The Nonlinear Filter Model of Stream Cipher Redivivus

Claude Carlet[1,2] and Palash Sarkar*[3]

[1]LAGA Laboratory, University of Paris 8, 93526 Saint-Denis, France
[2]University of Bergen, Norway
[3]Indian Statistical Institute, 203, B.T. Road, Kolkata, India 700108
Emails: claude.carlet@gmail.com, palash@isical.ac.in

June 5, 2025

Abstract

The nonlinear filter model is an old and well understood approach to the design of stream ciphers. Extensive research over several decades has shown how to attack stream ciphers based on this model and has identified the security properties required of the Boolean function used as the filtering function to resist such attacks. This led to the problem of constructing Boolean functions which provide adequate security *and* at the same time are efficient to implement. Unfortunately, over the last two decades no good solutions to this problem appeared in the literature. The lack of good solutions has effectively led to nonlinear filter model becoming more or less obsolete. This is a big loss to the cryptographic design toolkit, since the great advantage of the nonlinear filter model lies, beyond its simplicity and its ability to provide low-cost solutions for hardware-oriented stream ciphers, in the accumulated knowledge about the security requirements for the filtering function, which gives confidence in its security when all criteria are met. In this paper we construct balanced functions on an odd number $n \geq 5$ of variables with the following provable properties: linear bias equal to $2^{-\lfloor n/2 \rfloor -1}$, algebraic degree equal to $2^{\lfloor \log_2 \lfloor n/2 \rfloor \rfloor}$, algebraic immunity at least $\lceil (n-1)/4 \rceil$, fast algebraic immunity at least $1 + \lceil (n-1)/4 \rceil$, *and* the functions can be implemented using $O(n)$ NAND gates. The functions are obtained from a simple modification of the well known class of Maiorana-McFarland bent functions. Due to the efficiency of implementation, for any target security level, we can construct functions which provide the required level of resistance to fast algebraic and fast correlation attacks. By appropriately choosing n and the length L of the linear feedback shift register, we show that it is possible to obtain examples of stream ciphers which are *provably* κ-bit secure against well known classes of attacks for various values of κ. We provide concrete proposals for $\kappa = 80, 128, 160, 192, 224$ and 256 using LFSRs of lengths 163, 257, 331, 389, 449, 521 and filtering functions on 75, 119, 143, 175, 203 and 231 variables. For the 80-bit, 128-bit, and the 256-bit security levels, the circuits for the corresponding stream ciphers require about 1743.5, 2771.5, and 5607.5 NAND gates respectively. For the 80-bit and the 128-bit security levels, the gate count estimates compare quite well to the famous ciphers Trivium and Grain-128a respectively, while for the 256-bit security level, we do not know of any other stream cipher design which has such a low gate count.

Keywords: Boolean function, stream cipher, nonlinearity, algebraic immunity, efficient implementation.

*Corresponding author.

1 Introduction

The nonlinear filter model of stream ciphers is several decades old; one may note that the model was extensively discussed in the book by Rueppel [49] which was published in the mid-1980s. The nonlinear filter model consists of two components, namely a linear feedback shift register (LFSR) and a Boolean function f which is applied to a subset of the bits of the LFSR at fixed positions (called tap positions). At each clock, f is applied to the present state of the LFSR to produce a single keystream bit and simultaneously the LFSR also moves to the next state. LFSRs are efficient to implement in hardware. So the implementation efficiency of the nonlinear filter model is essentially determined by the efficiency of implementation of f.

Extensive research carried out over the last few decades has shown several approaches to cryptanalysing the nonlinear filter model of stream ciphers. The initial line of attack was based upon determining the linear complexity of the produced keystream. For an LFSR of length L and a Boolean function of algebraic degree d, under certain reasonable and easy-to-ensure conditions, the linear complexity of the keystream is known to be at least $\binom{L}{d}$ (see [49]). Using large enough values of L and d, linear complexity based attacks can be made infeasible. The second phase of attacks consisted of various kinds of (fast) correlation attacks. Starting with the first such attack in 1985 [53] (which was efficient on another model, the nonlinear combiner model), a long line of papers [52, 45, 29, 3, 21, 35, 36, 19, 13, 37, 12, 38, 20, 54, 60, 59, 39, 40] explored various avenues for mounting correlation attacks. Surveys of some of the older attacks appear in [10, 11, 43, 2]. Fast correlation attacks are based on affine approximation and apply to the nonlinear filter (as well as the nonlinear combiner) model. The resistance to fast correlation attacks is mainly determined by the linear bias ε of the Boolean function f. The linear bias is determined by the nonlinearity of the function f; the higher the nonlinearity, the lower the linear bias. The third phase of attacks started in 2003 with the publication of the algebraic attack [23] and was soon followed by the publication of the fast algebraic attack [22]. Resistance to these attacks requires the function f to possess high (fast) algebraic immunity.

The various attacks mentioned above have posed the following design challenge for a Boolean function to be used in the nonlinear filter model of stream ciphers. Construct balanced Boolean functions which achieve a good combination of nonlinearity and algebraic resistance *and* are also very efficient to implement. Unfortunately, since the time algebraic attacks were proposed in the early-2000s, no good solutions to the design challenge for Boolean functions have appeared in the literature (some Boolean functions satisfy all the necessary cryptographic criteria, but are too heavy and slow to compute [17], and some others are fast to compute but possess insufficient nonlinearity [55, 57, 15, 42]). A consequence of not being able to find good solutions to the design challenge is that the nonlinear filter model of stream ciphers became obsolete. This is somewhat unfortunate since the model is very old, well studied with well understood security, and the potential to provide low gate count solutions in hardware.

A class of guess-then-determine attacks are known against the nonlinear filter model. The inversion attack [27] is the first such attack. Subsequently, a number of papers have developed the idea into the generalised inversion attack, the filter state guessing attack, and the generalised filter state guessing attack [27, 28, 33, 47, 58]. Recommendations for resistance to the state guessing attack do not specify conditions on the filtering function. Rather, the recommendations specify conditions on the tap positions, i.e. the LFSR positions where the filtering function inputs are drawn. We note that strictly following these recommendations with a LFSR of usual size (say about 256 flip-flops) implies taking a number of variables from the filtering function much lower than the desired security level.

We note that even though the nonlinear filter model became obsolete, use of LFSRs in the design of stream cipher has continued for both hardware and software oriented proposals (see for example [32, 5, 26]). Instead of using a Boolean function, such designs typically use a nonlinear finite state machine to filter the output of the LFSR. Some ciphers such as [9, 4] have gone further and replaced the LFSR with one or more nonlinear feedback shift registers (NFSRs).

In this paper, we revisit the above mentioned design problem for Boolean functions towards the goal of reviving the nonlinear filter model. Bent functions [48] are a very well studied class of Boolean functions. They exist for even number of variables and provide the highest possible nonlinearity. However, their use in cryptography cannot be direct, since they are unbalanced, and even if we modify them into balanced functions (as Dobbertin [25] proposed) that use is not clear, after the invention of fast algebraic attacks. The introduction of Chapter 6 of [14] summarises the state-of-the-art as follows: "we do not know an efficient construction using bent functions which would provide Boolean functions having all the necessary features for being used in stream ciphers." Note that a result from [56] seemed to imply that it was impossible to obtain a Boolean function having a good resistance to fast algebraic attacks by modifying a bent function into a balanced function. This result happened to be incorrect as shown in [14] (see Theorem 22 which corrects the result, and the few lines before it). Even after this correction, the problem of building good cryptographic functions from bent functions seemed hard. In the present paper we provide a concrete solution to this problem. We even show that it is possible to start with a bent function obtained from the very basic Maiorana-McFarland construction.

