\widetilde{h_{0,<d+1}}$ through the transformation $x \mapsto x+1$, is an annihilator of $1 + \mathsf{XOR}_k + \mathsf{T}_{d,n}$ of minimal degree (or null) for any $h_\emptyset \in \mathcal{S}$. Finally it gives:

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

concluding the case $d < (n+1)/2$. The identity $\binom{n}{k} = \binom{n}{n-k}$ enables to obtain the value of $\mathcal{D}\mathsf{AN}(\mathsf{XOR}_k + \mathsf{T}_{d,n})$ for the case $d > (n+1)/2$.

□

Note that this lemma does not give the $\mathcal{D}\mathsf{AN}$ of XOR functions (the threshold part is constant, thus the result is given in Lemma 2) or threshold functions which is already given in Lemma 11.

6.5 Determining the $\mathcal{D}\mathsf{AN}$ of DSM.

We use Lemma 14 to compute the dimension of the annihilators of minimal degree of a DSM. First we show the behavior of this dimension on particular functions.

Lemma 20. *Let f be a degree $k > 0$ DSM, we consider the direct sum of f and the monomial y^T such that $t = |T| \geq k$, then:*

$$\mathcal{D}\mathsf{AN}(f + y^T) \leq \begin{cases} 1 & \textit{if}\ \mathsf{AI}(f) = t, \\ 1 + t \cdot \mathcal{D}\mathsf{AN}(f) & \textit{if}\ \mathsf{AI}(f) = t-1, \\ t \cdot \mathcal{D}\mathsf{AN}(f) & \textit{otherwise.} \end{cases}$$

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

Proof. In order to study the annihilators of $f + y^T$, we first use Lemma 14 on the (n,t)-partitioned algebraic normal form of $f + y^T$. As g is here a monomial function, for all $J \subsetneq T$ we get $g_J = 0$, and $g_T = 1$. Relatively to the PANF coefficients it gives:

$$\begin{cases} h_J f = 0 & \text{for all } J \subsetneq T, \\ h_T(f+1) = \sum\limits_{J \subsetneq T} h_J & \text{for the substet } T. \end{cases}$$

As we are looking for annihilators h of minimal degree, we also use the following equation for each subset $J \subseteq T$:

$$\mathsf{deg}(h_J) \leq \mathsf{AI}(f + y^T) - |J|. \qquad (7)$$

Then, three points are of main interest in this proof, and various cases will be derived from this points. First, the algebraic immunity of $f + y^T$ impacts the number of annihilators, its value (ether $\mathsf{AI}(f)$ or $\mathsf{AI}(f) + 1$) affects the cardinal of the subsets J such that h_J can be not null. Then, we analyze the degree of the PANF coefficients, more particularly, the algebraic immunity of f and its relation with t determines the potential degree of h_T, giving different annihilators. Finally, the algebraic immunity is a criterion which considers f and $f + 1$, whereas $\mathsf{AN}(f)$ is only related to f. Even for DSM, we can get $\mathsf{AN}(f) \neq \mathsf{AN}(f+1)$, therefore we take care of the cases where $\mathsf{AI}(f) < \mathsf{AN}(f)$ and $\mathsf{AI}(f) = \mathsf{AN}(f)$ as we do not assume the value of the constant coefficient.

Algebraic immunity of $f + y^T$:

As $f + y^T$ is a degree t function, its algebraic immunity is at most t, we show that if $\mathsf{AI}(f) < t$ then $\mathsf{AI}(f + y^T) = \mathsf{AI}(f) + 1$, and else, the case $\mathsf{AI}(f) = t$ implies $\mathsf{AI}(f + y^T) = \mathsf{AI}(f)$.

Let start with the case $\mathsf{AI}(f) < t$, we use the formalization of direct sum vector, considering $\mathbf{m}_f = [m_1, \dots, m_k]$ and $\mathbf{m}_{f+y^T}$ being the same vector completed with zeros and a 1 added at the position t:

$$\mathbf{m}_{f+y^T} = \begin{cases} [m_1, \dots, m_{k-1}, m_k + 1] & \text{if } t = k, \\ [m_1, \dots, m_k, \underbrace{0, \dots, 0}_{t-k-1}, 1] & \text{otherwise.} \end{cases}$$

We deduce the algebraic immunity of $f + y^T$ using the characterization of Theorem 1 on $f + y^T$. If $t = k$, the coefficient k of $\mathbf{m}_{f+y^T}$ is then $m_k + 1$, it gives:

$$\mathsf{AI}(f + y^T) = \min\left(\min_{0 \le d < k}\left[d + \left(\sum_{i=d+1}^{k} m_i\right) + 1\right], k\right).$$

As we are considering the case $\mathsf{AI}(f) < t$, which corresponds to $\mathsf{AI}(f) < k$ here it means that:

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

therefore the precedent equation can be simplified as:

$$\mathsf{AI}(f + y^T) = \min(\mathsf{AI}(f) + 1, k).$$

As $\mathsf{AI}(f) < k$, if $\mathsf{AI}(f) = k - 1$ then $\mathsf{AI}(f + y^T) = k = \mathsf{AI}(f) + 1$. Otherwise, $\mathsf{AI}(f) < k - 1$, then the minimum is directly $\mathsf{AI}(f) + 1$. It finishes the study of $\mathsf{AI}(f + y^T)$ when $\mathsf{AI}(f) < t$ and $t = k$.

If $k < t$, the characterization of Theorem 1 gives:

$$\mathsf{AI}(f + y^T) = \min\left(\min_{0 \le d \le k}\left[d + \left(\sum_{i=d+1}^{k} m_i\right) + 1\right], \min_{k < d < t} d + 1, t\right).$$

It can be simplified to $\mathsf{AI}(f + y^T) = \min(\mathsf{AI}(f) + 1, k + 2, t)$. As f is a degree k function and $t > k$, the minimum is always equal to $\mathsf{AI}(f) + 1$. It enables to concludes on the algebraic immunity of $f + y^T$ when $\mathsf{AI}(f) < t$.

The remaining case corresponds to $\mathsf{AI}(f) = t$. In this case f is already a function with algebraic immunity t, the function $f + y^T$ cannot have an algebraic immunity higher than t as its degree is equal to t, and as it is a direct sum we know by Lemma 1 that its algebraic immunity is at least $\mathsf{AI}(f)$, which is t. In conclusion, for the case $\mathsf{AI}(f) = t$ we get $\mathsf{AI}(f + y^T) = \mathsf{AI}(f)$.

On the degree of the PANF coefficients:

Considering Equation (7) with the precedent result on $\mathsf{AI}(f + y_t)$ gives more insight on the annihilators. In the case $\mathsf{AI}(f) = t$ Equation (7) can be rewritten as:

$$\begin{aligned} \mathsf{deg}(h_J) &\le \mathsf{AI}(f) - |J|, \\ &\le t - |J|. \end{aligned}$$

Using the system given by the PANF coefficients we get:

$$\begin{cases} \mathsf{deg}(h_\emptyset) \le t \text{ and } h_\emptyset f = 0 & \text{for the subset } \emptyset, \\ h_J = 0 & \text{for all } J \subsetneq T \text{ such that } 1 \le |J| < t, \\ \mathsf{deg}(h_T) \le 0 \text{ and } h_T(f+1) = h_\emptyset & \text{for the substet } T. \end{cases}$$

We focus on the potential value of h_T, if $h_T = 0$ then $h_\emptyset$ is null, which gives that h is the null function. Otherwise, $h_T = 1$, then it forces $h_\emptyset$ to be $1 + f$, which is possible as $\mathsf{deg}(f+1) = \mathsf{deg}(f) = t > 0$. In this case we can conclude: $\mathcal{D}\mathsf{AN}(f + y^T) = 1$.

Now we consider the case $\mathsf{AI}(f) < t$, it means that $\mathsf{AI}(f+y^T) = \mathsf{AI}(f)+1$ and therefore we can rewrite Equation (7) as:

$$\mathsf{deg}(h_J) \le \mathsf{AI}(f) + 1 - |J|. \tag{8}$$

Using the system given by the PANF coefficients we get:

$$\begin{cases} \mathsf{deg}(h_\emptyset) \le \mathsf{AI}(f) + 1 \text{ and } h_\emptyset f = 0 & \text{for the subset } \emptyset, \\ \mathsf{deg}(h_J) \le \mathsf{AI}(f) \text{ and } h_J f = 0 & \text{for all } J \subsetneq T \text{ such that } |J| = 1, \\ h_J = 0 & \text{for all } J \subsetneq T \text{ such that } 1 < |J| < t, \\ \mathsf{deg}(h_T) \le \mathsf{AI}(f) + 1 - t \text{ and } h_T(f+1) = \sum\limits_{\substack{J \subsetneq T \\ |J| \le 1}} h_J & \text{for the substet } T. \end{cases}$$

Focusing on the last equation, note that h_T can always be null. As we are in the case $\mathsf{AI}(f) < t$, the only possibility for h_T to be non null is when $\mathsf{AI}(f) = t - 1$. As $\mathsf{AI}(f) \le k$ and $k \le t$ it occurs only in two cases: $t = k$ and $\mathsf{AI}(f) = k - 1$, or $t = k + 1$ and $\mathsf{AI}(f) = k$. In these two cases h_T can be equal to 1, leading to the following equation:

$$f + 1 = \sum_{\substack{J \subsetneq T \\ |J| \le 1}} h_J. \tag{9}$$

It finishes our study on the potential degree of the PANF coefficients.

Determining $\mathcal{D}\mathsf{AN}(f)$:

For the case $\mathsf{AI}(f) = t$ we already know that $\mathcal{D}\mathsf{AN}(f + y_t) = 1$, in fact in corresponds to a case where $\mathsf{AI}(f) = \mathsf{deg}(f)$ so where $\mathsf{AN}(f) = \mathsf{AI}(f)$. In the following we show that this difference between $\mathsf{AN}(f)$ and $\mathsf{AI}(f)$ influences $\mathcal{D}\mathsf{AN}(f + y_t)$.

We consider the previous system of equations through the various values of this difference:

- Case $\mathsf{AN}(f) \ge \mathsf{AI}(f) + 2$.
  As $h_\emptyset f = 0$ and $\deg(h_\emptyset) < \mathsf{AN}(f)$ with this condition, it forces $h_\emptyset$ to be null. By a similar argument, all the PANF coefficients h_J such that $|J| = 1$ are null. Then, if we are in the case where h_T can be equal to 1, it implies that $f + 1 = 0$ which is impossible, therefore $h_T = 0$, the null function is the only annihilator of $f + y^T$ of degree less than or equal to $\mathsf{AI}(f + y^T)$, giving $\mathcal{D}\mathsf{AN}(f) = 0$.
- Case $\mathsf{AN}(f) = \mathsf{AI}(f) + 1$.
  In this case, as $h_J f = 0$ and $\deg(h_J) < \mathsf{AN}(f)$ for all $J \subsetneq T$ such that $|J| = 1$, all these PANF coefficients are null. Then, if $h_\emptyset = 0$ it corresponds to the setting of the precedent item, where $h = 0$ is the unique possibility. Otherwise, $h_\emptyset$ is a $\mathsf{AN}(f)$-degree non null annihilator of f, and Equation (9) gives $h_T(f + 1) = h_\emptyset$. Then, $h_T = 0$ is impossible, and when $h_T = 1$ is possible, it forces $h_\emptyset = f + 1$. Note that for the case $t = k + 1$ and $\mathsf{AI}(f) = k$ which enables to consider $h_T = 1$, the degree and the algebraic immunity of f are equal, then $\mathsf{AN}(f) \ne \mathsf{AI}(f) + 1$, so we cannot consider this case here. Only

when $t = k$ and $\mathsf{AI}(f) = k - 1$ this possibility appears, (it is the case for example for $f = 1 + x_1x_2$), for such cases the annihilators are 0 and $1 + f$, giving $\mathcal{D}\mathsf{AN}(f) = 1$.
For all other cases of this item, 0 is the unique annihilator of degree less than or equal to $\mathsf{AI}(f + y^T)$ then we get $\mathcal{D}\mathsf{AN}(f + y^T) = 0$.

- Case $\mathsf{AN}(f) = \mathsf{AI}(f)$.
In this case many choices of annihilators can be considered. $h_\emptyset$ can be an annihilator of of f of degree less than or equal to $\mathsf{AN}(f)+1$. For all $J \subsetneq T$ such that $|J| = 1$, h_J can be an annihilator of f of degree $\mathsf{AN}(f)$ or the null function. The only restriction is coming from the last equation:

$$h_T(f + 1) = \sum_{\substack{J \subsetneq T \\ |J| \leq 1}} h_J.$$

We consider the two possible cases for the value of h_T, and then the annihilators of $f + y^T$ it implies.

If $h_T = 0$, then the sum of the PANF coefficients such that $J \subsetneq T,\ |J| \leq 1$, is null. First it implies that $h_\emptyset$ cannot be of degree $\mathsf{AN}(f) + 1$. Then, among the $t + 1$ coefficients, exactly t can be chosen freely in the vectorial space of annihilators of f of degree less than or equal to $\mathsf{AN}(f)$ and the last one is equal to the sum. By construction, it gives $t \cdot \mathcal{D}\mathsf{AN}(f)$ linearly independent annihilators of $f + y^T$.

If $h_T = 1$, then the sum of the PANF coefficients such that $J \subsetneq T,\ |J| \leq 1$, is equal to $f + 1$. We saw below that only two cases enable h_T be be equal to 1, we treat separately these two cases.

If $t = k$ and $\mathsf{AI}(f) = k-1$, then for all $J \subsetneq T$ $|J| = 1$ the coefficients h_J are such that $\mathsf{deg}(h_T) \leq k-1$ whereas $f + 1$ has degree k. Therefore $h_\emptyset$ has to be of degree k, more precisely with all monomials of degree k being the ones of $f + 1$. Then, $h_\emptyset$ can be taken as any function of the form $f + 1 + h'$ where h' is a function such that $\mathsf{deg}(h') \leq k - 1$, under the restriction that $h_\emptyset$ annihilates f. As $f + 1$ is an annihilator of f, and as the annihilators of degree lesser than or equal to a fixed value form a vectorial space, h' is an annihilator of f of degree $\mathsf{AN}(f)$ or is null.
So, building an annihilator of $f + y^T$ under these conditions consists in choosing h' and $t - 1$ other coefficients from the vectorial space of annihilators of f degree less than or equal to $\mathsf{AN}(f)$, the last coefficient being equal to the sum of the others in order to fulfill Equation (9). Note that all the annihilators of $f + y^T$ created this way are of the form $f + 1 + y^T + h''$, where h'' is an annihilator of $f + y^T$ given by the case $h_T = 0$ (indeed, h' and the h_J such that $|J| = 1$ are chosen exactly under the same constraints as $h_\emptyset$ and the same h_J in the previously studied case). In conclusion, for the case $t = k$ and $\mathsf{AI}(f) = k - 1$ we get $\mathcal{D}\mathsf{AN}(f + y^T) = 1 + t \cdot \mathcal{D}\mathsf{AN}(f)$.