The well known Maiorana-McFarland class of bent functions is defined as follows. For $m \geq 1$, let $\mathbf{X}$ and $\mathbf{Y}$ be two vectors of m variables. Then a $2m$-variable Maiorana-McFarland bent function is defined to map $(\mathbf{X}, \mathbf{Y})$ to $\langle \pi(\mathbf{X}), \mathbf{Y} \rangle \oplus h(\mathbf{X})$, where π is a bijection from m-bit strings to m-bit strings and h is any m-variable Boolean function. It is well-known that the nonlinearity of Maiorana-McFarland functions does not depend on the choice of h nor on that of the permutation π. Bad choices of both π and h (for example, π to be the identity permutation and h to be a constant function) provide functions whose algebraic immunity is low. On the bright side, we observe that it may be possible to improve the algebraic resistance by properly choosing h. Assume that π is chosen to be an affine map, i.e. the coordinate functions of π are affine functions of its input variables. It is known [24] that the majority function possesses the best possible algebraic immunity. Motivated by this fact, we choose h to be the majority function on m variables. We *prove* that the resulting bent function on $2m$ variables has algebraic immunity at least $\lceil m/2 \rceil$, and hence fast algebraic immunity at least $1 + \lceil m/2 \rceil$. On the implementation aspect, we show that the majority function can be computed using $O(m)$ NAND gates. So the obtained bent function has maximum nonlinearity, sufficient (fast) algebraic immunity when the number of variables is large enough (to protect against fast algebraic attacks at a specified security level), and at the same time is quite efficient to implement. Indeed, the (fast) algebraic immunity is not the maximum possible, but due to the implementation efficiency, it is possible to increase the number of variables to achieve the desired level of algebraic resistance. A novelty of our work is *the observation and the proof* that choosing h to be the majority function improves the algebraic immunity. While Maiorana-McFarland bent functions have been extensively studied in the literature, this simple observation has escaped the notice of earlier researchers. To the best of our knowledge, our result is the first construction of bent functions with a provable lower bound on algebraic immunity.

One problem is that a bent function is not balanced. This problem is easily rectified by XORing a new variable to the bent function, obtaining a function on an odd number of variables. This modification requires only one extra XOR gate for implementation. In terms of security, the modification does not change the linear bias. We prove that the algebraic immunity of the modified function is at least the algebraic immunity of the bent function, and hence the fast algebraic immunity is at least one plus the algebraic immunity of the bent function. So the algebraic resistance of the modified function is essentially the same as that of the bent function. A consequence of unbalancedness of the filtering function is that the keystream sequence is unbalanced, which leads to a distinguishing attack. The simple modification of XORing an extra variable prevents such a distinguishing attack. The resistance to fast correlation attacks and fast algebraic attacks are ensured by the bent function (on an appropriate number of variables). A positive aspect of XORing a new variable is that it prevents (see Theorem 2 of [27]) certain kinds of information leakage which was identified in [3]. On the negative side, filtering functions of the form $W + g(\mathbf{Z})$ were shown to be susceptible to the inversion attack [27]. We prevent this attack by choosing the gap between the first and the last tap positions to be sufficiently large; this countermeasure was already proposed in [27]. More generally, we show that due to the fact that we use filtering functions on a large number of variables, the various kinds of state guessing attacks [28, 33, 47, 58] do not apply to our proposals.

The literature [27, 28, 33, 47, 58] provides guidance on the choice of tap positions, i.e. the positions of the LFSR which are to be tapped to provide input to the filtering function. One such recommendation is to ensure that the tap positions form a full positive difference set. As we show later, this condition requires the length L of the LFSR to be at least quadratic in the number n of variables of the filtering function. In our proposals, the value of n is close to the target security level κ. So if the tap positions are to be chosen to satisfy the full positive difference set condition, then L will become too large, and the resulting stream cipher will be of no practical interest. The recommendations in the literature on selection of tap positions are essentially sufficient (but not necessary) conditions to prevent state guessing attacks. For our proposals, we show that such attacks can be avoided even though we do not follow the recommendations on tap positions.

We propose to use filtering function on a large number of variables. On the other hand, due to efficiency considerations, it is not possible to make the length of the LFSR too long. In particular, in all our concrete proposals, the ratio n/L is close to half and is much higher than what was used in all previous proposals. Using a high value of n/L means that the tap positions are placed much closer together than in previous proposals. This creates possibility of an overlap in the two sets of state bits determining two nearby keystream bits. Such overlap has the potential to cancel out terms and reduce the overall "complexity". Our choice of tap positions is motivated by the requirement of ensuring that such cancellations do not take place. The generation of a keystream bit requires a call to the majority function. We set one of the goals of choosing the tap positions to be to ensure that the inputs to the two calls to majority for generating two different keystream bits have only a small overlap. In particular, we *prove* that for our choice of tap positions, there is no cancellation of terms of the majority function corresponding to two keystream bits. We evolved this design criterion for the tap positions in response to an attack [6] on an earlier version, and based on informal attack ideas (though not actual attacks) suggested by Subhadeep Banik, Willi Meier, and Bin Zhang also on a previous version. Further, we choose all the tap positions for the variables in $\mathbf{X}$ to the left of all the tap positions for the variables in $\mathbf{Y}$. This choice combined with the choice of π as the bit reversal permutation allows us to *prove* that the quadratic terms arising from the application of the filtering function to the state bits corresponding to two different states do not cancel out with each other. More generally we introduce a new idea, which we call the *shift overlap minimisation* strategy, of selecting tap positions, the principle being to try and minimise the maximum overlap that arises due to shifts.

We perform a detailed concrete security analysis of some of the well known attacks on the nonlinear filter model. As the outcome of this analysis, for various security levels, we provide concrete proposals for stream ciphers based on the nonlinear filter model using the Boolean functions described above as the filtering functions. A strong point in favour of these proposals is that at the appropriate security levels they provide *provable* assurance against well known classes of attacks. Further, we provide concrete gate count estimates for the entire circuit to implement the stream ciphers. For the 80-bit, 128-bit, 160-bit, 192-bit, 224-bit and the 256-bit security levels, we propose using LFSRs of lengths 163, 257, 331, 389, 449, 521 and filtering functions on 75, 119, 143, 175, 203 and 231 variables respectively. The gate count estimates for the 80-bit, 128-bit, 160-bit, 192-bit, 224-bit and the 256-bit security levels are 1743.5, 2771.5, 3520.5, 4188.5, 4854.5, and 5607.5 NAND gates respectively. The gate count estimates for the 80-bit and the 128-bit security levels compare quite well[1] with famous ciphers such as Trivium [9] and Grain-128a [1] which offer 80-bit and 128-bit security respectively. For the other security levels, we are not aware of other stream ciphers which have such low gate counts. So our revival of the nonlinear filter model of stream ciphers leads to concrete proposals which offer a combination of both provable security against well known classes of attacks at a desired level of security and also low gate count.

We note that there are some old (prior to the advent of algebraic attacks) works [50, 51, 30] on efficient implementation of Boolean functions on a large number of variables targeted towards the nonlinear combiner model of stream ciphers. There are also a few later works [46, 18] on implementation of Maiorana-McFarland type functions. These works, however, do not cover the implementation of the functions that we introduce, the reason being that these new functions themselves do not appear earlier in the literature.

Comparison between the nonlinear filter model and some modern stream ciphers. Well known stream ciphers such as Trivium [9], Mickey [4], Grain-128a [1], SNOW [26] and Sosemanuk [5], use novel and ingenious ideas. Nonetheless, these are standalone designs. The nonlinear filter model, on the other hand, is a *model* for stream cipher design. Due to the simplicity of the nonlinear filter model, provable properties of the filtering function can be translated into provable resistance of the stream cipher against well known classes of attacks. In particular, for the proposals that we put forward, the provable linear bias of the filtering function translates to provable protection against a large class of fast correlation attacks, and the provable lower bound on the (fast) algebraic immunity translates to provable resistance against algebraic attacks. Stream ciphers based on either nonlinear finite state machines, or nonlinear feedback shift registers do not enjoy this advantage, i.e. for such stream ciphers it is very hard to obtain provable guarantees against various well known types of attacks. As an example, our proposal at the 128-bit security level generates keystream for which the best linear approximation has *provable* linear bias of 2^{-60}, while at the same security level, for Sosemanuk [5] the best known [39] linear approximation has correlation $2^{-20.84}$, and it is not known whether there are approximations with higher correlations. Similarly, the time complexities of fast algebraic attacks against Trivium, Mickey, Grain-128a, SNOW and Sosemanuk are not known. It is believed that these stream ciphers can withstand fast algebraic attacks. In contrast, we are able to *prove* that at the appropriate security levels, the fast algebraic attack is ineffective against the new nonlinear filter model based stream cipher proposals that we put forward.