If $t = k + 1$ and $\mathsf{AI}(f) = k$, then for all $J \subsetneq T$ $|J| = 1$ the coefficients h_J are such that $\mathsf{deg}(h_T) \leq k$, $\mathsf{deg}(h_\emptyset) \leq k + 1$, and $\mathsf{deg}(f + 1) = k$. Then $h_\emptyset$ cannot be of degree $k + 1$, thereafter building an annihilator of $f + y^T$ under these conditions consists in choosing t coefficients from the vectorial space of annihilators of f of degree less than or equal to $\mathsf{AN}(f)$, the last one being equal to $f + 1$ plus this sum. Note that summing any two such built annihilator of $f + y^T$ gives an annihilator obtained in the case $h_T = 0$ (indeed, it gives an annihilator such that the $t + 1$ PANF coefficients related to the $t + 1$ smallest subsets are all annihilators of f of degree less than or equal to k such that the sum is null). In conclusion, for the case $t = k + 1$ and $\mathsf{AI}(f) = k$ we get $\mathcal{D}\mathsf{AN}(f + y^T) = 1 + t \cdot \mathcal{D}\mathsf{AN}(f)$.
Putting all together:
Let summarize what we obtain for the different cases:

- If $\mathsf{AI}(f) = t$ then $\mathcal{D}\mathsf{AN}(f + y^T) = 1$.
- Else, $\mathsf{AI}(f) < t$:
  - $*$ If $\mathsf{AN}(f) \geq \mathsf{AI}(f) + 2$, then $\mathcal{D}\mathsf{AN}(f + y^T) = 0$.
  - $*$ If $\mathsf{AN}(f) = \mathsf{AI}(f) + 1$, then:
    - $\cdot$ If $t = k$ and $\mathsf{AI}(f) = k - 1$ then $\mathcal{D}\mathsf{AN}(f + y^T) = 1$.
    - $\cdot$ Else, $\mathcal{D}\mathsf{AN}(f + y^T) = 0$.
  - $*$ Else, $\mathsf{AN}(f) = \mathsf{AI}(f)$:
    - $\cdot$ If $t = k$ and $\mathsf{AI}(f) = k - 1$ then $\mathcal{D}\mathsf{AN}(f + y^T) = 1 + t \cdot \mathcal{D}\mathsf{AN}(f)$.
    - $\cdot$ If $t = k + 1$ and $\mathsf{AI}(f) = k$ then $\mathcal{D}\mathsf{AN}(f + y^T) = 1 + t \cdot \mathcal{D}\mathsf{AN}(f)$.
    - $\cdot$ Else, $\mathcal{D}\mathsf{AN}(f + y^T) = t \cdot \mathcal{D}\mathsf{AN}(f)$.

Recalling that the two cases $\mathsf{AI}(f) = k$ and $t = k + 1$, and $\mathsf{AI}(f) = k - 1$ and $t = k$ correspond to the case $\mathsf{AI}(f) = t - 1$ enables to conclude.

□

Using Lemma 20 recursively we can compute the dimension of the annihilators of minimal degree of any DSM, as stated in the following theorem.

Theorem 4. *Let f be a DSM with associated direct sum vector $\mathbf{m}_f = [m_1, \ldots, m_k]$, let 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.

Proof. Note that Lemma 20 considers functions $f + y^T$ with f non constant, here to prove the final statement we also determine the $\mathcal{D}\mathsf{AN}$ of monomial functions. To link this dimension for a DSM to its direct sum vector we study more particularly the homogeneous DSM and the behavior of the $\mathcal{D}\mathsf{AN}$ when adding a homogeneous DSM of degree t to a DSM of degree k with $k < t$. Consequently we separate the proof in four parts, first we study the $\mathcal{D}\mathsf{AN}$ of monomial functions, then we focus on homogeneous DSM, thus we establish a corollary of Lemma 20 suited for addition of homogeneous DSM, and finally we prove the upper bound of the theorem by recursion.

On the $\mathcal{D}\mathsf{AN}$ of monomial functions:

Let f be a monomial function: $f = x^T$, with $|T| \in \mathbb{N}^*$. As we are considering direct sums of monomials, we neglect the null function (giving $|T| = t > 0$), and we also study the annihilators of $1 + x^T$. Throughout the proof we denote $|T|$ as t, and without loss of generality we assume $T = \{1, \ldots, t\}$.

$\mathsf{AN}(x^T) = 1$ as for any $i \in T$, $1 + x_i$ is an annihilator of x^T, and the constant function 1 does not annihilates x^T. Note that t independent annihilators can be created this way (1 for each variable indexed by T), reaching the maximal dimension (of a vectorial space generated by the degree 1 monomials in t variables), therefore $\mathcal{D}\mathsf{AN}(x^T) = t$.

For $1 + x_1$, (a case where $m_1 = 1$) the constant function 1 is not an annihilator contrarily to x_1. Then $\mathsf{AN}(1{+}x_1)$=1 and $\mathcal{D}\mathsf{AN}(1{+}x_1) = 1$. For $t > 1$, note than $1{+}x^T$ cannot be annihilated by a non null function of degree lesser than t. Indeed, let g be non null function of degree lesser than t, then $g \cdot (1 + x^T) = g$ or $g \cdot (1 + x^T) = g + x^T$, as each monomial x^I in the ANF of g is such that $I \subsetneq T$, implying that $x^I \cdot x^T = x^T$, then none of these products is null. Therefore for $t > 1$ it gives $\mathsf{AN}(1 + x^T) = t$, and as $\mathsf{AN}(x^T) = 1$, it finally gives $\mathcal{D}\mathsf{AN}(1 + x^T) = 0$.

Concluding this part, when f is a DSM whose associated direct sum vector has only one non null coefficient, which is moreover equal to one then $\mathcal{D}\mathsf{AN}(f) \leq t$, and the upper bound is reach when the constant coefficient of f is null.

Combining this result with Lemma 20 enables to get an upper bound of the $\mathcal{D}\mathsf{AN}$ of any DSM, by recursively adding monomials of equal or greater degree, one by one, initializing the recursion on a monomial function (or 1 plus this function). In the following, we generalize this idea to be able to determine the $\mathcal{D}\mathsf{AN}$ when we recursively add all the monomials of a fixed degree, in order to get an upper bound on the $\mathcal{D}\mathsf{AN}$ of a DSM of degree k by considering only k functions rather than $\sum_{i=1}^{k} m_i$. Therefore we consider a recursion with direct sum of homogeneous DSM functions, and the initialization step requires to determine the $\mathcal{D}\mathsf{AN}$ of homogeneous functions (or its complement).

On the $\mathcal{D}\mathsf{AN}$ of homogeneous DSM:

Let f be a homogeneous DSM of degree k (or its complement), its associated vector is $\mathbf{m}_f = [0, \ldots, 0, m_k]$, where $m_k > 0$. f can be constructed recursively by adding degree k monomials to x^T (or $1 + x^T$), where $|T| = t = k$.

To study the parameters of these functions used in the recursion, we denote f_i the function such that the k-th coefficient of $\mathbf{m}_{f_i}$ is equal to i, all other coefficients being zeros. Using Theorem 1 we obtain $\mathsf{AI}(f_i) = \min\{i, k\}$, we combine it with Lemma 20 for various values of i, (with $i > 0$):

- Case $1 \leq i < k - 1$:
  $\mathsf{AI}(f_i) = i < k - 1$, it corresponds to the third item of the lemma, giving $\mathcal{D}\mathsf{AN}(f_{i+1}) \leq k \cdot \mathcal{D}\mathsf{AN}(f_i)$. As $\mathcal{D}\mathsf{AN}(f_1) \leq t$ by the previous part of the proof, an immediate recursion gives:
  $$\mathcal{D}\mathsf{AN}(f_j) \leq k^j, \quad \text{for } j \mid 1 \leq j \leq k - 1.$$
  Note that the upper bound is reached when $\mathsf{AN}(f_i) = \mathsf{AI}(f_i)$ (for example when the constant coefficient is null).
- Case $i = k - 1$:
  $\mathsf{AI}(f_i) = k - 1$, it corresponds to the second item of Lemma 20, then $\mathcal{D}\mathsf{AN}(f_{i+1}) \leq 1 + k \cdot \mathcal{D}\mathsf{AN}(f_i)$, so as $i > 0$:
  $$\mathcal{D}\mathsf{AN}(f_k) \leq 1 + k^k, \quad \text{for } k > 1.$$
  Note that the upper bound is reached when $\mathsf{AN}(f_k) = \mathsf{AI}(f_k)$.
- Case $i \geq k$:
  $\mathsf{AI}(f_i) = k$, it corresponds to the first item of the lemma, then $\mathcal{D}\mathsf{AN}(f_{i+1}) = 1$, so:
  $$\mathcal{D}\mathsf{AN}(f_j) = 1, \quad \text{for } j \geq k + 1.$$

Note that the special case $i = 1 = k$ is not tackled by these cases, it corresponds to the monomial function x_1 or its complement, treated in the previous part.

This disjunction of cases gives an upper bound on the $\mathcal{D}\mathsf{AN}$ of any DSM with exactly one coefficient not null in its associated direct sum vector. It is the starting point of a recursion consisting in adding homogeneous DSM of increasing degree to get (an upper bound on) the $\mathcal{D}\mathsf{AN}$ of a DSM based on its direct sum vector. In the following part of the proof we tweak Lemma 20 to take care of these additions, using the same ideas as in this part.

Tweaking Lemma 20 for homogeneous DSM:

We prove the following:

Let f be a DSM of degree $k > 0$, and g be an Homogeneous DSM of degree t such that $t > k$ with associated vector $\mathbf{m}_g = [0, \dots, 0, m_t]$, the direct sum $f + g$ has the following property:

$$\mathcal{D}\mathsf{AN}(f+g) \leq \begin{cases} 1 & \text{if } \mathsf{AI}(f) + m_t > t, \\ 1 + t^{m_t} \cdot \mathcal{D}\mathsf{AN}(f) & \text{if } \mathsf{AI}(f) + m_t = t, \\ t^{m_t} \cdot \mathcal{D}\mathsf{AN}(f) & \text{if } \mathsf{AI}(f) + m_t < t. \end{cases}$$

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

Let denote g_i the function such that the t-th coefficient of $\mathbf{m}_{g_i}$ is equal to i, all other coefficients being zeros. Similarly to the precedent part of the proof we consider $\mathsf{AI}(f + g_i)$ using Theorem 1:

$$\mathsf{AI}(f + g_i) = \min\{\mathsf{AI}(f) + i, t\}.$$

Depending on the value of i, Lemma 20 enables to determine $\mathcal{D}\mathsf{AN}(f + g_i)$. The reasoning for $\mathcal{D}\mathsf{AN}(f_i)$ in the previous part applies to $\mathcal{D}\mathsf{AN}(f + g_i)$, with k replaced by $t - \mathsf{AI}(f)$ here.

- Case $1 \leq i < t - 1 - \mathsf{AI}(f)$:
  $\mathsf{AI}(f+g_i) < t-1$, it corresponds to the third item of the lemma, giving $\mathcal{D}\mathsf{AN}(f+g_{i+1}) \leq t \cdot \mathcal{D}\mathsf{AN}(f+g_i)$. As $\mathsf{AI}(f) < t - 1$, then $\mathcal{D}\mathsf{AN}(f + g_1) \leq t \cdot \mathcal{D}\mathsf{AN}(f)$, an immediate recursion gives:

  $$\mathcal{D}\mathsf{AN}(f + g_j) \leq t^j \mathcal{D}\mathsf{AN}(f), \quad \text{for } j \mid 1 \leq j \leq t - 1 - \mathsf{AI}(f).$$

  Note that the upper bound is reached when $\mathsf{AN}(f) = \mathsf{AI}(f)$ (for example when the constant coefficient is null).
- Case $i = t - 1 - \mathsf{AI}(f)$:
  $\mathsf{AI}(f + g_i) = t - 1$, then it corresponds to the second item of Lemma 20, $\mathcal{D}\mathsf{AN}(f + g_{i+1}) \leq 1 + t \cdot \mathcal{D}\mathsf{AN}(f + g_i)$, so:

  $$\mathcal{D}\mathsf{AN}(f + g_{t-AI(f)}) \leq 1 + t^{t-\mathsf{AI}(f)}.$$

  Note that the upper bound is reached when $\mathsf{AN}(f) = \mathsf{AI}(f)$.
- Case $i \geq t - \mathsf{AI}(f)$:
  $\mathsf{AI}(f + g_i) = t$, consequently it corresponds to the first item of the lemma, then $\mathcal{D}\mathsf{AN}(f + g_{i+1}) = 1$, so:

  $$\mathcal{D}\mathsf{AN}(f + g_j) = 1, \quad \text{for } j > t - \mathsf{AI}(f).$$

Summing up gives the condition as stated at the beginning of this part. In the following we use this result to recursively compute the $\mathcal{D}\mathsf{AN}$ of any DSM, and express it directly from the direct sum vector coefficients.

Connecting $\mathcal{D}\mathsf{AN}(f)$ to $\mathbf{m}_f$:

First, for any DSM f with associated vector $\mathbf{m}_f = [m_1, \dots, m_k]$ we consider the following set:

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

We prove by recursion on the non null m_i coefficients that:

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

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

We begin with the initialization. As f is a DSM, by Definition 11 it is non constant, then at least on of the m_i coefficients is not null and the basis of the recursion is therefore an homogeneous DSM or its complement.

The case $m_1 = 1$ corresponds to the function x_1 and $1+x_1$ only, with $\mathbf{m}_f = [1]$, for which $\mathcal{D}\mathsf{AN}(f) = 1$ from the first part of the proof. In this case, $\mathsf{AI}(f) = 1 = k = m_k$, so $\mathsf{S_d}(f) = \{1\}$, and:

$$\sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{k+1} i^{m_i} = \prod_{i>1}^{2} i^{m_i} = 1,$$

which is consistent with the hypothesis.

For the case $m_1 \neq 1$, we consider homogeneous DSM of degree greater than 1 and its complement, studied in the second part of this proof, with $\mathbf{m}_f = [0, \dots, 0, m_k]$, $m_k > 0$. Note that for these functions we have:

$$d + \sum_{i>d} m_i = \begin{cases} d + m_k & \text{if } 0 \leq d < k, \\ t & \text{if } d = k. \end{cases}$$

The minimum (used later to define $\mathsf{S_d}(f)$) is therefore m_k if $d < k$ or k if $d = k$.

Then, three cases are possible for $\mathcal{D}\mathsf{AN}(f)$ using the second part of the proof, depending on the value of the non null coefficient m_k relatively to k:

– Case $1 \leq m_k \leq k-1$:
In this case $\mathcal{D}\mathsf{AN}(f) \leq k^{m_k}$, it corresponds to $\mathsf{AI}(f) = m_k \neq k$ which implies $\mathsf{S_d}(f) = \{0\}$. Here:

$$\sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{k+1} i^{m_i} = \prod_{i>0}^{k+1} i^{m_i} = k^{m_k}.$$

– Case $m_k = k$:
This setting corresponds to $\mathcal{D}\mathsf{AN}(f) \leq 1 + k^k$ (as $k > 1$), and $\mathsf{AI}(f) = m_k = k$. The condition on the algebraic immunity gives $\mathsf{S_d}(f) = \{0, k\}$ (implicitly using Theorem 1 as each time we consider this set). Then:

$$\sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{k+1} i^{m_i} = \prod_{i>0}^{k+1} i^{m_i} + \prod_{i>k}^{k+1} i^{m_i} = k^k + 1.$$

– Case $m_k > k$:
In this case $\mathcal{D}\mathsf{AN}(f) = 1$, it corresponds to $\mathsf{AI}(f) = k \neq m_k$ which implies $\mathsf{S_d}(f) = \{k\}$, and then:

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

These three cases are consistent with the hypothesis, which concludes the initialization of the recursion.

Then, we consider the recursion step: let f be a DSM with $\ell + 1$ non null m_i coefficients, f can always be written as a direct sum of g and h such that:

– g is a DSM with $\ell \geq 1$ non null direct sum vector coefficients, with $\mathbf{m}_g = [m'_1, \dots, m'_k]$
– h is an homogeneous DSM of degree $t > k$.