One advantage of using LFSRs is that it is possible to provably ensure that the LFSR has a maximum period. For stream ciphers based on NFSRs, such as Trivium and Mickey, such provable assurance is not available. For stream ciphers which use a combination of LFSR and NFSR such as Grain-128a, it is possible to mount a divide-and-conquer attack. For example the attack in [54] on Grain-128a finds the state of the LFSR independently of the state of the NFSR. (In some ways this is reminiscent of the attack by Siegenthaler [53] on the nonlinear combiner model.) So even though Grain-128a uses a 256-bit state, due to the divide-and-conquer strategy the full protection of the large state is not achieved. On the other hand, for the nonlinear filter model, there is no known way to mount a divide-and-conquer attack. Our proposal at the 128-bit security level uses a 257-bit LFSR, and there is knwon no way to

[1]Following [1] we estimated 8 NAND gates for a flip-flop, whereas Trivium estimated 12 NAND gates for a flip-flop. Using 12 NAND gates for a flip-flop, our proposal at the 80-bit security level requires about 2395.5 NAND gates, whereas Trivium requires about 3488 NAND gates.

estimate half the state of the LFSR without involving the other half.

Lastly, we note that the nonlinear filter model provides a scalable design, while it is not clear how to scale the ideas behind standalone designs such as those in [9, 4, 1, 26, 5]. By properly choosing the LFSR and the filtering function, the nonlinear filter model can be instantiated to various security levels. This provides a *family* of stream ciphers rather than a single stream cipher. The scalability of the design makes it easier to target different security levels and also to ramp up parameters in response to improvements of known attacks. For example, at the 128-bit security level, the complexity of the fast algebraic attack on the proposal that we put forward is more than $2^{130.12}$, and the correlation of the best linear approximation of the keystream is 2^{-60}. By increasing the gate count by about 47 gates, it is possible to ensure that the complexity of the fast algebraic attack is more than $2^{135.83}$ and the best linear approximation of the keystream is 2^{-64}.

Important note. An earlier version of the proposal was attacked [6] using a differential attack. In response, we have modified the proposal to resist the attack in [6]. Though the modified proposal described in this version of the paper resists the attack in [6], we do not have any proof that the present proposal resists all kinds of differential attacks. We note that while provable properties of the filtering function can be translated to provable resistance against certain classes of attacks, it is by no means true that these properties provide resistance against *all* classes of attacks. We welcome further analysis and investigation of other avenues of attacks on the concrete stream cipher proposals that we put forward in this paper.

The paper is organised as follows. In Section 2 we describe the preliminaries. The Boolean function construction is described in Section 3. Section 4 performs the concrete security analysis and Section 5 provides the gate count estimates. Finally, Section 6 concludes the paper.

2 Preliminaries

This section provides the notation and the basic definitions. For details on Boolean functions we refer to [14].

By $\#S$ we will denote the cardinality of a finite set S. The finite field of two elements will be denoted by $\mathbb{F}_2$, and for a positive integer n, $\mathbb{F}_2^n$ will denote the vector space of dimension n over $\mathbb{F}_2$. By $\oplus$, we will denote the addition operator over both $\mathbb{F}_2$ and $\mathbb{F}_2^n$. An element of $\mathbb{F}_2^n$ will be considered to be an n-bit binary string.

For $n \geq 0$, let $\mathbf{x} = (x_1, \ldots, x_n)$ be an n-bit binary string. The support of $\mathbf{x}$ is $\mathsf{supp}(\mathbf{x}) = \{1 \leq i \leq n : x_i = 1\}$, and the weight of $\mathbf{x}$ is $\mathsf{wt}(\mathbf{x}) = \#\mathsf{supp}(\mathsf{supp}(\mathbf{x}))$. By $\mathbf{0}_n$ and $\mathbf{1}_n$ we will denote the all-zero and all-one strings of length n respectively. Let $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ be two n-bit strings. The distance between $\mathbf{x}$ and $\mathbf{y}$ is $d(\mathbf{x}, \mathbf{y}) = \#\{i : x_i \neq y_i\}$; the inner product of $\mathbf{x}$ and $\mathbf{y}$ is $\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 \oplus \cdots \oplus x_n y_n$; and we define $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for $i = 1, \ldots, n$.

An n-variable Boolean function f is a map $f : \mathbb{F}_2^n \to \mathbb{F}_2$. The weight of f is $\mathsf{wt}(f) = \#\{\mathbf{x} \in \mathbb{F}_2^n : f(\mathbf{x}) = 1\}$; f is said to be *balanced* if $\mathsf{wt}(f) = 2^{n-1}$.

Algebraic normal form. Let f be an n-variable function. The *algebraic normal form (ANF) representation* of f is the following: $f(X_1, \ldots, X_n) = \bigoplus_{\boldsymbol{\alpha} \in \mathbb{F}_2^n} a_{\boldsymbol{\alpha}} \mathbf{X}^{\boldsymbol{\alpha}}$, where $\mathbf{X} = (X_1, \ldots, X_n)$; for $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_2^n$, $\mathbf{X}^{\boldsymbol{\alpha}}$ denotes the monomial $X_1^{\alpha_1} \cdots X_n^{\alpha_n}$; and $a_{\boldsymbol{\alpha}} \in \mathbb{F}_2$. The (algebraic) degree of f is $\mathsf{deg}(f) = \max\{\mathsf{wt}(\boldsymbol{\alpha}) : a_{\boldsymbol{\alpha}} = 1\}$. Functions of degree at most 1 are said to be affine functions. Affine functions with $a_{\mathbf{0}_n} = 0$ are said to be linear functions. It is known that if f is balanced, then $\mathsf{deg}(f) \leq n - 1$.

We have the following relations between the coefficients $a_{\boldsymbol{\alpha}}$ in the ANF of f and the values of f (see for example Section 2.2 of [14]). For $\mathbf{x}, \boldsymbol{\alpha} \in \mathbb{F}_2^n$,

$$f(\mathbf{x}) = \bigoplus_{\boldsymbol{\beta} \leq \mathbf{x}} a_{\boldsymbol{\beta}} \quad \text{and} \quad a_{\boldsymbol{\alpha}} = \bigoplus_{\mathbf{z} \leq \boldsymbol{\alpha}} f(\mathbf{z}). \tag{1}$$

Nonlinearity and Walsh transform. For two n-variable functions f and g, the distance between them is $d(f,g) = \#\{\mathbf{x} \in \mathbb{F}_2^n : f(\mathbf{x}) \neq g(\mathbf{x})\}$. The *nonlinearity* of an n-variable function f is $\mathsf{nl}(f) = \min d(f,g)$, where the minimum is over all n-variable affine functions g.

The Walsh transform of an n-variable function f is the map $W_f : \mathbb{F}_2^n \to \mathbb{Z}$, where for $\boldsymbol{\alpha} \in \mathbb{F}_2^n$, $W_f(\boldsymbol{\alpha}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) \oplus \langle \boldsymbol{\alpha}, \mathbf{x} \rangle}$. The nonlinearity of a function f is given by its Walsh transform as follows: $\mathsf{nl}(f) = 2^{n-1} - \frac{1}{2} \max_{\boldsymbol{\alpha} \in \mathbb{F}_2^n} |W_f(\boldsymbol{\alpha})|$.

A function f such that $W_f(\boldsymbol{\alpha}) = \pm 2^{n/2}$ for all $\boldsymbol{\alpha} \in \mathbb{F}_2^n$ is said to be a bent function [48]. Clearly such functions can exist only if n is even. The nonlinearity of an n-variable bent function is $2^{n-1} - 2^{n/2-1}$ and this is the maximum nonlinearity that can be attained by n-variable functions.