We denote $\mathbf{m}_f = \mathbf{m}_{g+h}$ as $[m_1, \dots, m_t]$, note that by construction:

$$m_i = \begin{cases} m'_i & \text{for } 1 \le i \le k, \\ 0 & \text{for } k < i < t, \\ m_t > 0 & \text{for } i = t. \end{cases}$$

Applying the recursion hypothesis on the non constant DSM function g we use the third part of the proof (tweaked Lemma 20) to study $\mathcal{D}\mathsf{AN}(f)$. First, let consider the potential sets $\mathsf{S_d}(f)$, using Theorem 1, we get:

$$\mathsf{AI}(f) = \min_{0 \le d \le t} \left\{ d + \sum_{i>d}^{t} m_i \right\} = \min \left\{ \min_{0 \le d \le k} \left(d + \left(\sum_{i>d}^{k} m'_i \right) + m_t \right), t \right\} = \min \left\{ \mathsf{AI}(g) + m_t, t \right\}.$$

The values of d giving $\mathsf{AI}(f) + m_t$ are the one such that $d + \sum_{i>d}^{k} m'_i = \mathsf{AI}(f)$, which constitute $\mathsf{S_d}(g)$ when $m_1 \neq 1$, and withdrawing 1 it constitute $\mathsf{S_d}(g)$ when $m_1 = 1$. By construction $m_1 = m'_1$, then the potential sets for $\mathsf{S_d}(f)$ are $\mathsf{S_d}(g)$, $\mathsf{S_d}(g) \cup \{t\}$, or $\{t\}$, depending on the relation between $\mathsf{AI}(g) + m_t$ and t.

We study the three cases for $\mathcal{D}\mathsf{AN}(f)$ using the tweaked lemma, also depending on the relation between $\mathsf{AI}(g) + m_t$ and t:

- Case $\mathsf{AI}(g) + m_t < t$:
  In this case, the tweaked version of Lemma 20 gives $\mathcal{D}\mathsf{AN}(f) = \mathcal{D}\mathsf{AN}(g + h) \le t^{m_t} \mathcal{D}\mathsf{AN}(g)$. As in this case $\mathsf{AI}(f) = \mathsf{AI}(g) + m_t \neq t$, it implies $\mathsf{S_d}(f) = \mathsf{S_d}(g)$, and then:

  $$\sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{t+1} i^{m_i} = \sum_{d \in \mathsf{S_d}(g)} \prod_{i>d}^{t+1} i^{m_i} = t^{m_t} \left(\sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{k+1} i^{m'_i} \right).$$

  By the recursion hypothesis, this last sum is an upper bound of $\mathcal{D}\mathsf{AN}(g)$, so the hypothesis is valid at this step.
- Case $\mathsf{AI}(g) + m_t = t$:
  This setting corresponds to $\mathcal{D}\mathsf{AN}(f) = \mathcal{D}\mathsf{AN}(g+h) \le 1 + t^{m_t} \mathcal{D}\mathsf{AN}(g)$, and to $\mathsf{AI}(f) = \mathsf{AI}(g) + m_t = t$, then $\mathsf{S_d}(f) = \mathsf{S_d}(g) \cup \{t\}$. We consider the sum indexed by $\mathsf{S_d}(f)$:

  $$\begin{aligned} \sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{t+1} i^{m_i} &= \left(\sum_{d \in \mathsf{S_d}(f) \setminus t} \prod_{i>d}^{t+1} i^{m_i} \right) + \prod_{i>t}^{t+1} i^{m_i} \\ &= t^{m_t} \left(\sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{k+1} i^{m'_i} \right) + 1. \end{aligned}$$

  By the recursion hypothesis, the last sum is an upper bound of $\mathcal{D}\mathsf{AN}(g)$, so the hypothesis is valid at this step.
- Case $\mathsf{AI}(g) + m_t > t$:
  In this case $\mathcal{D}\mathsf{AN}(f) = \mathcal{D}\mathsf{AN}(g+h) = 1$, it corresponds to $\mathsf{AI}(f) = t \neq \mathsf{AI}(g) + m_t$, then $\mathsf{S_d}(f) = \{t\}$. Therefore:

  $$\sum_{d \in \mathsf{S_d}(f)} \prod_{i>d}^{t+1} i^{m_i} = \prod_{i>t}^{t+1} i^{m_i} = 1.$$

In conclusion, for the three cases we proved that:

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

Note that the property on the upper bound being reached in all cases when $\mathsf{AI}(f) = \mathsf{AN}(f)$ comes directly from this property in Lemma 20. It finishes the recursion step and therefore concludes the proof.

□

This formula gives a tight upper bound on the dimension of the annihilators of a DSM, and it corresponds to item 4 of Theorem **??**. In the following remark we give some intuition on the shape of these annihilators.

Remark 6. The relation between the dimension of the annihilator space of f of degree less than or equal to $\mathsf{AI}(f)$ and the set $\mathsf{S_d}(f)$ gives more intuition on these annihilators. We consider the case here of f being a DSM such that the constant coefficient is null. From the proof of Theorem 1 (lower bound part) we know that these functions are such that $\mathsf{AN}(f) = \mathsf{AI}(f)$ as we can construct non null annihilators of degree $\mathsf{AI}(f)$. These annihilators are built considering $d \in \mathsf{S_d}(f)$ and f as $f_1 + f_2$ where f_1 consists in the monomials of degree less than or equal to d of f, and f_2 is the part with the other monomials. $1 + f_1$ is an annihilator of f_1, and taking one variable of each monomial of f_2, the product of the complement of each of these variables is an annihilator of f_2. Then, $\prod_{i>d}^{k} i^{m_i}$ linearly independent annihilators can be created this way.

Note that in the formula of Theorem 4 the products runs until $k+1$ where k is the degree of the considered function, this extra term in the product aims to get the " $+\ 1$" term occurring only when $k \in \mathsf{S_d}(f)$. The variation of definition for $\mathsf{S_d}(f)$ depending on the value of m_1 can also been remarked with these annihilators. Indeed, when $m_1 = 1$, both 0 and 1 can fulfill the formula $\mathsf{AI}(f) = d + \sum_{d>1} m_i$, and in this case, the same annihilators are produced, as the degree-exactly-1 component function of f is a monomial function then annihilated by $1+x_1$ both for $d = 0$ (canceling the monomial x_1 in f_2) and $d = 1$ (annihilating f_1 with its complement). In all the other cases, by construction two different d lead to linearly independent annihilators of f.

Finally, a counting argument shows directly that for these functions all the annihilators of degree less than or equal to $\mathsf{AI}(f)$ are linear combination of the annihilators described in this remark, it characterize the annihilators of minimal degree of all DSM with null constant coefficient.

7 Error-growth with the Third Generation of FHE.

In this section we investigate the error-growth involved when DSM or XOR-MAJ functions are homomorphically evaluated with an homomorphic encryption scheme of the third generation. Then, it directly determines the noise produced by an IFP instantiated with such functions in the hybrid fully homomorphic framework.

The third generation of FHE have been introduced in [GSW13] based on the idea of approximate eigenvector, we refer to this article and to [AP14] for formal descriptions and security proofs. As we focus on the error-growth given by some functions, we consider a general type of GSW-like encryption scheme. We defined this scheme only with the properties of the homomorphic addition and multiplication, in order to be general enough to apply for all variants. It could then be more particularly applied to different cases *e.g.* the initial GSW scheme, multiple bits variant [HAO15], ring versions [GSW13, KGV14].

Definition 23 (GSW-like Scheme). *We call GSW-like scheme an encryption scheme such that each valid ciphertext $\mathbf{C}_i$ (relatively to the secret $\mathbf{s}$) has error component $\mathbf{e}_i$ which coefficients following a subgaussian parameter of σ_i, and the following applies for the homomorphic operations:*

- $H.\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2) : \mathbf{C}_+ = \mathbf{C}_1 + \mathbf{C}_2$.
- $H.\mathsf{Mul}(\mathbf{C}_1, \mathbf{C}_2) : \mathbf{C}_\times = \mathbf{C}_1 \times \mathbf{G}^{-1}\mathbf{C}_2$, *where* $\mathbf{G}^{-1}$ *is a function such that* $\mathbf{G}\mathbf{G}^{-1}(\mathbf{C}) = \mathbf{C}$ *for all* $\mathbf{C}$ *and the values of* $\mathbf{G}^{-1}(\mathbf{C})$ *follow a subgaussian distribution with constant parameter.* $\mathbf{G}$ *corresponds to the gadget matrix([MP12]).*
- *Let* $\mathbf{C}_f = H.\mathsf{Add}(\mathbf{C}_i$*, for* $1 \leq i \leq k)$*, then* $\mathbf{e}_f$ *the related error follows a subgaussian distribution with parameter* σ' *such that:*

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

- *Let* $\mathbf{C}_f$ *be the result of a multiplicative homomorphic chain:*

$$\mathbf{C}_f = H.\mathsf{Mul}(\mathbf{C}_1, H.\mathsf{Mul}(\mathbf{C}_2, H.\mathsf{Mul}(\cdots, H.\mathsf{Mul}(\mathbf{C}_k, \mathbf{G})))),$$

  and $\mathbf{e}_f$ *is the corresponding error with subgaussian parameter* σ' *such that:*

$$\sigma' = \mathcal{O}\left(y\sqrt{\sigma_1^2 + \sum_{i=2}^{k}\left(\sigma_i \Pi_{j=1}^{i-1} Norm(m_j)\right)^2}\right),$$

  where y *is a constant depending on the ring and the norm also depends on the ring.*

With these definitions and error-growth of additions and multiplication we can derive the error-growth related to DSM and XOR-MAJ. First we study the error-growth relatively to combs (particular products) and MUX gates. Then, we analyze the case of DSM functions. Finally we investigate the error-growth relatively to XOR-MAJ functions.

7.1 Error-growth of Combs and MUX Gates.

Error-growth in H.Comb We use the notion of comb when we consider a sequential multiplication of ciphertext with bounded noise. This kind of multiplication enables to maintain a low noise when nonlinear functions with a sparse ANF are evaluated with a GSW-like scheme.

Definition 24 (Homomorphic Comb $H.\mathsf{Comb}$). *Let* $\mathbf{C}_1, \cdots, \mathbf{C}_k$ *be* k *ciphertexts from a GSW-like scheme with error coefficients from independent distributions with same subgaussian parameter* σ*. We define* $H.\mathsf{Comb}(y, \sigma, c, k) = H.\mathsf{Mul}(\mathbf{C}_1, \cdots, \mathbf{C}_k, \mathbf{G})$ *where:*

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

and $\mathbf{C}_{comb} = H.\mathsf{Comb}(y, \sigma, c, k)$ *has error components following a subgaussian distribution of parameter* $\mathcal{O}(\sigma_{comb})$.

Lemma 21 (Comb Error-growth, σ_{comb} Quantity). *Let* $\mathbf{C}_1, \cdots, \mathbf{C}_k$ *be* k *ciphertexts of a GSW-like scheme with same error parameter* σ *and* $\mathbf{C}_{comb} = H.\mathsf{Comb}(y, \sigma, c, k)$*. Then we have:*

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

Proof. Using the property on the homomorphic multiplication we obtain:

$$\sigma_{comb} = y\sqrt{\sigma^2 + \sum_{i=2}^{k}(\sigma \Pi_{j=1}^{i-1} Norm(m_j))^2} = y\sqrt{\sigma^2 + \sum_{i=2}^{k}(\sigma c^{i-1})^2},$$
$$= \sigma\sqrt{\sum_{i=1}^{k}(c^{i-1})^2} = \sigma_{comb} = y\sigma c_k.$$

□

Note that when the value of the plaintext is kept in $\{-1, 0, 1\}$, it leads to to $c = 1$ and therefore $c_k = \sqrt{k}$.

Error-growth in MUX gates MUX gates are well adapted to third generation error-growth, as observed in the context of branching programs [BV14], or in the context of deterministic automata [CGGI16]. The asymmetric error-growth of the third generation enables to produce low-noise ciphertexts when a MUX gate or a combination of MUX gates are evaluated, as the final error depends only on the errors from the ciphertext of the control bit and only one of the two other errors.

Lemma 22 (MUX Error-growth).
Let $\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d$ *be* 3 *GSW-like ciphertexts with error parameter* $\sigma_a, \sigma_b, \sigma_d$, *and* $\sigma_{max} = \max\{\sigma_a, \sigma_b\}$, *and* y *the constant from the ring. Defining the MUX ciphertext as* $\mathbf{C}_{MUX} = H.\mathsf{Add}(H.\mathsf{Mul}(\mathbf{C}_d, \mathbf{C}_a - \mathbf{C}_b), \mathbf{C}_b)$, *we have:*

$$\sigma_{MUX} = \mathcal{O}\left(\sqrt{y^2\sigma_d^2 + \sigma_{max}^2}\right).$$

Proof.

$$\begin{aligned}
\mathbf{C}_{MUX} &= \mathbf{C}_d\mathbf{G}^{-1}(\mathbf{C}_a - \mathbf{C}_b) + \mathbf{C}_b,\\
\mathbf{s}^\top\mathbf{C}_{MUX} &= \mathbf{s}^\top\mathbf{C}_d\mathbf{G}^{-1}(\mathbf{C}_a - \mathbf{C}_b) + \mathbf{s}^\top\mathbf{C}_b,\\
&= \mathbf{e}_d^\top\mathbf{G}^{-1}(\mathbf{C}_a - \mathbf{C}_b) + m_d\mathbf{s}^\top(\mathbf{C}_a - \mathbf{C}_b) + \mathbf{e}_b^\top + m_b\mathbf{s}^\top\mathbf{G},\\
&= \mathbf{e}_d^\top\mathbf{G}^{-1}(\mathbf{C}_a - \mathbf{C}_b) + m_d(\mathbf{e}_a^\top + \mathbf{e}_b^\top) + m_d(m_a - m_b)\mathbf{s}^\top\mathbf{G} + \mathbf{e}_b^\top + m_b\mathbf{s}^\top\mathbf{G}.
\end{aligned}$$

We obtain two cases depending on the value of d:

- If $m_d = 0$ then: $\mathbf{s}^\top\mathbf{C}_{MUX} = \mathbf{e}_d^\top\mathbf{G}^{-1}(\mathbf{C}_a - \mathbf{C}_b) + \mathbf{e}_b^\top + m_b\mathbf{s}^\top\mathbf{G}$.
  In term of errors, the first part has an error parameter of $y\sigma_d$ from the formula of the homomorphic multiplication, which gives a total error of $\sigma_{MUX} = \mathcal{O}\left(\sqrt{y^2\sigma_d^2 + \sigma_b^2}\right)$.
- If $m_d = 1$ then: $\mathbf{s}^\top\mathbf{C}_{MUX} = \mathbf{e}_d^\top\mathbf{G}^{-1}(\mathbf{C}_a - \mathbf{C}_b) + \mathbf{e}_a^\top + m_a\mathbf{s}^\top\mathbf{G}$.
  With the same reasoning we get a total error of $\sigma_{MUX} = \mathcal{O}\left(\sqrt{y^2\sigma_d^2 + \sigma_a^2}\right)$.

As d is a control bit, only these two cases can happen, giving the final formula.

□

Note that the formula in this lemma gives a final error parameter which depends on the error parameter of the 3 ciphertexts (due to the max), and not only 2, as it relates to an upper bound to tackle worst case scenarii rather than the exact value which depends on two of the 3 ciphertexts only.

7.2 Error-growth of DSM Functions.

We determine the error growth involved by the evaluation of a DSM function.

Lemma 23 (DSM Error-growth).

Let F be the DSM in n variables with associated direct sum vector $\mathbf{m}_F = [m_1, m_2, \cdots, m_k]$.

Assume that $\mathbf{C}_i$ for $1 \leq i \leq n$ are GSW-like ciphertexts with same subgaussian parameter σ and $c = 1$. We define $\mathbf{C}_F = H.\mathsf{Eval}(F, \mathbf{C}_i)$ the output of the homomorphic evaluation of the ciphertexts $\mathbf{C}_i$'s along the circuit F. Then the error parameter σ' is:

$$\sigma' = \mathcal{O}\left(\sigma\sqrt{m_1 + y^2\left(\sum_{i=1}^{k} i \cdot m_i\right)}\right) \approx \mathcal{O}\left(\sigma y \sqrt{n}\right).$$

Proof. We first evaluate the error given by the monomials of a given degree i, that we denote σ_i. σ_1 corresponding to the error of the linear part, using the formula relatively to homomorphic addition we obtain $\sigma_1 = \sigma\sqrt{m_1}$. Then the monomials of degree i with $2 \leq i \leq d$ are evaluated as product of $i + 1$ ciphertexts, giving:

$$H.\mathsf{Mul}(\mathbf{C}_j, \cdots, \mathbf{C}_{j+i-1}, \mathbf{G}) = H.\mathsf{Comb}(y, \sigma, 1, i),$$

with an error with subgaussian parameter $\mathcal{O}(y\sigma\sqrt{i})$. Adding all the ciphertexts related to the same degree we get:

$$\sigma_i = \mathcal{O}(y\sigma\sqrt{i}\sqrt{m_i}) = \mathcal{O}(y\sigma\sqrt{i \cdot m_i}).$$

Adding all these ciphertexts gives $\mathbf{C}_F$ with error parameter:

$$\mathcal{O}\left(\sigma\sqrt{m_1 + y^2\left(\sum_{i=2}^{k} i \cdot m_i\right)}\right).$$

By definition of the direct sum vector: $\sum_{i=1}^{k} i \cdot m_i \leq n$ (the equality corresponding to the case where all variables appear in the ANF of F), giving the final result:

$$\sigma' \approx \mathcal{O}\left(\sigma y \sqrt{n}\right).$$

□

7.3 Error-growth of XOR-MAJ Functions.

In this part we focus on how to evaluate the majority function without producing an important error-growth with GSW-like ciphertexts. Threshold functions can be evaluated in various ways, which is promising relatively to homomorphic evaluation. First, functions $\mathsf{XOR}_k\mathsf{MAJ}_{2^{j+1}-1}$ have degree 2^j, which already gives an idea of their evaluation relatively to the second generation. Having a closer look on their ANF, the number of monomials is exponential (all monomials of degree 2^j), which seems prohibitive for 3G evaluation. Nevertheless, evaluating branching programs [BV14] or finite automata [CGGI16] has been shown to be very promising with the 3G, then, evaluating the majority function with MUX gates (multiplexers) rather than based on the ANF representation can still give a very low error-growth.

First we explain an evaluation 'in clear' of the majority function, then we study the error-growth it can generate. Finally, using the formula of homomorphic addition enable to derive the final noise of a XOR-MAJ function.

First, let us consider a branching program for the function majority on n bits $\mathsf{T}_{d,2d-1}$, described in Figure 3. Barrington's theorem proves the existence of a width 5, polynomial length branching program for majority; here we focus on a circuit whose homomorphic evaluation with a GSW-like FHE produces a small error-growth. Therefore we consider a branching program of $n+1$ layers, where each transition from layer i to $i+1$ is indexed by the variable x_i, each dashed vertical arrow corresponds to the value 0 of the variable and each diagonal arrow corresponds to the value 1. The final result is 0 if the path ends in the left half of the last layer, 1 otherwise. The idea behind this circuit is to force a path to finish in the right half when at least d variables are equal to 1.

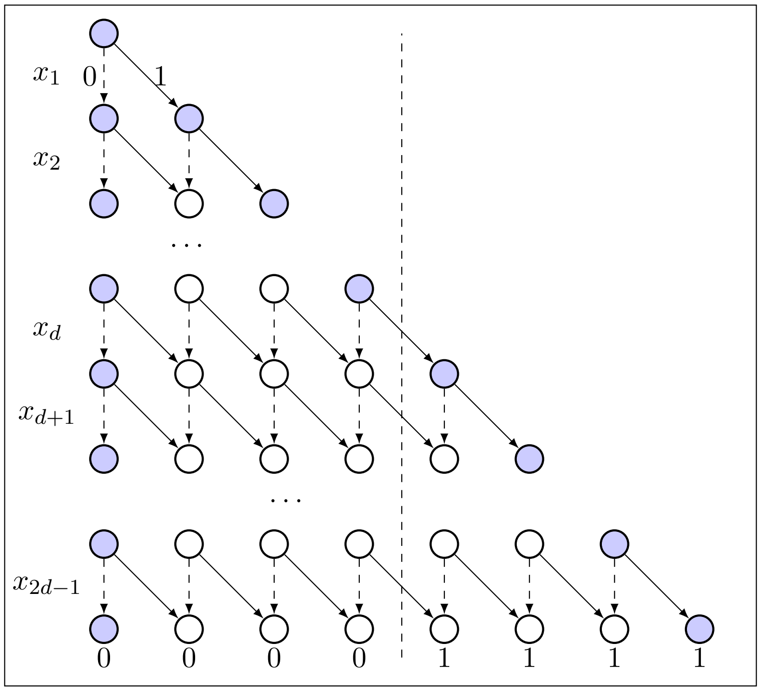

Fig. 3. Branching program for majority.

Proposition 3 (Branching Program for Majority (informal)). *Let $\mathcal{B}_{2n+1}$ be the branching program of $2n+2$ layers and $\frac{(2n+2)(2n+3)}{2}$ nodes described in Figure 3, then $\mathcal{B}_{2n+1}$ computes the majority function in $2n+1$ bits.*

Proof. First, the circuit is well defined, for all nodes not in the terminal layer there is a transition for the two potential values of the variable associated to the layer. Then, each variable is used once and only once: the variable i is used between layers i and $i+1$. Each transition is either leading to the same position in the following layer, or leading to the successive (right) position in the following layer. For all layers the transition keeps the position if and only if the variable is equal to 0. As the first layer begins at the leftmost position, by induction, the end node of the path (in the final layer) is in the right half if and only if at least d variables took the value 1. Finally the result of $\mathcal{B}_n$ on the entry $(x_1, \cdots, x_n)$ is 1 if and only if $\mathsf{w_H}(x_1, \cdots, x_n) \geq d$ which is the definition of Maj_n. $\square$

Evaluating this branching program in clear, the final node reached gives the result: 0 if we reached the left part, 1 otherwise. Considering homomorphic evaluation we want to get a unique ciphertext, encrypting this binary result. We modify this branching program to get a circuit with gates AND, MUX, XOR easy to homomorphically evaluate, and with a unique final node. We use a standard technique using the truth table of a function, the result can be obtained by summing the results of all the paths leading to 1 as only one path corresponds to the n bits entry really evaluated, if the entry corresponds to one of these paths then the result is 1 otherwise it is 0. Therefore, homomorphically at each layer a transition indexed by x_i is replaced by an homomorphic product by $\mathbf{C}_i$ for the value 1 and by $\bar{\mathbf{C}}_i = \mathbf{G} - \mathbf{C}_i$ for the value 0.

The branching program is modified in the following way: all the transitions leading to final value 0 are cut, therefore all the bottom left part is deleted, and all the nodes for which all future transitions lead to 1 are merged: all the paths of the bottom right part are compressed, using additions. It gives the circuit to homomorphically evaluate, given in Figure 4. To deal with additions we modify the representation, the nodes in red are not computed as a MUX, but as a sum. The first summand is the product of the left parent node by the ciphertext of this layer, the second summand is the right parent node. Black arrows, dashed or not, symbolize the product by the ciphertext of the corresponding layer (its complementary for a dashed arrow), and a red arrow symbolizes the addition. The color of the nodes symbolizes how they are computed, a white node corresponds to a MUX, a red one to an addition, and a blue one to an AND or nothing.

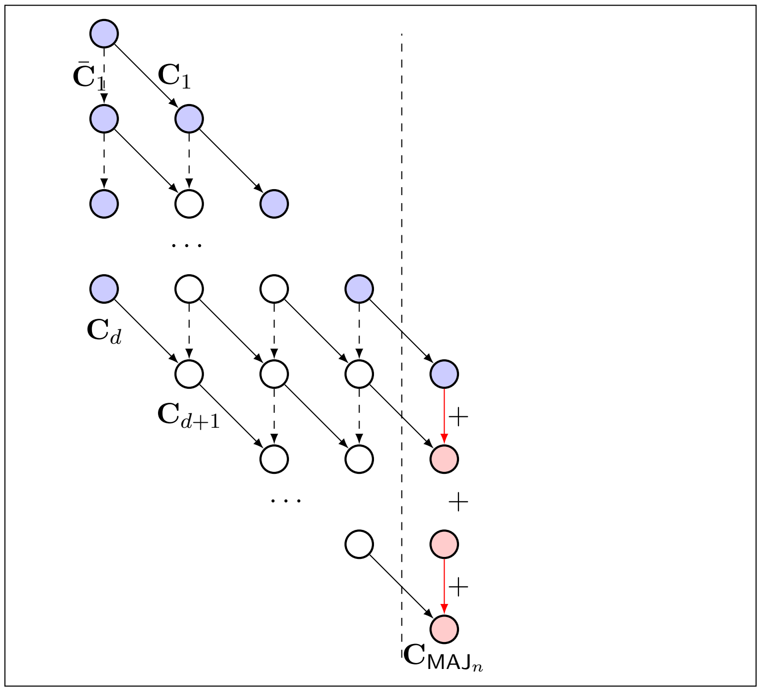

Fig. 4. Homomorphic circuit for majority.

From this circuit we can compute the standard deviation of the ciphertext obtained by evaluating the majority function MAJ_n.

Lemma 24. *Let $\mathbf{C}_i$ for $0 \leq i \leq n$ be ciphertexts of a GSW-like scheme with same subgaussian parameter σ and $c = 1$. We define $\mathbf{C}_{\mathsf{MAJ}_n}$ the output of the homomorphic evaluation of the ciphertexts $\mathbf{C}_i$'s along the* *circuit of Figure 4. Then the error parameter σ' associated to $\mathbf{C}_{\mathsf{MAJ}_n}$ is:*

$$\sigma' = \mathcal{O}\left(y\sigma n\sqrt{n}\right).$$

Proof. We decompose the circuit in three parts to do the proof; the part of blue nodes, white nodes and red nodes on Figure 4. The blue nodes correspond to ciphertexts obtained by a chain of multiplications of freshly encrypted ciphertexts *i.e.* homomorphic comb, the white nodes are the output of a MUX gate and the red nodes are the output of an addition.

We first prove that the ciphertext of a blue or white node of the layer i has an error parameter of $\mathcal{O}(y\sqrt{i})$. Let us focus on the blue nodes; the ciphertext obtained at the layer i is the product of i freshly encrypted ciphertexts with error parameter σ, obtained from the ciphertexts $\bar{\mathbf{C}}_j$ (with $1 \leq j \leq i$) for the left part of the parallelogram or from the ciphertexts $\mathbf{C}_j$ (with $1 \leq j \leq i$) for the right part of the parallelogram. From Lemma 21 the associated error-growth is therefore:

$$H.\mathsf{Comb}(y, \sigma, 1, i) = y\sigma\sqrt{i}.$$

Then we prove by induction that the ciphertext of a blue or white node of the layer k has the following error parameter:

$$\mathcal{O}(y\sigma\sqrt{k}),\ \text{for layer } 1 \leq k \leq n-1.$$

- $k = 1$, this layer has only two nodes, both blue, corresponding to the ciphertexts $\bar{\mathbf{C}}_1$ and $\mathbf{C}_1$. As $\bar{\mathbf{C}}_1 = \mathbf{G} - \mathbf{C}_1$, the associated error parameter is the same as the one of $\mathbf{C}_1$: $\sigma = \mathcal{O}(y\sigma\sqrt{1})$ validating the initialization (Note that the constant y appears naturally if we begin the initialization at step $k = 2$).
- $k \to k+1$, the layer $k+1$ has at least one white node. If the ciphertext corresponds to a blue node, the associate error has parameter $\mathcal{O}(y\sqrt{k+1})$ as previously proved. Otherwise, a ciphertext corresponding to a white node is obtained by a MUX gate with input two ciphertexts from the precedent layer and the control bit encrypted by $\mathbf{C}_i$. Then using Lemma 22 together with the induction hypothesis, the associated error parameter is:
$$\mathcal{O}(\sqrt{y^2\sigma^2 + \max\{y^2\sigma^2 k, y^2\sigma^2 k\}}) = \mathcal{O}(y\sigma\sqrt{k+1}),$$
proving the induction.

Note that this property also applies for the layer 0, but as for the layer 1 it serves more for notation (and for understanding the principle of the circuit) than for the final result.

The remaining part of the proof concerns the red nodes, which are the one adding two inputs; one of them (left) being the product of a cipher from the precedent level by $\mathbf{C}_i$, and the other (right) being a ciphertext of the precedent level. Note that the two summands may have not independent error terms, as they have been computed from the same ciphertexts (or minus the matrix $\mathbf{G}$), and that contrarily to the MUX gates, additions does not ensure error independence. Then we prove by induction that the ciphertext corresponding to the red node of the layer k has the following error parameter:

$$\mathcal{O}(y\sigma\sum_{i=d}^{k}\sqrt{i}),\ \text{for layer } d+1 \leq k \leq n.$$

- $k = d+1$, the parent nodes are a blue and a white node of the layer d, so corresponding to ciphertexts with error parameter $\mathcal{O}(y\sigma\sqrt{d})$. One of the two ciphertexts (left, white) is multiplied by $\mathbf{C}_{d+1}$, giving an error parameter of $\mathcal{O}(y\sigma\sqrt{d+1})$. The error parameter (recall that it is a standard deviation) of the sum is upper bounded by the sum of the two error parameters as the distributions can be correlated, giving an error parameter of:

$$\mathcal{O}(y\sigma(\sqrt{d}+\sqrt{d+1})),$$

validating the initialization.

– $k \rightarrow k+1$, the parent nodes are a white node and the red node of the layer k. The ciphertext corresponding to the white node of layer k is multiplied by $\mathbf{C}_{k+1}$, giving an error parameter of:

$$\mathcal{O}(y\sigma\sqrt{k+1}).$$

By the induction hypothesis, the other ciphertext has associated error parameter of:

$$\mathcal{O}(y\sigma\sum_{i=d}^{k}\sqrt{i}).$$

As the error of these two ciphertexts may be correlated, performing the sum we obtain an error parameter of:

$$\mathcal{O}(y\sigma\sum_{i=d}^{k+1}\sqrt{i}),$$

proving the induction.

The red node of the layer n corresponding to $\mathbf{C}_{Maj}$, we get that the final error is:

$$\mathcal{O}(y\sigma\sum_{i=d}^{n}\sqrt{i}),$$

and as

$$\sum_{i=d}^{n}\sqrt{i} \leq d\sqrt{n},$$

we obtain the final result of the lemma:

$$\sigma' = \mathcal{O}(y\sigma d\sqrt{n})$$

□

This result shows that the majority function can be used in an homomorphic framework using a GSW-like FHE as the homomorphic error-growth involved is quite small (for a function in n variables it is proportional to $n^{1.5}$). Using a randomization technique it can be even more reduced (proportional to $n^{0.5}$). To do so, we avoid the use of additions which obliges to consider the sum of errors and we use a circuit of bigger size but using only AND and MUX gates.

The principle is to duplicate the circuit of Figure 3 without the part leading to 0 and to add the copy at the end of the first circuit in reverse order. This construction enables to get only one node in the final layer, and it guarantees that every path to 1 from the top circuit is by construction completed by the symmetric path (horizontal symmetry) to the final node on the bottom part. No path leading to 0 gets to the bottom circuit, and every path to 1 in the first circuit is completed by a unique path to 1 in the bottom circuit due to the symmetry. It enables to use the same technique based on the truth table of a function as for the first circuit for the homomorphic evaluation. The main difference is that the bottom circuit obliges to have new encryptions of the n plaintexts x_i, with independent errors. The new circuit is presented in Figure 5.