A function is said to be plateaued if its Walsh transform takes only the values $0, \pm v$, for non-zero v. We define the linear bias of an n-variable Boolean function f to be $\mathsf{LB}(f) = 1/2 - \mathsf{nl}(f)/2^n$.

Algebraic resistance. Let f be an n-variable function. The *algebraic immunity* of f is defined [23, 44] as follows: $\mathsf{AI}(f) = \min_{g \neq 0}\{\mathsf{deg}(g) :$ either $gf = 0$, or $g(f \oplus 1) = 0\}$. It is known [23] that $\mathsf{AI}(f) \leq \lceil n/2 \rceil$.

The fast algebraic attack (FAA) was introduced in [22]. The idea of the attack is based on the following observation. Let f be an n-variable function and suppose g is another n-variable function of degree e such that gf has degree d. If both e and d are small, then f is susceptible to an FAA. Given f, and for e and d satisfying $e + d \geq n$, it is known [22] that there exists functions g and h with $\mathsf{deg}(g) = e$ and $\mathsf{deg}(h) \leq d$ such that $gf = h$. For any pair of functions g and h of degrees e and d respectively, satisfying $gf = h$, we have $h = gf = gf^2 = (gf)f = hf$ and so if $h \neq 0$, then $(1 \oplus h)$ is an annihilator of f which implies $d \geq \mathsf{AI}(f)$. If $1 \leq e < \mathsf{AI}(f)$, then g is not an annihilator of f and so $h = gf \neq 0$, and we obtain $e + d \geq 1 + \mathsf{AI}(f)$. The *fast algebraic immunity (FAI)* of f is a combined measure of resistance to both algebraic and fast algebraic attacks: $\mathsf{FAI}(f) = \min\left(2\mathsf{AI}(f), \min_{g \neq 0}\{\mathsf{deg}(g) + \mathsf{deg}(fg) : 1 \leq \mathsf{deg}(g) < \mathsf{AI}(f)\}\right)$. For any function f, $1 + \mathsf{AI}(f) \leq \mathsf{FAI}(f) \leq 2\,\mathsf{AI}(f)$.

Majority function. For $n \geq 1$, let $\mathsf{Maj}_n : \{0,1\}^n \to \{0,1\}$ be the majority function defined in the following manner. For $\mathbf{x} \in \{0,1\}^n$, $\mathsf{Maj}(\mathbf{x}) = 1$ if and only if $\mathsf{wt}(\mathbf{x}) > \lfloor n/2 \rfloor$. Clearly Maj_n is a symmetric function. The following results were proved in [24].

Theorem 1 (Theorems 1 and 2 of [24]) *Let n be a positive integer.*

1. Maj_n *has the maximum possible AI of* $\lceil n/2 \rceil$.
2. *The degree of* Maj_n *is equal to* $2^{\lfloor \log_2 n \rfloor}$.
3. *Any monomial occurring in the ANF of* Maj_n *has degree more than* $\lfloor n/2 \rfloor$.

3 Construction from Maiorana-McFarland Bent Functions

The Maiorana-McFarland class of bent functions is defined as follows. For $m \geq 1$, let $\pi : \{0,1\}^m \rightarrow \{0,1\}^m$ be a bijection and $h : \{0,1\}^m \rightarrow \{0,1\}$ be a Boolean function. Let $\pi_1, \ldots, \pi_m$ be the coordinate functions of π. Let $\mathbf{X} = (X_1, \ldots, X_m)$ and $\mathbf{Y} = (Y_1, \ldots, Y_m)$. For $m \geq 1$, MM_{2m} is defined to be the following.

$$\mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y}) \;=\; \langle \pi(\mathbf{X}), \mathbf{Y} \rangle \oplus h(\mathbf{X}) = \pi_1(\mathbf{X})Y_1 \oplus \cdots \oplus \pi_m(\mathbf{X})Y_m \oplus h(\mathbf{X}). \quad (2)$$

Since MM_{2m} is bent, $\mathsf{nl}(\mathsf{MM}_{2m}) = 2^{2m-1} - 2^{m-1}$, and $\mathsf{LB}(\mathsf{MM}_{2m}) = 2^{-m-1}$. Note that the nonlinearity of MM_n does not depend on the choices of the bijection π and the function h. The degree of MM_{2m} is given by the following result.

Proposition 1 *For $m \geq 1$, $\mathsf{deg}(\mathsf{MM}_{2m}) = \max(\mathsf{deg}(\pi_1) + 1, \ldots, \mathsf{deg}(\pi_m) + 1, \mathsf{deg}(h))$.*

To the best of our knowledge the following result on the algebraic immunity of MM_{2m} is new.

Theorem 2 *Suppose $m \geq 1$. There is an $\boldsymbol{\omega}^\star \in \mathbb{F}_2^m$ such that*

$$\mathsf{AI}(\mathsf{MM}_{2m}) \;\geq\; \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{AI}\big(\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})\big).$$

Suppose that π is an affine map, i.e. for $1 \leq i \leq m$, $\pi_i(X_1, \ldots, X_m)$ is an affine function. Then

$$\mathsf{AI}(\mathsf{MM}_{2m}) \;\geq\; \mathsf{AI}(h).$$

Proof: Suppose $g(\mathbf{X}, \mathbf{Y})$ is an annihilator for $\mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y})$. Recall that for $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m) \in \mathbb{F}_2^m$, by $\mathbf{Y}^{\boldsymbol{\omega}}$ we denote the monomial $Y_1^{\omega_1} \cdots Y_m^{\omega_m}$. Using this notation, we write $g(\mathbf{X}, \mathbf{Y}) = \bigoplus_{\boldsymbol{\omega} \in \mathbb{F}_2^m} \mathbf{Y}^{\boldsymbol{\omega}} g_{\boldsymbol{\omega}}(\mathbf{X})$, for some functions $g_{\boldsymbol{\omega}}(\mathbf{X})$'s. We have

$$\begin{aligned} 0 &= g(\mathbf{X}, \mathbf{Y})\mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y}) \\ &= \left(\bigoplus_{\boldsymbol{\omega} \in \mathbb{F}_2^m} \mathbf{Y}^{\boldsymbol{\omega}} g_{\boldsymbol{\omega}}(\mathbf{X}) \right) \big(\pi_1(\mathbf{X})Y_1 \oplus \cdots \oplus \pi_m(\mathbf{X})Y_m \oplus h(\mathbf{X}) \big). \end{aligned} \quad (3)$$

Since the right hand side of (3) is equal to 0, for $\boldsymbol{\omega} \in \mathbb{F}_2^m$, the coefficient of $\mathbf{Y}^{\boldsymbol{\omega}}$ in the expansion on the right hand side of (3) must be equal to 0. Since $g(\mathbf{X}, \mathbf{Y}) \neq 0$, let $w \geq 0$ be the minimum integer such that there is an $\boldsymbol{\omega}^\star$ with $\mathsf{wt}(\boldsymbol{\omega}^\star) = w$ and $g_{\boldsymbol{\omega}^\star}(\mathbf{X}) \neq 0$. In (3), equating the coefficient of $\mathbf{Y}^{\boldsymbol{\omega}^\star}$ to 0, we have

$$\begin{aligned} 0 &= g_{\boldsymbol{\omega}^\star}(\mathbf{X}) \left(h(\mathbf{X}) \oplus \left(\bigoplus_{i \in \mathsf{supp}(\boldsymbol{\omega}^\star))} \pi_i(\mathbf{X}) \right) \right) \\ &= g_{\boldsymbol{\omega}^\star}(\mathbf{X}) \big(\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \big). \end{aligned}$$

So $g_{\boldsymbol{\omega}^\star}(\mathbf{X})$ is an annihilator for $\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})$. Consequently, $\mathsf{deg}(g) \geq \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{deg}(g_{\boldsymbol{\omega}^\star}) \geq \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{AI}\big(\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})\big)$.

If, on the other hand, $g(\mathbf{X}, \mathbf{Y})$ is an annihilator for $1 \oplus \mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y})$, then a similar argument shows that $g_{\boldsymbol{\omega}^\star}(\mathbf{X})$ is an annihilator for $\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus 1 \oplus h(\mathbf{X})$, and again we have $\mathsf{deg}(g) \geq \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{AI}\big(\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})\big)$.