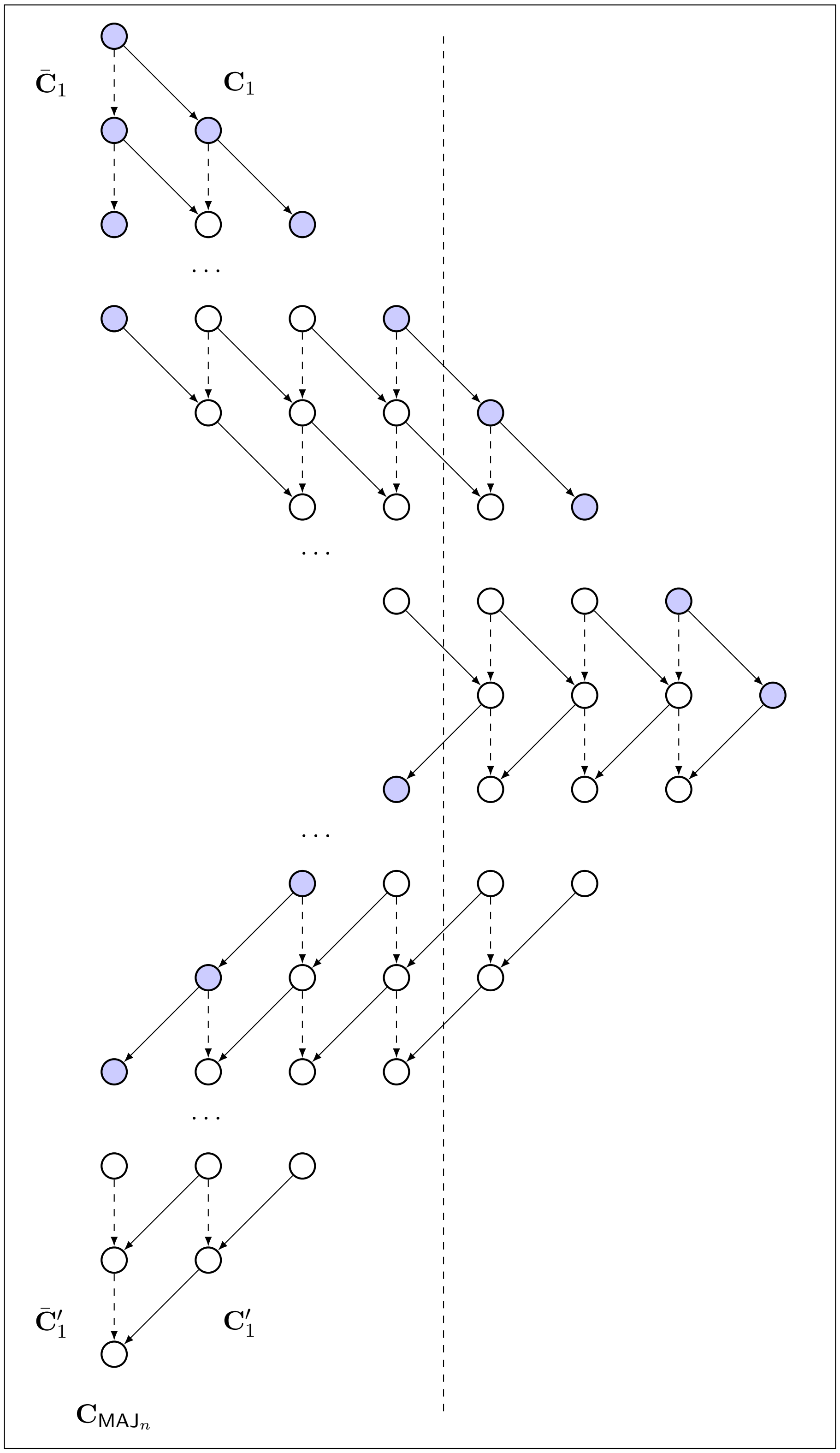

Fig. 5. Homomorphic circuit for majority, using randomization.

Note that this construction preserves the result of the homomorphic evaluation. As $x_i^2 = x_i$ in $\mathbb{F}_2$, re-use variables do not change the result. Then the final value is a sum of all potential entries of the majority function giving 1 and all corresponding paths are counted exactly once by construction (the symmetry). The final ciphertext is an encryption of the majority over the n encrypted bits. From this circuit of length $2n+1$ and size $3(d)^2$ with MUX and AND gates we can compute the standard deviation of the ciphertext obtained by evaluating the majority function MAJ_n.

Lemma 25 (Majority Error-growth).
Let $\mathbf{C}_i$ for $0 \leq i \leq n$ be ciphertexts of a GSW-like scheme with same subgaussian parameter σ and $c = 1$. Let $\mathbf{C}'_i$ for $0 \leq i \leq n$ be ciphertexts of the same plaintext as $\mathbf{C}_i$ (but independent distribution) of a GSW-like scheme with same subgaussian parameter σ and $c = 1$. We define $\mathbf{C}_{\mathsf{MAJ}_n}$ the output of the homomorphic evaluation of the ciphertexts $\mathbf{C}_i$ and $\mathbf{C}'_i$ along the circuit of Figure 5. Then the error parameter σ' associated to $\mathbf{C}_{\mathsf{MAJ}_n}$ is:

$$\sigma' = \mathcal{O}\left(y\sigma\sqrt{2n+1}\right).$$

Proof. The proof is similar to the one of Lemma 24; the circuit containing only blue and white nodes, all ciphertexts of the layer $i > 0$ is computed by a product of a ciphertext from the precedent layer with a freshly encrypted ciphertext or by a MUX gate with input two ciphertexts from the precedent layer. Then the proof by induction giving that the error parameter associated to a ciphertext of the layer i is $\mathcal{O}(y\sigma\sqrt{i})$ can here be extended to all layers, from $k = 1$ to $k = 2n+1$. Finally $\mathbf{C}_{Maj}$ being the ciphertext of the layer $2n+1$, it enables to conclude:

$$\sigma' = \mathcal{O}\left(y\sigma\sqrt{2n+1}\right).$$

□

Combining this lemma with the error-growth of the addition gives the error-growth of XOR-MAJ functions.

Corollary 4 (XOR-MAJ Error-growth). *Let $\mathbf{C}_i, \mathbf{C}'_i$ for $0 \leq i \leq n$, and $\mathbf{C}"_i$ for $i \in k$ be ciphertexts of a GSW-like scheme with same subgaussian parameter σ and $c = 1$ such that $\mathbf{C}_i, \mathbf{C}'_i$ are ciphertexts of the same plaintext as $\mathbf{C}_i$ (but independent distribution). We define $\mathbf{C}_{\mathsf{MAJ}_n}$ as the output of the homomorphic evaluation of the ciphertexts $\mathbf{C}_i$ and $\mathbf{C}'_i$ along the circuit of Figure 5. We define $\mathbf{C}_{\mathsf{XOR}_k\mathsf{MAJ}_n}$ as the output of H.*Add *performed on $\mathbf{C}_{\mathsf{MAJ}_n}$ and the $\mathbf{C}"_i$ for $i \in [k]$.Then the error parameter $\sigma_{\mathsf{XOR}_k\mathsf{MAJ}_n}$ associated to $\mathbf{C}_{\mathsf{XOR}_k\mathsf{MAJ}_n}$ is:*

$$\sigma_{\mathsf{XOR}_k\mathsf{MAJ}_n} = \mathcal{O}\left(\sigma\sqrt{k + y^2(2n+1)}\right).$$

In conclusion, the branching program approach enables to homomorphically compute the majority function with a small error-growth whereas the ANF approach leads to a different conclusion. The bigger length and size of the second circuit enables to have an error-growth similar to the one of direct sum of monomials, nevertheless it requires to perform more homomorphic gates and to randomize ciphertexts, which has an additional cost in time and in data. Note that these evaluations of the majority function can be adapted to the other threshold functions. It allows to derive the error-growth of other XOR-THR functions, or other functions obtained by combining symmetric functions.

8 Instantiating the Improved Filter Permutators with DSM Functions: $\mathsf{FiLIP}_{\mathsf{DSM}}$ Instances.

We instantiate the IFP paradigm with filtering functions being direct sums of monomials, and denote these instances as $\mathsf{FiLIP}_{\mathsf{DSM}}$. This choice is motivated by the functions considered in [MJSC16] under the name of FLIP functions, which are a sub-family of DSM functions. DSM functions are very structured functions, are easy to represent through their direct sum vector, and we can determine all their parameters relatively to Boolean criteria. Recall that to estimate the most correctly the security given by a filtering function, it is necessary to determine the parameters of all its descendants (up to λ variables). As DSM are bit-fixing stable, knowing the standard properties of all the family enables to determine the bit-fixing properties of any DSM filtering function, giving a very accurate estimation of the security against guess-and-determine attacks. Finally, it is a good choice in terms of homomorphic error-growth as shown in Section 7 for the 3G, and also for the 2G (FHE schemes such as BGV [BGV12]) due to their very small multiplicative depth.

We instantiate the forward secure PRNG following Bellare and Yee [BY03] construction, using the AES as underlying block cipher. The PRNG is set with two constants C_0 and C_1, For each K_i the first block $AES(K_i, C_0)$ gives the key K_{i+1} of the next iteration and the second block $AES(K_i, C_1)$ gives 128 bits being the i-th part of the PRG's output. For each key-stream bit the PRNG outputs $\lceil \log \binom{N}{n} \rceil$ bits used to select the subset considering the variables in lexicographic order. Then, the permutation over n bits is instantiated with the Knuth shuffle [Knu97] with the following bits output by the PRNG. Finally, n last bits are used to generate the whitening. If the number of ciphertext bits $m \leq 2^{\lambda}$ requires more pseudorandom bits that the secure limitation of the PRNG, another instance is used with other constants. The number of possible instances for the PRNG makes that for the parameters we consider the limitation comes from m.

In this section we first give in Subsection 8.1 the modifications performed on the algorithms estimating the attack complexities. The large number of different descendants of a DSM function leads to compare the parameters of different descendants to reduce the complexity of the algorithms. The proofs to determine the descendant to consider are given constitute the main part of this subsection. Then, in Subsection 8.2 we give the concrete instances chosen for a bit security of 80 or 128.

8.1 Simplifiying the Attack Complexities Algorithms for DSM Functions.

In Section 4 we give the general framework to compute the complexity of the attacks. We describe in the following the modifications made to the algorithm in order to compute security bounds more efficiently on DSM functions. First note that for the criteria of resiliency, nonlinearity, and algebraic immunity, the direct sum vector is sufficient to compute the parameter of the function. For the $\mathcal{D}\mathsf{AN}$, the constant term of the function matters, nevertheless the upper bound defined as $\max(\mathcal{D}\mathsf{AN}(f), \mathcal{D}\mathsf{AN}(f+1))$ can also be derived from $\mathbf{m}_f$. For the and fast algebraic immunity the lower bound we use is also given by the direct sum vector. Therefore, two functions with the same direct sum vector are considered as equivalent, and the number of descendants to consider decreases using this property, it corresponds to the first modification mentioned in Section 4.2.

Then, the number of descendants with different direct sum vector is still very important, and the number of different parameters also. consequently, we use the second modification relatively to the nonlinearity, the $\mathcal{D}\mathsf{AN}$ and the fast algebraic immunity. For DSM functions, the exact value of the bias δ varies a lot, hence we consider only the values of $-\lfloor(\log(\delta)\rfloor$. For the $\mathcal{D}\mathsf{AN}$ we use an upper bound, considering the maximum over all DSM of degree at most k. For fast algebraic attack we consider only 1 and 2 as possible values for e and the reached algebraic immunity as possible value for d.

Finally, we decide to not consider all descendants, and attribute the probability of the ones with good parameters to others (as called third modification in Section 4.2). It is the modification affecting most the complexity of the algorithm. It is realized through proving relations between the parameters of DSM functions and an order on their direct sum vector. In the case of DSM, the descendants obtained by fixing zeros are always with worse parameter, and we prove it in the following.

Note that a DSM function f with $\mathbf{m}_f = [m_1, m_2, \ldots, m_k]$ has at most $M = \prod_{i=1}^{k}(m_i + 1)$ different descendants obtained by fixing ℓ zeros (as fixing a variable to 0 decreases one of the m_i by 1). To compute the profiles of a DSM, we use the following representation, as only the descendants obtained by fixing zeros are considered, we store a vector of length M and each index represent one descendant. The number of descendants at each step being the most expensive part of the algorithm in term of storage and time, the algorithm is better suited for function with relatively small M. It justifies why in the next subsection instances with sparse direct sum vectors are more considered than sums of triangular functions.

Second Modification, Bounding FAI and $\mathcal{D}$AN First we define and give a lower bound on the FAI of a DSM.

Proposition 4. *Let define for any DSM function h the lower bound on the* FAI, $\mathsf{bFAI}(h)$ *such that:*

$$\mathsf{bFAI}(h) = \begin{cases} \mathsf{AI}(h) + 2 & \text{if } \mathsf{AI}(h) = \deg(h), \mathsf{AI}(h) > 1, \text{and } m_k > 1 \\ \mathsf{AI}(h) + 1 & \text{otherwise.} \end{cases}$$

Proof. From Lemma 6 we know that $\mathsf{FAI}(h) = \mathsf{AI}(h) + 2$ if $\mathsf{AI}(h) = \deg(h)$, $\mathsf{AI}(h) > 1$, and $m_k > 1$, and that for any non constant function its FAI is at least its AI plus one, justifying the lower bound. □

Then we define and give an upper bound on the $\mathcal{D}\mathsf{AN}$ of a DSM. The next proposition gives an upper bound on the $\mathcal{D}\mathsf{AN}$ of a DSM which is compatible with the covering order of direct sums vectors (detailed in the next subsection), whereas the $\mathcal{D}\mathsf{AN}$ itself is not. For example, we can take the direct sum vectors $[1,1], [1,2]$, and $[1,1,1]$. We have $[1,1] \preceq [1,2]$ and $[1,1] \preceq [1,1,1]$ but (considering the functions represented by these direct sum vectors) $\mathcal{D}\mathsf{AN}([1,2]) < \mathcal{D}\mathsf{AN}([1,1]) < \mathcal{D}\mathsf{AN}([1,1])$.

Proposition 5. *For any DSM function h of degree k:*

$$\mathcal{D}\mathsf{AN}(h) \leq \begin{cases} k^k + 1 & \text{if } m_1 = 0 \\ k^{k-1} + 1 & \text{if } m_1 > 0. \end{cases}$$

We define the following upper bound on the $\mathcal{D}\mathsf{AN}$, $\mathsf{b}\mathcal{D}\mathsf{AN}(h)$ *as* $\mathsf{b}\mathcal{D}\mathsf{AN}(h) = k^k + 1$.

Proof. First we prove the upper bound of $\mathcal{D}\mathsf{AN}(h)$ depending on the value of m_1 for any DSM h of degree k. Then, $\mathsf{b}\mathcal{D}\mathsf{AN}(h)$ is trivially an upper bound of this one, and as replacing the value of k by a greater integer keeps this property, we prove that the bound is coherent with the covering order.

In order to prove the first upper bound on $\mathcal{D}\mathsf{AN}(h)$ for any DSM h of degree k we need to recall some results and definitions, in order to use properties of the set $\mathsf{S_d}(h)$ defined in Theorem 4. A direct sum of monomials h of degree k is associated to its direct sum vector $\mathbf{m}_h = [m_1, \ldots, m_k]$ where $m_k \neq 0$ (Definition 12). From Theorem 1 we know that:

$$\mathsf{AI}(h) = \min_{0 \leq d \leq k} d + \sum_{i>d}^{k} m_i.$$

From Theorem 4 we know that:

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

where:

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

As explained in Remark 6 we can also notice the impact of m_1 on $\mathsf{S_d}(h)$:

- If $m_1 = 0$ then $0+\sum_{i>0}^{k} < 1+\sum_{i>0}^{k}$, so 0 can be in $\mathsf{S_d}(h)$ but 1 cannot, giving $\mathsf{S_d}(h) \subseteq \{0, 2, 3, \dots, k\}$,
- if $m_1 = 1$ then $0 + \sum_{i>0}^{k} = 1 + \sum_{i>0}^{k}$, as 0 cannot be in $\mathsf{S_d}(h)$ by definition, in this case it gives $\mathsf{S_d}(h) \subseteq \{1, \dots, k\}$,
- if $m_1 > 1$ then $0 + \sum_{i>0}^{k} > 1 + \sum_{i>0}^{k}$, so 0 cannot be in $\mathsf{S_d}$, giving $\mathsf{S_d}(h) \subseteq \{1, \dots, k\}$.