Suppose now that π is an affine map. Then $\ell(\mathbf{X}) = \langle \boldsymbol{\omega}^\star, \pi(\mathbf{X})\rangle$ is an affine function. If $w = 0$, then $\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X})\rangle \oplus h(\mathbf{X}) = h(\mathbf{X})$, and so we have the result. So suppose $w > 0$. From Lemma 1 of [16], we have $\mathsf{AI}(\ell(\mathbf{X}) \oplus h(\mathbf{X})) \geq \mathsf{AI}(h(\mathbf{X})) - 1$. So

$$\begin{aligned}\mathsf{AI}(\mathsf{MM}_{2m}) &\geq \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{AI}\big(\ell(\mathbf{X}) \oplus h(\mathbf{X})\big) \\ &\geq w + \mathsf{AI}(h(\mathbf{X})) - 1 \geq \mathsf{AI}(h(\mathbf{X})).\end{aligned}$$

□

Extension to MM_n, for odd n. We consider a very well known extension of MM_{2m} to odd number of variables.

$$\begin{aligned}\mathsf{MM}_1(W) &= W, \\ \mathsf{MM}_{2m+1}(W, \mathbf{X}, \mathbf{Y}) &= W \oplus \mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y}), \quad \text{for } m \geq 1.\end{aligned} \tag{4}$$

The following result states the properties of MM_{2m+1}.

Proposition 2 *For $m \geq 1$, MM_{2m+1} is balanced, and*

1. $\mathsf{nl}(\mathsf{MM}_{2m+1}) = 2^{2m} - 2^m$. *In particular,* $\mathsf{LB}(\mathsf{MM}_{2m+1}) = \mathsf{LB}(\mathsf{MM}_{2m}) = 2^{-(m+1)}$.
2. $\mathsf{deg}(\mathsf{MM}_{2m+1}) = \mathsf{deg}(\mathsf{MM}_{2m})$.
3. $\mathsf{AI}(\mathsf{MM}_{2m}) \leq \mathsf{AI}(\mathsf{MM}_{2m+1}) \leq 1 + \mathsf{AI}(\mathsf{MM}_{2m})$.

Proof: The first point is well known. The second point is immediate from the definition of MM_{2m+1}. We provide the proof of the third point. For brevity, let us write $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$. Clearly if $g(\mathbf{Z})$ is an annihilator for MM_{2m} (resp. $1 \oplus \mathsf{MM}_{2m}$), then $(1 \oplus W)g(\mathbf{Z})$ is an annihilator for MM_{2m+1} (resp. $1 \oplus \mathsf{MM}_{2m+1}$). This shows the upper bound. Next we consider the lower bound. Suppose $g(W, \mathbf{Z}) \neq 0$ is an annihilator for $\mathsf{MM}_{2m+1}(W, \mathbf{Z})$. We write $g(W, \mathbf{Z})$ as $g(W, \mathbf{Z}) = Wg_1(\mathbf{Z}) + g_0(\mathbf{Z})$. Noting that $\mathsf{MM}_{2m+1}(W, \mathbf{Z}) = W \oplus \mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y})$, we obtain

$$\begin{aligned}0 &= g(W, \mathbf{Z})\mathsf{MM}_{2m+1}(W, \mathbf{Z}) \\ &= g_0(\mathbf{Z})\mathsf{MM}_{2m}(\mathbf{Z}) \oplus W\big(g_0(\mathbf{Z}) \oplus g_1(\mathbf{Z})(1 \oplus \mathsf{MM}_{2m}(\mathbf{Z}))\big).\end{aligned}$$

So $g_0(\mathbf{Z})\mathsf{MM}_{2m}(\mathbf{Z}) = 0$ and $g_0(\mathbf{Z}) \oplus g_1(\mathbf{Z})(1 \oplus \mathsf{MM}_{2m}(\mathbf{Z})) = 0$. If g_0 is non-zero, then g_0 is an annihilator for MM_{2m} and so $\mathsf{deg}(g) \geq \mathsf{deg}(g_0) \geq \mathsf{AI}(\mathsf{MM}_{2m})$. If $g_0 = 0$, then since $g \neq 0$, it follows that $g_1 \neq 0$. In this case, g_1 is an annihilator for $1 \oplus \mathsf{MM}_{2m}$, and so $\mathsf{deg}(g) \geq 1 + \mathsf{deg}(g_1) \geq 1 + \mathsf{AI}(\mathsf{MM}_{2m})$. Consequently, in both cases $\mathsf{deg}(g) \geq \mathsf{AI}(\mathsf{MM}_{2m})$.

On the other hand, if $g(W, \mathbf{Z}) \neq 0$ is an annihilator for $1 \oplus \mathsf{MM}_{2m+1}(W, \mathbf{Z})$, then noting that $W(1 \oplus W) = 0$, we obtain $g_0(\mathbf{Z})(1 \oplus \mathsf{MM}_{2m}(\mathbf{Z})) = 0$ and $g_0(\mathbf{Z}) \oplus g_1(\mathbf{Z})\mathsf{MM}_{2m}(\mathbf{Z}) = 0$. If $g_0 \neq 0$, then g_0 is an annihilator for $1 \oplus \mathsf{MM}_{2m}$, and if $g_0 = 0$, then g_1 is an annihilator for MM_{2m}. So again we have $\mathsf{deg}(g) \geq \mathsf{AI}(\mathsf{MM}_{2m})$. □

From Theorem 2, choosing π to be any affine permutation results in the AI of MM_{2m} to be lower bounded by the AI of h. We make the following concrete choices.

> *Concrete choice of π in MM_{2m}*: Choose the m-bit to m-bit permutation π in the construction of MM_{2m} given by (2) to be the bit reversal permutation rev, i.e. $\pi(X_1, \ldots, X_m) = \mathsf{rev}(X_1, \ldots, X_m) = (X_m, \ldots, X_1)$.
>
> *Concrete choice of h in MM_{2m}*: Choose the m-variable function h in the construction of MM_{2m} given by (2) to be Maj_m, which is the m-variable majority function.

Our choice of Maj_m for h is motivated by the fact that the majority function has the maximum possible algebraic immunity (see Theorem 1). There are other functions which achieve maximum algebraic immunity [16] and these could also be chosen to instantiate h. Our choice of Maj_m is the simplest choice of a function achieving maximum algebraic immunity.

By $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_{2m}$ we denote the function obtained by instantiating the definition of MM_{2m} given by (2) with $h = \mathsf{Maj}_m$ and $\pi = \mathsf{rev}$. Similarly, by $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_{2m+1}$, we denote the function obtained from $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_{2m}$ using (4).

Proposition 3 *For $n \geq 4$, the degree of $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_n$ is $2^{\lfloor \log_2 \lfloor n/2 \rfloor \rfloor}$.*

Proof: The degree of Maj_m is $2^{\lfloor \log_2 m \rfloor}$ (see Theorem 1). Applying Proposition 1, we obtain the required result for even n. For odd n, the result follows from the second point of Proposition 2. $\square$

Proposition 4 *For $n \geq 4$, if n is even, then $\mathsf{AI}((\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_n) \geq \lceil n/4 \rceil$, and if n is odd then $\mathsf{AI}((\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_n) \geq \lceil (n-1)/4 \rceil$.*

Proof: Suppose $n = 2m$ is even. From Theorem 2 we have $\mathsf{AI}((\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_{2m}) \geq \mathsf{AI}(\mathsf{Maj}_m)$. Since $\mathsf{AI}(\mathsf{Maj}_m) = \lceil m/2 \rceil$, we have the result. For odd $n = 2m+1$, from Proposition 2, we have $\mathsf{AI}((\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_{2m+1} \geq \mathsf{AI}((\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_{2m}$, which shows the result. $\square$

We computed the algebraic immunity of $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_n$ for small values of n and observed that for $n \geq 4$, $\mathsf{AI}((\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_n) = 1 + \lfloor n/4 \rfloor$. This suggests that the lower bound in Proposition 4 is exact if $n \not\equiv 0 \bmod 4$ and if $n \equiv 0 \bmod 4$, the actual algebraic immunity is one more than the lower bound. In a work [41] subsequent to the present paper, it was shown that for even n, $\mathsf{AI}((\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_n) \leq 1 + 2^{\lceil \log(n/4) \rceil}$. As mentioned in Section 2, for any Boolean function f, $\mathsf{FAI}(f) \geq 1 + \mathsf{AI}(f)$. Our experiments suggest that for $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_n$, the fast algebraic immunity is actually one more than the algebraic immunity.

So $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_n$ does not have the best possible algebraic immunity. On the other hand, it can be implemented using $O(m)$ gates (as we show in Section 5). So it is possible to increase the number of variables to achieve the desired level of algebraic resistance without increasing the circuit size too much.

From Proposition 2 we have MM_{2m+1} is balanced, and the cryptographic resistance provided by MM_{2m+1} is very similar to that provided by MM_{2m}. In particular, the linear bias of MM_{2m+1} is the same as that of MM_{2m}, and the algebraic resistance of MM_{2m+1} is at least that of MM_{2m}. In terms of implementation efficiency, MM_{2m+1} requires only one extra XOR gate in addition to the circuit to implement MM_{2m}. We consider $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_{2m+1}$, i.e. where h is chosen to be Maj_m and π is chosen to be rev. The function MM_{2m+1} maps the all-zero string to 0. This can be a problem for use in a linear system, since such a system will map the all-zero string to the all-zero string. We modify MM_{2m+1} so that the all-zero string is mapped to 1. We define $f_{2m+1} : \{0,1\}^{2m+1} \to \{0,1\}$ to be

$$f_{2m+1}(W, \mathbf{X}, \mathbf{Y}) = 1 \oplus \mathsf{MM}_{2m+1}(W, \mathbf{X}, \mathbf{Y}) = 1 \oplus W \oplus \langle \mathsf{rev}(\mathbf{X}), \mathbf{Y} \rangle \oplus \mathsf{Maj}_m(\mathbf{X}). \quad (5)$$

The cryptographic properties of f_{2m+1} are exactly the same as those of MM_{2m+1}. In the next section, we propose the use of f_{2m+1} in the construction of the filter generator model of stream ciphers.

4 Concrete Stream Cipher Proposals

We revisit the filter generator model where the state of an LFSR of length L is filtered using a Boolean function. *By $\mathcal{S}(L, m)$, we denote the class of stream ciphers obtained from an LFSR of length L with a* *primitive connection polynomial, and with f_{2m+1} as the filtering function.* For the κ-bit security level, we assume that the stream cipher supports a κ-bit secret key and a κ-bit initialisation vector (IV), and so $L \geq 2\kappa$. Below we describe the construction of $\mathcal{S}(L,m)$ which requires specifying a few more parameters in addition to L and m.

LFSR maps. Let $\tau(x) = x^L \oplus c_{L-1}x^{L-1} \oplus \cdots \oplus c_1 x \oplus c_0$ be a primitive polynomial of degree L over $\mathbb{F}_2$. Using τ, we define the linear next bit map $\mathsf{nb} : \{0,1\}^L \rightarrow \{0,1\}$ and the linear next state map $\mathsf{NS} : \{0,1\}^L \rightarrow \{0,1\}^L$ as follows.

$$\begin{aligned} \mathsf{nb}(u_{L-1},\ldots,u_1,u_0) &= c_{L-1}u_{L-1} \oplus \cdots \oplus c_1 u_1 \oplus c_0 u_0, \\ \mathsf{NS}(u_{L-1},\ldots,u_1,u_0) &= (\mathsf{nb}(u_{L-1},\ldots,u_1,u_0), u_{L-1},\ldots,u_1). \end{aligned}$$

Filter function and tap positions. The function $f_{2m+1}(W,\mathbf{X},\mathbf{Y})$, where $\mathbf{X} = (X_1,\ldots,X_m)$ and $\mathbf{Y} = (Y_1,\ldots,Y_m)$, is a function on $2m+1$ variables. This function will be used as the filtering function and applied to a subset of the bits of the L-bit state. We next describe the tap positions of the L-bit state which provides the input bits to f_{2m+1}.

Let $i_1,\ldots,i_m,j_1,\ldots,j_m$ be integers satisfying the following condition.

$$0 = i_1 < i_2 < \cdots < i_{m-1} < i_m < \kappa \leq j_1 < j_2 < \cdots < j_{m-1} < j_m = 2\kappa - 2. \tag{6}$$

Given the integers $i_1,\ldots,i_m$ and $j_1,\ldots,j_m$ we define a function $\mathsf{proj} : \{0,1\}^L \rightarrow \{0,1\}^{2m+1}$ in the following manner.

$$\mathsf{proj}(u_{L-1},\ldots,u_0) = (u_{L-2\kappa}, u_{L-1-i_1},\ldots,u_{L-1-i_m}, u_{L-1-j_1},\ldots,u_{L-1-j_m}). \tag{7}$$

The function proj extracts the subset of $2m+1$ bits from the L-bit state to which the function f_{2m+1} is to be applied. We define the composition $f_{2m+1} \circ \mathsf{proj}$ to be the following L-variable Boolean function.

$$(u_{L-1},\ldots,u_0) \mapsto f_{2m+1}(\mathsf{proj}(u_{L-1},\ldots,u_0)). \tag{8}$$

Note that the tap position for W is $L-2\kappa$, the tap position for X_p is $L-1-i_p$, and the tap position for $Y_{p'}$ is $L-1-j_{p'}$, where $1 \leq p,p' \leq m$. There is no tap position in the rightmost $L-2\kappa$ positions of the state; the leftmost position (i.e. position $L-1$) of the state is the tap position for X_1, all the tap positions for the variables in $\mathbf{X}$ are in the leftmost κ positions of the state; all the tap positions for the variables in $\mathbf{Y}$ are in the next κ positions of the state, and the position $L-2\kappa+1$ of the state is the tap position for Y_m. We have

$$\begin{aligned} & f_{2m+1}(\mathsf{proj}(u_{L-1},\ldots,u_0)) \\ &= 1 \oplus u_{L-2\kappa} \oplus \left(\bigoplus_{p=1}^{m} u_{L-1-i_p} u_{L-1-j_{m+1-p}} \right) \oplus \mathsf{Maj}_m(u_{L-1-i_1},\ldots,u_{L-1-i_m}) \\ &= 1 \oplus u_{L-2\kappa} \oplus u_{L-1-i_m}u_{L-1-j_1} \oplus \cdots \oplus u_{L-1-i_1}u_{L-1-j_m} \oplus \mathsf{Maj}_m(u_{L-1-i_1},\ldots,u_{L-1-i_m}). \end{aligned} \tag{9}$$

Let $\mathsf{pos}[L-1,\ldots,0]$ be an L-bit string such that

$\mathsf{pos}[L-1-i_p] = 1$, for $1 \leq p \leq m$;
$\mathsf{pos}[L-1-j'_p] = 1$, for $1 \leq p' \leq m$;
$\mathsf{pos}[L-2\kappa] = 1$;
and pos is 0 at all the other $L-2m-1$ positions.

Note that the indexing of pos has $L-1$ as the leftmost position and 0 as the rightmost position. The string pos encodes the $2m+1$ tap positions of the state. Let posX be the leftmost κ-bit string of pos, i.e. the segment of pos given by $\mathsf{pos}[L-1,\ldots,L-\kappa]$. The string posX encodes the tap positions of the variables in $\mathbf{X}$.