Then we highlight the link between the membership of d in $\mathsf{S_d}(h)$ and the values m_i, and the influence on the $\mathcal{D}\mathsf{AN}$. First, the membership of d in $\mathsf{S_d}(h)$ gives a relation on the successive m_i, as developed in the proof of Theorem 1:

$$d' \in \mathsf{S_d}(h) \iff d' + \sum_{i>d'}^{k} m_i = \min_{0 \leq d \leq k} d + \sum_{i>d}^{k} m_i.$$

It implies the following relation on the m_i:

- For all $e < d'$, $e \geq 0$:
$$d' + \sum_{i>d'}^{k} m_i \leq e + \sum_{i>e}^{k} m_i \iff \sum_{i>e}^{d'} m_i \geq d' - e, \tag{10}$$
- for all $e > d'$, $e \leq k$:
$$d' + \sum_{i>d'}^{k} m_i \leq e + \sum_{i>e}^{k} m_i \iff \sum_{i>d'}^{e} m_i \leq e - d'. \tag{11}$$

Then, we focus on the maximal contribution of each $d \in \mathsf{S_d}(h)$ to $\mathcal{D}\mathsf{AN}(h)$. Based on the formula of $\mathcal{D}\mathsf{AN}(h)$ from Theorem 4 the contribution of $d \in \mathsf{S_d}(h)$ is equal to $\prod_{i>d}^{k+1} i^{m_i}$. In the following we denote this quantity P_d for readability and consider $P_d = 0$ when $d \notin \mathsf{S_d}(h)$. Using Equation 11 on the m_i for $d' \in \mathsf{S_d}(h)$ (note that by definition $\mathsf{S_d}(h)$ cannot be empty) and k ($k \geq d'$ by definition) it gives $\sum_{i>d'}^{k} m_i \leq k - d'$. It leads to the following upper bound on $P_{d'}$ we will use:

$$P_{d'} \leq k^{k-d'}.$$

Now we bound $\mathcal{D}\mathsf{AN}(h)$ based on this upper bound and the value of m_1.

- Case $m_1 = 0$:
In this case, we saw that $\mathsf{S_d}(h) \subseteq \{0, 2, 3, \dots, k\}$. Let first consider the sub-case where $0 \notin \mathsf{S_d}(h)$, then:
$$\mathcal{D}\mathsf{AN}(h) \leq \sum_{d \in \mathsf{S_d}(h)} P_d \leq \sum_{j=2}^{k} P_j \leq \sum_{j=2}^{k} k^{k-j} = \frac{k^{k-1} - 1}{k-1} \leq k^k + 1,$$
where the equality is obtained as $k > 1$ (as m_k is the non null coefficient m_i such that i is maximal).

Then we consider the other sub-case *i.e.* where $0 \in \mathsf{S_d}(h)$, using Equation 11 it implies $\sum_{i=1}^{k} m_i \leq k$ then we denote t the integer such that $t = k - m_k$, as $m_k \neq 0$ by definition we get $0 \leq t < k$. Rewriting Equation 11 with $e = k$, for all d' in $\mathsf{S_d}(h)$ we obtain:

$$\sum_{i>d'}^{k} m_i \leq k - d' \Longleftrightarrow \sum_{i>d'}^{k-1} m_i \leq t - d'.$$

Therefore $\mathsf{S_d}(h)$ does not contain any integer in $\{t+1, \ldots, k-1\}$ (note that k can be in $\mathsf{S_d}(h)$). We denote u the maximal integer in $\mathsf{S_d}(h) \backslash \{k\}$, (properly defined as we assumed $0 \in \mathsf{S_d}(h)$). The precedent equation relatively to u gives $\sum_{i>u}^{k-1} m_i \leq t - u$, combining it with Equation 11 for the elements inferior to u we get:

$$\forall j \leq u,\ j \in \mathsf{S_d}(h), \quad P_j \leq u^{u-j} P_u \leq u^{u-j}(k-1)^{t-u} k^{k-t}.$$

Summing the upper bounds of all contributions:

$$\begin{aligned}\mathcal{D}\mathsf{AN}(h) &\leq \sum_{d \in \mathsf{S_d}(h)} P_d \leq P_k + \sum_{j=0}^{u} P_j \leq 1 + \sum_{j=0}^{u} u^{u-j}(k-1)^{t-u} k^{k-t}, \\ &\leq 1 + \left(\sum_{j=0}^{u} u^j \right) k^{k-u} \leq 1 + k^k.\end{aligned}$$

– Case $m_1 \geq 1$: In this case, we saw that $\mathsf{S_d}(h) \subseteq \{1, 2, 3, \ldots, k\}$, we can use the same strategy proof as in the precedent item, the main difference is that 0 has no contribution in the $\mathcal{D}\mathsf{AN}$ in this case. Note that $k = 1$ is possible, the case of degree 1 DSM is already taken in Theorem 4, giving in this case $\mathcal{D}\mathsf{AN}(h) \leq 1 < 1 + k^k$, thereafter we consider $k > 1$. As in the case $m_1 = 0$ we consider two sub-cases based on the membership of the minimal element possible in $\mathsf{S_d}(h)$. We begin with the sub-case $1 \notin \mathsf{S_d}(h)$, it gives:

$$\mathcal{D}\mathsf{AN}(h) \leq \sum_{d \in \mathsf{S_d}(h)} P_d \leq \sum_{j=2}^{k} P_j \leq \sum_{j=2}^{k} k^{k-j} = \frac{k^{k-1} - 1}{k - 1} \leq k^{k-1} + 1.$$

Then, for the other sub-case, $1 \in \mathsf{S_d}(h)$, Equation 11 on 1 relatively to k it gives: $\sum_{i=2}^{k} \leq k - 1$. As before we denote t the integer such that $t = k - m_k$, and u the maximal integer in $\mathsf{S_d}(h) \backslash \{k\}$. In the precedent part we prove that $u \leq t$ and for $j \leq u$ such that $j \in \mathsf{S_d}(h)$ we have $P_j \leq u^{u-j}(k-1)^{t-u} k^{k-t}$. Summing the upper bounds of all contributions we obtain:

$$\begin{aligned}\mathcal{D}\mathsf{AN}(h) &\leq \sum_{d \in \mathsf{S_d}(h)} P_d \leq P_k + \sum_{j=1}^{u} P_j \leq 1 + \sum_{j=1}^{u} u^{u-j}(k-1)^{t-u} k^{k-t}, \\ &\leq 1 + \left(\sum_{j=0}^{u-1} u^j \right) k^{k-u} \leq 1 + k^{k-1}.\end{aligned}$$

It concludes the proof on the first upper bound of $\mathcal{D}\mathsf{AN}(h)$ for any DSM h of degree k. As for all strictly positive integer k $k^{k-1}+1 \leq k^k+1$, $\mathsf{b}\mathcal{D}\mathsf{AN}(h)$ is an upper bound of $\mathcal{D}\mathsf{AN}(h)$. Then, considering any integer k' greater than $k > 0$ still gives an upper bound as $k'^{k'}+1 > k^k+1$.

□

Remark 7. Note that the first upper bound of Proposition 5 is reached by some functions. For all functions h of degree $k > 1$ such that $m_k = k$ and all others m_i are null are such that $\mathcal{D}\mathsf{AN}(h) = k^k+1$. It comes from the fact that $\mathsf{S}_\mathsf{d}(h)$ is reduced to $\{0, k\}$ and the associated contributions reach their upper bound. For all functions h of degree $k > 1$ such that $m_k = k$, $m_1 > 0$ and all others m_i are null are such that $\mathcal{D}\mathsf{AN}(h) = k^{k-1}+1$ (in this case $\mathsf{S}_\mathsf{d}(h)$ is reduced to $\{1, k\}$).

Note that the direct sum vector of the function of degree $k > 1$ of the second family covers the direct sum vector of the function of same degree of the first family. It is sufficient to show that this upper bound is not compatible with the covering order defined in the next subsection, explaining the choice of $\mathsf{b}\mathcal{D}\mathsf{AN}(h)$.

Third Modification, Determining DSM Descendant with Wosrse Properties First we show a relation between the direct sum vector of various descendants of the same function. Then, we use this relation to bound the parameters relatively to the standard criteria of all descendant functions on a same subset by the parameters of only one of these descendant. The purpose of these results is to be able to upper bound the number of descendants of a function which parameter is equal to a targeted value.

Definition 25 (Covering Vector). *Let $x \in \mathbb{F}_2^n$, and $y \in \mathbb{F}_2^n$, we say that y "covers" x, denoted $x \preceq y$ if and only if $\forall i \in [n],\ x_i \leq y_i$, where x_i and y_i are considered as natural numbers and "$\leq$" refers to the natural order of $\mathbb{N}$.*

Similarly, let f and g be DSM variables in n variables, we say that $\mathbf{m}_g$ "covers" $\mathbf{m}_g$, denoted $\mathbf{m}_f \preceq \mathbf{m}_g$ if and only if $\forall i \in [|\mathbf{m}_f|],\ \mathbf{m}_f(i) \leq \mathbf{m}_g(i)$.

The next property shows that the order on the binary vector of descendants with the same subset is conserved for the direct sum vector.

Property 2 (Covering and Direct Sum Vector). Let f be a DSM function in n variables, for any ℓ such that $1 \leq \ell < n$, and any subset $I \subseteq [n]$ such that $|I| = \ell$, for any $v \in \mathbb{F}_2^n$ and $w \in \mathbb{F}_2^n$:

$$\text{If } v \preceq w \text{ then } \mathbf{m}_{f_{I,v}} \preceq \mathbf{m}_{f_{I,w}}.$$

Note that when the descendant is a constant function, its associated direct sum vector is the null vector.

Proof. The two functions are descendants relatively to the same subset, so dealing with the same variables. Let us consider each monomial of f separately, when at least one variable is fixed to 0 the monomial is canceled (giving a degree 0 monomial), otherwise the degree of the monomial is reduced by the number of variables fixed to 1. The condition $v \preceq w$ means that for each $i \in [n]$ $v_i \leq w_i$, so for each monomial of f the remaining monomial of $f_{I,v}$ has a degree inferior than or equal to the degree of the same monomial of $f_{I,w}$. It finally gives $\mathbf{m}_{f_{I,v}}(i) \leq \mathbf{m}_{f_{I,w}}(i)$ for all $i \in |\mathbf{m}_f|$.

□

As for all $b \in \mathbb{F}_2^\ell$, $(0, \dots, 0) \preceq b \preceq (1, \dots, 1)$, the parameters of all descendant functions relatively to the same subset can be bound using the direct sum vectors of the all 0 or all 1 descendant. We show it it the following propositions.

Proposition 6. *Let f and g be DSM in n variables, if $\mathbf{m}_f \preceq \mathbf{m}_g$ then:*

- $\mathsf{res}(f) \leq \mathsf{res}(g)$,
- $\mathsf{NL}(f) \leq \mathsf{NL}(g)$,
- $\mathsf{AI}(f) \leq \mathsf{AI}(g)$,
- $\mathsf{bFAI}(f) \leq \mathsf{bFAI}(g)$,
- $\mathsf{b}\mathcal{D}\mathsf{AN}(f) \leq \mathsf{b}\mathcal{D}\mathsf{AN}(g)$.

Proof. **Resiliency**. Using Lemma 2 the resiliency of a DSM h is $\mathbf{m}_h(1) - 1$, then as $\mathbf{m}_f \preceq \mathbf{m}_g$, it implies $\mathbf{m}_f(1) \leq \mathbf{m}_g(1)$, so $\mathsf{res}(f) \leq \mathsf{res}(g)$.

Nonlinearity. Using Lemma 2, the nonlinearity of a DSM h in n variables is:

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

From the proof of the lemma we know that the contribution of each monomial of h to the nonlinearity depends on its degree. Each variable not appearing in the ANF (*i.e.* associated with null ANF coefficients only) or appearing in a degree 1 monomial gives a term equal to 2 in the product, whereas a monomial of degree $d \geq 2$ gives a term equal to $2^d - 2$ in this product. It gives that relatively, having variables in a degree greater than 2 monomials gives a smaller contribution in the product than having them not appearing in the ANF or in degree 1 monomials.

In the case we consider here, f and g are Boolean function in the same number of variables, and $\mathbf{m}_f \leq \mathbf{m}_g$. Then, for each $i \in [|\mathbf{m}_g|]$ such that $\mathbf{m}_f(i) < \mathbf{m}_g(i)$, the number of variables of these monomials in g corresponds to the same number of variables of f not appearing in its ANF. So, for each monomial of $j \geq 2$ variables relatively to $\mathbf{m}_g$ not in $\mathbf{m}_f$, it gives a contribution of $2^j - 2$ relatively to the nonlinearity of g and 2^j relatively to the one of f. Applying it to each monomials its gives:

$$\begin{aligned}
&\forall i \geq 2, \quad (2^i-2)^{\mathbf{m}_f(i)} (2^i)^{\mathbf{m}_g(i)-\mathbf{m}_f(i)} \geq (2^i-2)^{\mathbf{m}_g(i)},\\
&\Rightarrow \prod_{2}^{|\mathbf{m}_g|} (2^i-2)^{\mathbf{m}_f(i)} (2^i)^{\mathbf{m}_g(i)-\mathbf{m}_f(i)} \geq \prod_{2}^{|\mathbf{m}_g|} (2^i-2)^{\mathbf{m}_g(i)},\\
&\Rightarrow 2^{\left(n-\sum_{i=1}^{|\mathbf{m}_g|} i\mathbf{m}_g(i)\right)} \prod_{2}^{|\mathbf{m}_g|} (2^i-2)^{\mathbf{m}_f(i)} A \geq 2^{\left(n-\sum_{i=1}^{|\mathbf{m}_g|} i\mathbf{m}_g(i)\right)} \prod_{2}^{|\mathbf{m}_g|} (2^i-2)^{\mathbf{m}_g(i)},\\
&\Rightarrow 2^{\left(n-\sum_{i=1}^{|\mathbf{m}_f|} i\mathbf{m}_f(i)\right)} \prod_{2}^{|\mathbf{m}_f|} (2^i-2)^{\mathbf{m}_f(i)} \geq 2^{\left(n-\sum_{i=1}^{|\mathbf{m}_g|} i\mathbf{m}_g(i)\right)} \prod_{2}^{|\mathbf{m}_g|} (2^i-2)^{\mathbf{m}_g(i)},\\
&\Rightarrow \mathsf{NL}(f) \leq \mathsf{NL}(g),
\end{aligned}$$

where $A = (2^i)^{\mathbf{m}_g(i)-\mathbf{m}_f(i)}$.

Algebraic Immunity. Using Theorem 1, we get for a DSM h: $\mathsf{AI}(h) = \min_{0 \le d \le k} \left(d + \sum_{i=d+1}^{k} m_i\right)$. As for $i \ge 1$ we have $\mathbf{m}_f(i) \le \mathbf{m}_g(i)$:

$$\forall d \in [|\mathbf{m}_g|], \quad d + \sum_{i=d+1}^{|\mathbf{m}_g|} \mathbf{m}_f(i) \le d + \sum_{i=d+1}^{|\mathbf{m}_g|} \mathbf{m}_g(i),$$
$$\forall d \in [|\mathbf{m}_g|], \quad d + \sum_{i=d+1}^{|\mathbf{m}_f|} \mathbf{m}_f(i) \le d + \sum_{i=d+1}^{|\mathbf{m}_g|} \mathbf{m}_g(i),$$
$$\mathsf{AI}(f) \le \mathsf{AI}(g).$$

Fast Algebraic Immunity. we need to distinguish the cases where f and g are in the case $\mathsf{AI} = \mathsf{deg} > 1$ and $m_k > 1$, or not.

If one of the conditions is not fulfilled by f, then $\mathsf{bFAI}(f) = \mathsf{AI}(f) + 1$, and as from the precedent item $\mathsf{AI}(g) \ge \mathsf{AI}(f)$, it gives $\mathsf{bFAI}(f) \le \mathsf{bFAI}(g)$.

If f fulfills the three conditions, and so does g, $\mathsf{AI}(f) \le \mathsf{AI}(g)$ implies $\mathsf{bFAI}(f) \le \mathsf{bFAI}(g)$.

If f fulfills the three conditions, and not g, then at least one of the two conditions $\mathsf{AI}(g) = \mathsf{deg}(g)$ or $\mathbf{m}_g(|\mathbf{m}_g|) > 1$ is false. If the first condition is false, then $\mathsf{deg}(g) > \mathsf{AI}(g)$, so using the relation between $\mathbf{m}_f$ and $\mathbf{m}_g$ together with Theorem 1 (on f and g) gives:

$$\begin{aligned}
\mathsf{AI}(g) = \min_{0 \le d \le |\mathbf{m}_g|} d + \sum_{i=d+1}^{|\mathbf{m}_g|} \mathbf{m}_g(i) &= \min_{0 \le d \le |\mathbf{m}_g|} d + \sum_{i=d+1}^{|\mathbf{m}_f|} \mathbf{m}_g(i) + \sum_{i=|\mathbf{m}_f|+1}^{|\mathbf{m}_g|} \mathbf{m}_g(i), \\
&> \min\left(\min_{0 \le d \le |\mathbf{m}_f|} d + \sum_{i=d+1}^{|\mathbf{m}_f|} \mathbf{m}_g(i), |\mathbf{m}_f|\right), \\
&> \min\left(\min_{0 \le d \le |\mathbf{m}_f|} d + \sum_{i=d+1}^{|\mathbf{m}_f|} \mathbf{m}_f(i), |\mathbf{m}_f|\right), \\
&> \mathsf{AI}(f).
\end{aligned}$$

Therefore $\mathsf{bFAI}(g) = \mathsf{AI}(g) + 1 \ge \mathsf{AI}(f) + 2$, so $\mathsf{bFAI}(f) \le \mathsf{bFAI}(g)$.

If the second condition is false, then $\mathbf{m}_g(|\mathbf{m}_g|) = 1$. As f fulfills the condition $\mathbf{m}_f(|\mathbf{m}_f|) > 1$ and $\mathbf{m}_f \preceq \mathbf{m}_g$ it implies that $|\mathbf{m}_g| > |\mathbf{m}_f|$. Noticing that $\mathsf{AI}(f) = \mathsf{deg}(f)$, using Theorem 1 it gives that:

$$AI(g) > \min\left(\min_{0 \le d \le |\mathbf{m}_f|} d + \sum_{i=d+1}^{|\mathbf{m}_f|} \mathbf{m}_f(i), |\mathbf{m}_f|\right),$$

as previously. Then, the right hand side term being equal to $\mathsf{AI}(f)$ in this case, we get $\mathsf{AI}(f) < \mathsf{AI}(g)$, and we can conclude as in the previous case.

$\mathcal{D}\mathsf{AN}$. Finally, $\mathbf{m}_f \preceq \mathbf{m}_g$ so $|\mathbf{m}_f| \le |\mathbf{m}_g|$, enabling to conclude:

$$\mathsf{b}\mathcal{D}\mathsf{AN}(f) = |\mathbf{m}_f|^{|\mathbf{m}_f|} + 1 \le |\mathbf{m}_g|^{|\mathbf{m}_g|} + 1 = \mathsf{b}\mathcal{D}\mathsf{AN}(g).$$

□

Combining Property 2 with Proposition 6, we get:

Corollary 5. *Let f be a DSM function in n variables, for any ℓ such that $1 \le \ell < n$, and any subset $I \subseteq [n]$ such that $|I| = \ell$, and for any $b \in \mathbb{F}_2^n$:*

- $\mathsf{res}(f_I, b) \ge \mathsf{res}(f_{I,(0,\ldots,0)})$,
- $\mathsf{NL}(f_I, b) \ge \mathsf{NL}(f_{I,(0,\ldots,0)})$,
- $\mathsf{AI}(f_I, b) \ge \mathsf{AI}(f_{I,(0,\ldots,0)})$,
- $\mathsf{bFAI}(f_I, b) \ge \mathsf{bFAI}(f_{I,(0,\ldots,0)})$,
- $\mathsf{b}\mathcal{D}\mathsf{AN}(f_I, b) \le \mathsf{b}\mathcal{D}\mathsf{AN}(f_{I,(1,\ldots,1)})$,

where $(0,\ldots,0)$ and $(1,\ldots,1)$ denote the all 0 and the all 1 vectors of $\mathbb{F}_2^\ell$.

Corollary 5 justifies the consideration of the descendants obtained by fixing zeros only (and ones only) in the algorithm estimating the complexity of the attacks.

8.2 Concrete Instances with DSM Functions.

Some instances of $\mathsf{FiLIP}_{\mathsf{DSM}}$ are given in Table 1, λ is the conjectured security parameter, $\mathbf{m}_F$ is the direct sum vector notation of F, n is the number of variables of F, N is the size of the key register and d is the multiplicative depth of the function.

$\mathbf{m}_F$	n	N	d	Name
$\lambda = 80$				
$[89, 67, 47, 37]$	512	16384	2	FiLIP-512
$[80, 40, 0, 472]$	2048	16384	2	
$[80, 40, 15, 15, 15, 15]$	430	1792	3	FiLIP-430
$[80, 40, 0, 20, 0, 0, 0, 10]$	320	1800	3	FiLIP-320
$[86, 43, 0, 23, 0, 0, 0, 17]$	400	1024	3	
$[80, 40, 0, 0, 0, 0, 0, 32]$	416	1024	3	
$[80, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 16]$	416	1024	4	
$\lambda = 128$				
$[128, 64, 0, 80, 0, 0, 0, 80]$	1216	16384	3	FiLIP-1216
$[146, 149, 0, 139, 0, 0, 0, 131]$	2048	16384	3	
$[128, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64]$	1280	4096	4	FiLIP-1280

Table 1. $\mathsf{FiLIP}_{\mathsf{DSM}}$ Instances.

9 Improved Filter Permutators and XOR-THR Family of Functions.

The XOR-THR functions are good candidates to consider for the filtering function of IFPs: we call such instances $\mathsf{FiLIP}_{\mathsf{XMAJ}}$. First, this family of function is bit-fixing stable, and both XOR functions and majority functions are symmetric functions which have received a lot of attention (*e.g.* [Car04, CV05, DMS06, QLF07, SM07, QFLW09]). Second, this family corresponds to the main choice of function made in the context of Goldreich's PRG [MST03, AL16]. We show in Section 6 that these functions reach the best known locality bound of local PRG. Finally, threshold functions are known to be evaluable with different types of circuits, and we show how it can benefit to homomorphic evaluation as shown in Section 7.

9.1 Simplifying the Attack Complexities Algorithm for XOR-THR Functions.

In Section 4.2 we give the general framework to compute the complexity of the attacks. We describe in the following the modifications made to the algorithm in order to compute security bounds on XOR-THR functions.

First, note that in order to study the parameters of a XOR-THR function, the values of k, d and n are sufficient, then in comparison with DSM functions way less descendants have to be considered. When the profile relatively to one criterion is computed, only two parameters are needed to be stored for each descendant at order ℓ, as denoting $\mathsf{XOR}_{k'} + \mathsf{T}_{d',n'}$ the descendant and $\mathsf{XOR}_k + \mathsf{T}_{d,n}$ the filtering function, the following equations always hold: $0 \leq k' \leq k$, $0 \leq d' \leq d$, $0 \leq n' \leq n$, and $k' + n' = n - \ell$.

Then, at most $(n+1)*(d+1)$ have to be considered at each step, which makes the data complexity of the algorithm computing the profile way smaller than for DSM functions. With this quantity of descendants to consider, we do not modify the algorithm to neglect some descendants as in the DSM case, then the security estimates of the attacks are closer for these functions.

Then, the nonlinearity of XOR-THR functions is too low to consider values of δ close to $2^{-\lambda}$. Hence, we deviate from the analysis of Section 4 and consider the following functions as challenges rather than instances. In order to estimate the security of challenges with XOR-THR functions relatively to correlation-like attacks we estimate the costs of an XL attack based on linear approximation and simple correlation attacks. The first attack, introduced in [Cou03b], exploits the so-called XL algorithm. The principle is to look for good correlations between F a function in N variables, and an approximation g of degree d, in order to solve a linearized system based on the values of this approximation. The value ε is defined such that g is equal to F with probability greater than $1-\varepsilon$. Such attacks have a (very conservative) time complexity estimate:

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

For the challenges based on XOR-THR functions, we consider only degree one approximation (higher degree give higher value of the binomial coefficient which is generally the dominant term), with $D = 1$ and $m = N$. More precisely, in our algorithm we consider as time complexity estimation the previous bound times 2^ℓ at step ℓ, and the profile of nonlinearity stores the value of $\log(1-\varepsilon) = 1 - \frac{\mathsf{NL}}{2^{n-\ell}}$ with a precision of 10^{-3}. For the data complexity, the estimation is $P^{-1}(N-\ell)$, where P is the probability of having descendants with a value of $\log(1-\varepsilon)$ of at most k.

The other attacks we consider, simple correlation attacks, try to distinguish the key-stream from a random binary stream. We assume that a variant of these attacks aiming at recovering the seed have similar or greater data and time complexity. These attack consist in fixing some of the variables and distinguish the remaining part from the uniform distribution on $\mathbb{F}_2$. Consequently, (relatively to a function F) we consider $2^{\mathsf{res}}(F)\delta(F)^{-2}$ as a lower bound on the data complexity and $\delta(F)^{-2}$ as a lower bound on the time complexity. For the complexities of the general algorithm using the profile, as usual the time complexity considers an additional factor of 2^ℓ for the guesses and the data complexity a factor P^{-1} where the probability is computed from N and the profile. Note that here, this probability could require to consider the nonlinearity (or δ) and resiliency of each descendant, but we consider a profile easier to handle in practice, considering separately resiliency and nonlinearity, and leading to under estimated costs. More precisely we consider the data complexity based on the nonlinearity profile with a fixed value of the resiliency: the worst parameter obtained at fixed ℓ. Alternatively we consider the data complexity based on the resiliency profile with a fixed value of the nonlinearity: the worst parameter obtained at fixed ℓ. The maximum of these two numbers is the estimation we use for this kind these attacks.

Finally, we focus on a particular sub-family of XOR-MAJ functions based on consideration relatively to the homomorphic evaluation and the representation of the filtering function. As the multiplicative depth of the decryption algorithm have been enhanced in precedent works, we focus on functions with a degree being a power of two, and with maximum possible algebraic immunity relatively to this degree. The degree of functions of the form $\mathsf{T}_{2^r,2^{r+1}-1}$ is given in [CV05]. It explains why we choose the threshold part as $\mathsf{T}_{2^r,2^{r+1}-1}$, but other choices are also valid, and it should be taken in adequation with the homomorphic evaluation.

9.2 Concrete Challenges with XOR-MAJ Functions.

We give potential instances with XOR-THR functions, in terms of $\mathsf{FiLIP}_{\mathsf{XMAJ}}$. The motivation of these instances is to instantiate IFPs as close as possible to considered instantiations of Goldreich's PRG, then they should more be seen as an initial challenge to understand the concrete security of such constructions (as investigated in [CDM$^+$18]) rather than instances to implement. The main reason is that the nonlinearity of such functions is extremely low, it does not seem to give security issues asymptotically, but for fixed instances with relatively small n the system of equation may be too close to a linear system to avoid correlation-like attacks. To illustrate this claim, note that using Stirling approximation (and Theorem 2), the bias δ of a majority function is $\delta(\mathsf{T}_{\frac{n+1}{2},n}) \approx \frac{1}{2\sqrt{\pi(n-1)/2}}$. Also, majority functions are known to have relatively small nonlinearity, in [Lob09] Lobanov shows that they correspond to the functions with optimal algebraic immunity with worst possible nonlinearity. Furthermore, considering δ as small as in the security analysis of Section 4 corresponds to functions with too many variables to be considered in our context.

In terms of security, two modifications of these functions will be preferred for the context of IFPs. A rather natural construction to increase the nonlinearity is to consider the direct sum of a XOR-THR function with the quadratic bent function with direct sum vector $[0, \lambda/2]$. On the homomorphic side, the error-growth mostly depends on the evaluation of the threshold function, and as it adds only $\lambda/2$ products, the time of the evaluation has the same order. On the Boolean function side, the properties on direct sum constructions enable to derive the resiliency and nonlinearity, and lower bounds on the algebraic immunity and fast algebraic immunity, showing that these functions improve the security upon XOR-THR functions. Another modification consists in building the direct sums of various threshold functions. An efficient evaluation of threshold functions on the homomorphic side leads to an efficient evaluation of the whole function in this context. On the Boolean function side, determining all exact parameters (as the algebraic immunity and fast variant) might be very challenging, but known bounds enable to give security estimates, as for the other modification.

Potential challenges of $\mathsf{FiLIP}_{\mathsf{XMAJ}}$ are given in Table 2, n is the number of variables of F, d is the multiplicative depth of the function in ANF representation, r is the number of multiplications using the MUX gates (see Section 7), and N is the size of the key register relatively to λ the conjectured security parameter.

F	n	d	r	$N, \lambda = 80$	$N, \lambda = 128$
$\mathsf{XOR}_{81} + \mathsf{T}_{8,15}$	96	3	69	32768	
$\mathsf{XOR}_{113} + \mathsf{T}_{16,31}$	144	4	269	4096	65536
$\mathsf{XOR}_{81} + \mathsf{T}_{32,63}$	144	5	1053	2048	16384
$\mathsf{XOR}_{129} + \mathsf{T}_{64,127}$	256	6	4157	1536	8192
$\mathsf{XOR}_{257} + \mathsf{T}_{128,255}$	512	7	16509	1024	4096

Table 2. $\mathsf{FiLIP}_{\mathsf{XMAJ}}$ Instances.

10 Performance Evaluation.

Ultimately, the goal of SE-FHE applications is to obtain the result of the homomorphic computations with the best latency and throughput. However, such performance metrics can only be evaluated if the functions to be evaluated by the Cloud are known in advance. In previous evaluations of symmetric ciphers for FHE evaluation, this issue was (partially) circumvented by studying the latency and throughput of homomorphic ciphertexts that will just enable decryption or a fixed number of levels of multiplications. This allows getting lower bounds on the timings necessary to evaluate any function, and the performances are reasonably accurate for simple functions with the given multiplicative depth. Yet, one important drawback of this approach remains that optimizing latency and throughput requires to fix parameters such as the size of the ciphertexts and the quantity of noise (which set the security of the FHE scheme). More precisely, in homomorphic encryption, it is the quantity of noise that determines the size of the ciphertexts required to correctly handle the operations. This size is in turn the main factor determining the latency and throughput of the homomorphic operations. Therefore, optimizing throughput and latency is ideal for one specific function, but it looses its accuracy when the application deviates from this particular function. We next propose an alternative comparison methodology, based on the homomorphic noise, that aims to be more independent of the applications.

10.1 Methodology.

Considering the performances of SE-FHE relatively to the homomorphic noise is based on two simple principles. The smaller is the noise, the wider is the class of functions still evaluable on these ciphertexts. The smaller is the noise, the smaller are the homomorphic ciphertexts, the faster are the evaluations. It means that the homomorphic noise dictates the ciphertext parameters, and eventually the latency and throughput of the final application. Consequently, an appealing performance evaluation could consist in determining exactly the error-growth (in average or with overwhelming probability) given by an SE scheme relatively to a specific FHE scheme. As there is no simple parameter (such as the multiplicative depth) which encompasses totally the error-growth, we use a simpler methodology consisting in measuring the noise just after evaluating the symmetric decryption or after some additive levels of multiplications.