Remark 1 *In the filtering function f_{2m+1}, the particular choice of the bit permutation to instantiate π is to be seen in conjunction with the choice of the tap positions for the variables $X_1,\ldots,X_m$. We have chosen π to be the bit reversal permutation* rev*. The tap positions for $X_1,\ldots,X_m$ are chosen keeping* rev *in the mind. If instead of* rev*, we choose some other bit permutation for π, then a corresponding permutation has to be applied to the tap positions for $X_1,\ldots,X_m$.*

Initialisation phase. The stream cipher uses a κ-bit key $(k_{\kappa-1},\ldots,k_0)$ and a κ-bit initialisation vector $(v_{\kappa-1},\ldots,v_0)$. Let $\mathbf{b}=(b_{L-2\kappa-1},\ldots,b_0)$ be an $(L-2\kappa)$-bit string defined in the following manner: if $L-2\kappa$ is even, then $\mathbf{b}=(01)^{(L-2\kappa)/2}$, and if $L-2\kappa$ is odd, then $\mathbf{b}=1(01)^{(L-2\kappa-1)/2}$. The initial L-bit state $\mathbf{s}$ of the stream cipher is

$$\mathbf{s} \;=\; (k_{\kappa-1},\ldots,k_0,v_{\kappa-1},\ldots,v_0,b_{L-2\kappa-1},\ldots,b_0). \tag{10}$$

The initialisation round function is the map $\mathsf{IR}:\{0,1\}^L\rightarrow\{0,1\}^L$, where $\mathsf{IR}(u_{L-1},\ldots,u_0)$ is defined in the following manner. Let $c=\mathsf{nb}(u_{L-1},\ldots,u_0)$ and $b=f_{2m+1}(\mathsf{proj}(u_{L-1},\ldots,u_0))$. Then $\mathsf{IR}(u_{L-1},\ldots,u_0)=(w_{L-1},\ldots,w_0)$, where $w_{L-1}=c\oplus b$, and for $i=0,\ldots,L-2$,

$$w_i \;=\; \begin{cases} u_{i+1}\oplus b & \text{if } i\geq L-2\kappa \text{ and } i\equiv 0 \bmod \mu,\\ u_{i+1} & \text{otherwise.}\end{cases}$$

In the above, μ is a parameter of the initialisation round function. The round function feeds back the output of $f_{2m+1}\circ\mathsf{proj}$ to the state at $\lceil 2\kappa/\mu\rceil$ positions. This requires $\lceil 2\kappa/\mu\rceil$ XOR gates. We propose

$$\mu \;=\; \lfloor\sqrt{2\kappa}\rfloor \tag{11}$$

so that the feedback does not require too many XOR gates.

The initialisation round function is applied to update the initial state $\mathbf{s}$ given in (10) a total of 2κ times as follows.

set $\mathbf{s}$ as in (10)
for $i\leftarrow 1$ to 2κ do
    $\mathbf{s}\leftarrow\mathsf{IR}(\mathbf{s})$
end for
output $\mathbf{s}$

Keystream generation phase. The generation of the keystream bits $z_0,z_1,z_2,\ldots$ is done as follows.

let $\mathbf{s}$ be the output of the initialisation phase
for $t\geq 0$ do
    $z_t\leftarrow f_{2m+1}(\mathsf{proj}(\mathbf{s}))$
    $\mathbf{s}\leftarrow\mathsf{NS}(\mathbf{s})$
end for.

4.1 Effect of Shifts on Tap Positions

In our concrete proposals, the number of variables $n = 2m + 1$ of the filtering function is almost about half the length L of the LFSR (see Table 1). In fact, the ratio n/L in our proposals is much higher than similar ratios in all previous proposals. Using a high value of n prevents certain well known attacks as we discuss later. On the other hand, the high (compared to previous proposals) value of n/L ratio means that the tap positions are placed close together. Since $L - 1$ bits of the next state of the LFSR are obtained by a shift of the previous state, two states of the LFSR which are close in time share a number of bits. Further, with a high value of n/L ratio, two keystream bits arising from two such nearby states may depend on a number of common state bits. This creates the possibility that in the XOR of two such keystream bits, a number of terms involving the state bits cancel out. A bad choice of the tap positions can indeed lead to such an effect, as is explained in Section 4.4. Below we show that such cancellations do not occur in our proposals.

No cancellation of quadratic terms. From (5), the filtering function $f_{2m+1}(W, \mathbf{X}, \mathbf{Y})$ has the term $\mathsf{Maj}_m(\mathbf{X})$. For $m \geq 6$, Theorem 1 assures us that the ANF of $\mathsf{Maj}_m(\mathbf{X})$ does not have any quadratic terms. All our concrete proposals have m (much) greater than 6, so we assume that $\mathsf{Maj}_m(\mathbf{X})$ does not contribute any quadratic term to the ANF of f_{2m+1}. Hence, the quadratic terms of f_{2m+1} arise from the inner product $\langle \mathsf{rev}(\mathbf{X}), \mathbf{Y} \rangle$.

When applied to the bits of the state, the inner product $\langle \mathsf{rev}(\mathbf{X}), \mathbf{Y} \rangle$ leads to m quadratic terms involving the state bits. Two states which are separated by d time periods share a number of bits. More precisely, the rightmost $L - d$ bits of the later state are the leftmost $L - d$ bits of the former state. This creates the possibility that some of the quadratic terms of the former state cancels with some of the quadratic terms of the later state. We show that due to our choice of the permutation π as the bit reversal permutation rev and the choice of placing the tap positions for $\mathbf{X}$ to the left of the tap positions for $\mathbf{Y}$, such cancellations do not occur.

Let $\mathbf{s}^{(t)}$ be the state at time point t as defined as in (13). The p-th quadratic term of $\mathbf{s}^{(t)}$ arising from $\langle \mathsf{rev}(\mathbf{X}), \mathbf{Y} \rangle$ is $s_{L+t-1-i_p} s_{L+t-1-j_{m+1-p}}$, where $1 \leq p \leq m$. Let $t_1 \geq 0$ and $d > 0$ be integers, and $t_2 = t_1 + d$. Let $\mathbf{s}^{(t_1)}$ and $\mathbf{s}^{(t_2)}$ be the states at time points t_1 and t_2 respectively (obtained from (13) by substituting t_1 and t_2 for t). If there are integers p and p' with $1 \leq p, p' \leq m$ such that the p-th quadratic term of $\mathbf{s}^{(t_1)}$ is identically equal to the p'-th quadratic term of $\mathbf{s}^{(t_2)}$, then we say that a cancellation of quadratic terms occur between $\mathbf{s}^{(t_1)}$ and $\mathbf{s}^{(t_2)}$.

Proposition 5 *No cancellation of quadratic terms occur between* $\mathbf{s}^{(t_1)}$ *and* $\mathbf{s}^{(t_2)}$.

Proof: The quadratic terms arising from $\mathbf{s}^{(t_1)}$ and $\mathbf{s}^{(t_2)}$ are

$$s_{L+t_1-1-i_p} s_{L+t_1-1-j_{m+1-p}} \quad \text{and} \quad s_{L+t_2-1-i_{p'}} s_{L+t_2-1-j_{m+1-p'}} \tag{12}$$

respectively, for $1 \leq p, p' \leq m$. Cancellation of the p-th quadratic term of $\mathbf{s}^{(t_1)}$ with the p'-th quadratic term of $\mathbf{s}^{(t_2)}$ occurs if $s_{L+t_1-1-i_p} s_{L+t_1-1-j_{m+1-p}}$ and $s_{L+t_2-1-i_{p'}} s_{L+t_2-1-j_{m+1-p'}}$ are identically equal. There are two cases to consider.

Case 1: This case corresponds to the possibility that $s_{L+t_1-1-i_p}$ and $s_{L+t_2-1-i_{p'}}$ are identically equal and further $s_{L+t_1-1-j_{m+1-p}}$ and $s_{L+t_2-1-j_{m+1-p'}}$ are identically equal. These two conditions are equivalent to $L + t_1 - 1 - i_p = L + t_2 - 1 - i_{p'}$ and $L + t_1 - 1 - j_{m+1-p} = L + t_2 - 1 - j_{m+1-p'}$, i.e. $i_{p'} = i_p + d$ and $j_{m+1-p'} = j_{m+1-p} + d$. Since $d > 0$, we have $i_{p'} > i_p$ and from the definition of $i_1, \ldots, i_m$, we obtain $p' > p$. Similarly, we have $j_{m+1-p'} > j_{m+1-p} + d$, and from the definition of $j_1, \ldots, j_m$, we obtain $m + 1 - p' > m + 1 - p$, i.e. $p > p'$, which is a contradiction to $p' > p$.