In contrast with the aforementioned latency/throughput oriented methodology, which leads to fix the homomorphic parameters to optimize the timings for a given target function, a noise-oriented methodology can ensure that the ciphertext parameters are the same for all SE schemes to be compared. This has two advantages. First, all homomorphic ciphertexts obtained have the same security, that we fix to λ, the security level of the symmetric scheme. Second, once the symmetric decryption is performed, the evaluation time of any function will be independent of the SE scheme used for the transciphering. Such a scheme is then only limited by the ciphertext noise, which determines the quantity of operations that can be performed until decryption becomes impossible. Consequently, with this methodology, the smaller is the measured noise, the better suited is the SE scheme. We believe this approach provides interesting insights for the comparison of SE schemes in an application-dependent manner.

Additionally, and for completeness, we give some indications on the time performances, using the strategy of previous works. To do so, for each symmetric scheme we select homomorphic parameters that are sufficient to evaluate the decryption circuit, but no more. It gives an idea on the minimal size of homomorphic ciphertext and minimal evaluation time required for each SE scheme relatively to the library used. The result corresponds to a minimum as for any application, bigger ciphertexts are necessary to make the evaluations of the computation part.

10.2 Performances and Comparisons.

We chose to compare the following symmetric schemes: LowMC [ARS$^+$15], FLIP [MJSC16], Rasta and Agrasta [DEG$^+$18] and $\mathsf{FiLIP}_{\mathsf{DSM}}$, all designed for the SE-FHE framework. We did not consider Kreyvium [CCF$^+$16] as its implementation is very different (based on previous studies the related numbers would be slightly better than the one of LowMC due to a multiplicative depth of 12 and 13). All implementations were made with the HElib library [HS14]. The LowMC implementations were taken from the publicly available code (`https://bitbucket.org/malb/lowmc-helib`). The one of Rasta were built from this implementation and the publicly available one (`https://github.com/iaikkrypto/rasta`). We use the same code for computing the "Four Russians" method, used for multiplying binary matrices. The FLIP and $\mathsf{FiLIP}_{\mathsf{DSM}}$ implementations were made ad hoc. These implementations were evaluated on laptop computer with processor Intel(R) Core(TM) i5-4210M CPU at 2.60GHz.

Accordingly to the previously described methodology, we chose parameters in HElib enabling to evaluate the decryption of all these schemes. These parameters are in this case dictated by LowMC (due to its higher multiplicative depth), so we choose the minimal parameters such that LowMC ciphertexts can be decrypted while keeping an FHE security of at least the security of the symmetric ciphers. The noise level after evaluation is estimated thanks to HElib function log_of_ratio() which returns the difference $\log(\sigma) - \log(q)$ where σ^2 is the noise variance[3] of the error part of the ciphertext, and q is the modulus. In order to have a glimpse of what this noise level represents, we also computed 1 (respectively 2) level(s) of multiplications between ciphertexts (after the homomorphic evaluation of the symmetric schemes).

The results for 80-bit security and 128-bit security are given in Table 3 and Table 4[4]. Symbol d denotes the multiplicative depth of the decryption circuit of the SE scheme, N is the key size, symbol b denotes the number of bits produced. The latency refers to the time required to have the first ciphertext after evaluation, the noise columns refer to the output of the log_of_ratio() function, with respectively 0, 1 and 2 levels of multiplications (after evaluation of the symmetric decryption function).

From these results we can conclude that LowMC ciphertexts have the biggest error-growth. For the 80-bit security instances the noise after evaluating FLIP, Rasta or Agrasta is similar whereas the instances of $\mathsf{FiLIP}_{\mathsf{DSM}}$ enable 1 or 2 additional levels of multiplications. For 128 bits of security, FiLIP-1280 ciphertexts are slightly less noisy than Agrasta and FLIP ciphertexts, whereas FiLIP-1216 offers an additional level of multiplications. In terms of evaluation time, the parameters are more suited for LowMC, but relatively to this size of ciphertexts we can conclude that Agrasta evaluations produce more ciphertexts per second. The instances of $\mathsf{FiLIP}_{\mathsf{DSM}}$ produce the ciphertexts one by one, and have then a throughput around 50 times

[3] This variance is derived from bounds on the error-growth of addition, product, automorphism, and switchings.

[4] These security levels are the one given by HElib, more accurate estimations are given in [Alb17]

Cipher	d	N	b	Latency (s)	noise	noise $\times$	noise $\times^2$
LowMCv2 (12, 31, 128)	12	80	128	329.38	-2.966	n/a	n/a
LowMCv2 (12, 49, 256)	12	80	256	699.10	-2.495	n/a	n/a
Agrasta (81, 4)	4	81	81	67.48	-155.722	-139.423	-119.459
Rasta (327, 4)	4	327	327	290.99	-154.502	-139.423	-119.459
Rasta (327, 5)	5	327	327	366.30	-135.727	-119.459	-100.641
FLIP-530	4	530	1	42.06	-157.201	-139.423	-119.459
FiLIP-512	2	16384	1	33.74	-194.342	-177.739	-158.241
FiLIP-430	3	1792	1	31.25	-176.039	-158.241	-139.423
FiLIP-320	3	1800	1	21.41	-176.588	-158.241	-139.423

Table 3. Noise comparison for 80-bit security. HElib parameters: LWE dimension of the underlying lattice = 15709, HElib Depth L = 14, B = 28 (Bit per level parameter that influence BGV security), BGV security = 84.3, Nslots = 682, log_of_ratio() of fresh ciphertext : -237.259.

slower for 80-bit instances and 200 for 128-bit instances. These results confirm the excellent behavior of $\mathsf{FiLIP}_{\mathsf{DSM}}$ in terms of noise, enabling 1 or 2 supplementary levels of multiplication (at the cost of a moderate decrease of the time performances detailed next).

Note that the gain in depth of $\mathsf{FiLIP}_{\mathsf{DSM}}$ relatively to Agrasta or Rasta is obtained at the price of larger key sizes. When choosing which scheme to use in the hybrid homomorphic framework, a trade-off can be considered between these schemes, depending on the number of levels of multiplications required (computation phase) and constraints on the key-size (initialization phase). The more computations over the data will be considered, the more important will be the influence of the error-growth, making negligible the impact of the key-size (provided that its storage is manageable in clear by the user, and in homomorphic by the server).

We also note that in [DEG$^+$18], instances with a smaller multiplicative depth are considered, but the authors recommend a depth at least 4 for security reason. These alternative instances always involve way bigger keys than $\mathsf{FiLIP}_{\mathsf{DSM}}$ instances with the same multiplicative depth, and due to the high number of XORs in these alternative instances, the error-growth would be higher. Rasta ciphers were not optimized for the metric we consider, instances designed for the error-growth could lead to better performances. We argue that minor modifications would benefit to evaluation over HElib, but by design the noise is larger than the one from IFPs. For example, the high number of additions occurring at different levels between multiplications prohibits Rasta design to be used in a SE-FHE framework using 3G FHE, whereas IFPs are performing well for all known FHE.

For completeness we also study the performance results in time for the different SE ciphers considered. For this purpose, we chose the HElib parameters such that the ciphers can just be decrypted (by setting the appropriate L value), while keeping a similar security level for the HE scheme (by modifying with trial and errors the other parameters). These numbers here have to be taken as a global behavior of the achievable performances of the ciphers. We report the results in Table 5 for 80-bit and Table 6 for 128-bit security (note that these estimations of security are the one given by HElib, more accurate one are can be found in [Alb17], but it does not affect the comparison). B is the bit per level parameter, m is the LWE dimension, L is the HElib depth, λ' is the BGV security, ns the number of slots. The latency refers to the time required to have

Cipher	d	N	b	Latency (s)	noise	noise $\times$	noise $\times^2$
LowMCv2(14, 63, 256)	14	128	256	1629.03	-3.418	n/a	n/a
Agrasta (129, 4)	4	128	129	207.68	-207.478	-190.086	-169.011
Rasta (525, 5)	5	525	525	1264.30	-185.885	-169.011	-148.313
Rasta (351, 6)	6	351	351	967.62	-164.945	-148.313	-129.716
FLIP-1394	4	1394	1	272.31	-207.831	-190.086	-169.011
FiLIP-1216	3	16384	1	251.28	-227.93	-210.437	-190.086
FiLIP-1280	4	4096	1	325.04	-208.112	-190.086	-169.011

Table 4. Noise comparison for 128-bit security. HElib parameters: LWE dimension dimension of the underlying lattice = 24929, HElib Depth = 16, B = 30, BGV security = 132.1, Nslots = 512, log_of_ratio() of fresh ciphertext : -293.929.

the first ciphertext after evaluation, the noise columns refers to the output of the log_of_ratio() function after evaluation of the symmetric scheme decryption.

Many optimizations can still be made in the code itself but also in the choice of the FHE parameters. These results show that, adapting the FHE parameters to the decryption of the symmetric scheme only, the throughput can be sensibly increased. For some schemes the ciphertexts are still usable for more evaluations, it comes from the fact that HElib rejects smaller values of L, whereas the multiplicative depth of the scheme is inferior. Then it does not enable us to compare perfectly the different schemes.

Cipher	B	m	L	λ'	ns	Latency (s)	noise
LowMCv2(12, 31, 128)	28	15709	14	84.3	682	329.38	-2.966
LowMCv2(12, 49, 256)	28	15709	14	84.3	682	699.10	-2.495
Agrasta (81, 4)	26	5461	5	82.9	378	12.97	-2.03
Rasta (327, 4)	26	8435	5	84.6	240	76.33	-1.903
Rasta (327, 5)	25	7781	7	85.1	150	90.78	-14.42
FLIP-530	21	4859	5	85.3	168	6.48	-1.23
FiLIP-512	21	4859	5	85.3	168	7.05	-29.09
FiLIP-430	21	4859	5	85.3	168	6.01	-15.457
FiLIP-320	21	4859	5	85.3	168	5.04	-16.02

Table 5. Performances for minimal FHE parameters, 80-bits security.

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.

Cipher	B	m	L	λ'	ns	Latency (s)	noise
LowMCv2(14, 63, 256)	30	24929	16	132.1	512	1629.3	-3.418
Agrasta (129, 4)	27	7781	5	134.7	150	20.26	-3.03
Rasta (525, 5)	27	10261	7	128.9	330	277.24	-20.441
Rasta (351, 6)	27	10261	8	128.9	330	195.40	-1.92
FLIP-1394	28	8191	6	146.8	630	26.53	-5.11
FiLIP-1216	22	7781	5	186.3	150	24.37	-15.94
FiLIP-1280	28	8191	6	146.8	630	26.59	-5.11

Table 6. Performances for minimal FHE parameters, 128-bits security.

[AD18] Tomer Ashur and Siemen Dhooghe. Marvellous: a stark-friendly family of cryptographic primitives. Cryptology ePrint Archive, Report 2018/1098, 2018. `https://eprint.iacr.org/2018/1098`.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 191–219. Springer, Heidelberg, December 2016.

[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.

[AL18] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. *SIAM J. Comput.*, (1):52–79, 2018.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Heidelberg, May 2017.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314. Springer, Heidelberg, August 2014.

[App12] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 805–816. ACM Press, May 2012.

[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+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, April 2015.

[ARS+16] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. *IACR Cryptology ePrint Archive*, page 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.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, August 2012.

[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.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014*, pages 1–12. ACM, January 2014.

[BY03] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 1–18. Springer, Heidelberg, April 2003.

[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.

[CDM$^+$18] Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the concrete security of goldreich's pseudorandom generator. In *ASIACRYPT 2018, Part I*, LNCS. Springer, Heidelberg, December 2018.

[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.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 311–328. Springer, Heidelberg, March 2014.

[CM03] Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*. Springer, Heidelberg, May 2003.

[CMR17] Claude Carlet, Pierrick Méaux, and Yann Rotella. Boolean functions with restricted input and their robustness; application to the FLIP cipher. *IACR Trans. Symmetric Cryptol.*, (3), 2017.

[Cou03a] Nicolas Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 176–194. Springer, Heidelberg, August 2003.

[Cou03b] Nicolas Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of toyocrypt. In Pil Joong Lee and Chae Hoon Lim, editors, *ICISC 02*, volume 2587 of *LNCS*. Springer, Heidelberg, November 2003.

[CV05] Anne Canteaut and Marion Videau. Symmetric boolean functions. *IEEE Trans. Information Theory*, (8):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.

[DGM04] Deepak Kumar Dalai, Kishan Chand Gupta, and Subhamoy Maitra. Results on algebraic immunity for cryptographically significant Boolean functions. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 92–106. Springer, Heidelberg, December 2004.

[DGM05] Deepak Kumar Dalai, Kishan Chand Gupta, and Subhamoy Maitra. Cryptographically significant Boolean functions: Construction and analysis in terms of algebraic immunity. In Henri Gilbert and Helena Handschuh, editors, *FSE 2005*, volume 3557 of *LNCS*, pages 98–111. Springer, Heidelberg, February 2005.

[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*, (1), 2006.

[DSES14] Yarkin Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. Toward practical homomorphic evaluation of block ciphers using prince. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *FC 2014 Workshops*, volume 8438 of *LNCS*, pages 208–220. Springer, Heidelberg, March 2014.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing groebner bases. *Journal of Pure and Applied Algebra*, 139:61–88, june 1999.

[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.

[GGNS13] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block ciphers that are easier to mask: How far can we go? In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013.*, volume 8086 of *Lecture Notes in Computer Science*. Springer, 2013.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, August

2012.

[GLSV14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. Ls-designs: Bitslice encryption for efficient masked software implementations. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, 2014.

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

[GRR$^+$16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. Mpc-friendly symmetric key primitives. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[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.

[HAO15] Ryo Hiromasa, Masayuki Abe, and Tatsuaki Okamoto. Packing messages and optimizing bootstrapping in GSW-FHE. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*. Springer, Heidelberg, 2015.

[HKM17] Matthias Hamann, Matthias Krause, and Willi Meier. LIZARD – A lightweight stream cipher for power-constrained devices. *IACR Trans. Symm. Cryptol.*, 2017(1):45–79, 2017.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2014.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *40th ACM STOC*, pages 433–442. ACM Press, May 2008.

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

[Knu97] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley Professional, third edition, November 1997.

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 14*, volume 8469 of *LNCS*, pages 318–335. Springer, Heidelberg, May 2014.

[LNV11] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? Cryptology ePrint Archive, Report 2011/405, 2011.

[Lob09] M. S. Lobanov. Exact relations between nonlinearity and algebraic immunity. *Journal of Applied and Industrial Mathematics*, (3):367–376, Jul 2009.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. Cryptology ePrint Archive, Report 2017/250, 2017.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.

[Mes17] Sihem Mesnager. On the nonlinearity of boolean functions with restricted input. Talk at The 13th International Conference on Finite Fields and their Applications, 2017.

[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.

[MMM$^+$18] Subhamoy Maitra, Bimal Mandal, Thor Martinsen, Dibyendu Roy, and Pantelimon Stănică. Tools in analyzing linear approximation for boolean functions related to flip. In Debrup Chakraborty and Tetsu Iwata, editors, *INDOCRYPT 2018*, pages 282–303, 2018.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

[MS78] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th FOCS*, pages 136–145. IEEE Computer Society Press, October 2003.

[MZD18] Sihem Mesnager, Zhengchun Zhou, and Cunsheng Ding. On the nonlinearity of boolean functions with restricted input. *Cryptography and Communications*, Mar 2018.

[PRC12] Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - A block cipher allowing efficient higher-order side-channel resistance. In *ACNS 2012, Singapore, 2012. Proceedings*, pages 311–328, 2012.

[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*, 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*, (19):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.