Case 2: This case corresponds to the possibility that $s_{L+t_1-1-i_p}$ and $s_{L+t_2-1-j_{m+1-p'}}$ are identically equal and further $s_{L+t_2-1-i_{p'}}$ and $s_{L+t_1-1-j_{m+1-p}}$ are identically equal. These two conditions are equivalent to $L+t_1-1-i_p = L+t_2-1-j_{m+1-p'}$ and $L+t_2-1-i_{p'} = L+t_1-1-j_{m+1-p}$, i.e. $j_{m+1-p'} = i_p + d$ and $i_{p'} = j_{m+1-p} + d$. Since $d > 0$, we obtain $i_{p'} > j_{m+1-p}$, but by definition $i_{p'} < j_{m+1-p}$ (see (6)), and so we obtain a contradiction. □

Suppose two keystream bits are XORed. Proposition 5 assures us that the quadratic terms arising in the expressions for these keystream bits do not cancel out with each other. During the keystream generation, suppose that at some point the state is $\mathbf{s}^{(0)} = (s_{L-1}, \ldots, s_0)$. For $t \geq 1$, define $s_{L+t-1} = \mathsf{nb}(s_{L+t-2}, \ldots, s_{t-1})$, and

$$\mathbf{s}^{(t)} = (s_{L+t-1}, s_{L+t-2}, \ldots, s_t). \tag{13}$$

No cancellation of terms of the majority function. As mentioned above, two calls to the majority function can have an overlap. This has the potential of some terms in the ANF of the majority function arising from one of the calls cancelling out with terms in the ANF of the majority function arising from the other calls. This is undesirable and reduces the "complexity". The next result provides a sufficient condition to prevent such undesirable cancellations.

Proposition 6 *Let m and p be positive integers with $1 \leq p < m$. Let $\mathbf{U} = (U_1, \ldots, U_p)$, $\mathbf{V} = (V_1, \ldots, V_{m-p})$ and $\mathbf{W} = (W_1, \ldots, W_{m-p})$. If $p \leq \lfloor m/2 \rfloor$, then no monomial in the ANF of $\mathsf{Maj}_m(\mathbf{U}, \mathbf{V})$ cancels with any monomial in the ANF of $\mathsf{Maj}_m(\mathbf{U}, \mathbf{W})$.*

Proof: From Theorem 1, we have that any monomial occuring in the ANF of $\mathsf{Maj}_m(\mathbf{U}, \mathbf{V})$ has degree more than $\lfloor m/2 \rfloor$. So any monomial occurring in the ANF of $\mathsf{Maj}_m(\mathbf{U}, \mathbf{V})$ must involve one of the V_i's, and similarly, any monomial occurring in the ANF of $\mathsf{Maj}_m(\mathbf{U}, \mathbf{W})$ must involve one of the W_j's. Since the ANF of $\mathsf{Maj}_m(\mathbf{U}, \mathbf{V})$ does not involve any W_j, and the ANF of $\mathsf{Maj}_m(\mathbf{U}, \mathbf{W})$ does not involve any V_i, there can be no cancellation of terms between the two ANFs. □

Proposition 6 assures us that if the overlap between two calls to majority is at most $\lfloor m/2 \rfloor$, then there is no cancellation of terms arising from the ANF corresponding to the two calls.

To generate keystream bits, the function $f \circ \mathsf{proj}$ is applied to $\mathbf{s}^{(0)}$ and $\mathbf{s}^{(t)}$. This in turn requires applying the majority function to a subset of the bits of $\mathbf{s}^{(0)}$ and $\mathbf{s}^{(t)}$. More precisely, the functions calls $\mathsf{Maj}(s_{L-1-i_1}, \ldots, s_{L-1-i_m})$ and $\mathsf{Maj}(s_{L+t-1-i_1}, \ldots, s_{L+t-1-i_m})$ occur corresponding to $\mathbf{s}^{(0)}$ and $\mathbf{s}^{(t)}$ respectively. If $t < \kappa$, then there is a possibility that the inputs to the two calls to majority have an overlap, i.e. some of the inputs are common to both the majority calls. Since majority is a symmetric function, the actual positions where the common inputs occur do not matter. For $t \geq 1$, let

$$\nu_t = \#\left(\{L-1-i_1, \ldots, L-1-i_m\} \cap \{L+t-1-i_1, \ldots, L+t-1-i_m\}\right). \tag{14}$$

Note that $\nu_t = 0$ for $t \geq \kappa$, and for $1 \leq t < \kappa$,

$$\nu_t = \mathsf{wt}(\mathsf{posX} \wedge (\mathsf{posX} \gg t)), \tag{15}$$

where $\wedge$ denotes the bitwise AND operation, and $\gg t$ denotes right shift by t places. Define

$$\nu = \max_{1 \leq t < \kappa} \nu_t = \max_{1 \leq t < \kappa} \mathsf{wt}(\mathsf{posX} \wedge (\mathsf{posX} \gg t)). \tag{16}$$

A design goal is to choose posX so as to minimise ν. In particular, if ν is at most $\lfloor m/2 \rfloor$, then from Proposition 6 there is no cancellation of terms arising from the two majority calls. Later we provide concrete values of ν (see Table 4) for our specific proposals. For all cases, the condition $\nu \leq \lfloor m/2 \rfloor$ holds, and so the no cancellation property of terms arising from the ANF of the majority function corresponding to the two calls is ensured.

Overlap of tap positions due to shifts. Let

$$\delta \quad = \quad \max_{1 \le t < 2\kappa} \mathsf{wt}(\mathsf{pos} \wedge (\mathsf{pos} \gg t)). \tag{17}$$

Then δ is the maximum overlap between the tap positions that can be obtained by shifting the state by any value.

Procedure for choosing the tap positions. Suppose κ, $L \ge 2\kappa$, and m are given. We need to determine $i_1, \ldots, i_m$ and $j_1, \ldots, j_m$, or equivalently, to determine pos. (Recall that the tap position for W is fixed to be $L - 2\kappa$.) The leftmost κ bits of pos is the string posX. The value of ν is determined by posX, while the value of δ is determined by pos. We would like to minimise ν as well as δ. It is difficult to simultaneously minimise both. From (19) (provided later), the value of δ is one of the factors relevant to protection against certain kinds of state guessing attacks. As explained in Remark 2 (also provided later), it is not essential to minimise δ. The parameter ν, on the other hand, determines the size of the maximum overlap between two inputs to the majority function corresponding to two different keystream bits. Minimising ν improves resistance to possible differential attacks (see Section 4.4). Since it is not essential to minimise δ, we chose to focus on minimising ν. The task of choosing posX such that ν is minimised is a combinatorial optimisation problem. There does not seem to be any good way to obtain posX such that the corresponding ν is *guaranteed* to be the minimum possible value. Instead, we adopted the following procedure. Given κ, L and m, make 10000 choices of pos satisfying the constraints on $i_1, \ldots, i_m$ and $j_1, \ldots, j_m$, and for each choice extract posX and then compute ν from posX. Finally return the pos for which the least value of ν is obtained. Then, for this pos compute the value of δ. Later we provide the values of pos, ν and δ for the concrete proposals that we make.

Note that both ν and δ denote the maximum overlaps (of different kinds) that can arise due to shifts. The idea of tap position selection that we put forward is to minimise ν and δ. While minimising both simultaneously is difficult, we call the general principle of minimising the maximum overlap due to shifts to be the *shift overlap minimisation (SOM)* strategy of selecting tap positions. Selecting the tap positions to be a full positive difference set is a particular case of the SOM principle, where the minimum overlap is zero. Since using tap positions which form a full positive difference set leads to the length of the shift register being too long, the SOM principle mandates that the maximum overlap due to shifts is as small as possible.

4.2 Security Analysis

The parameters $L \ge 2\kappa$, m, and $\mu = \lfloor \sqrt{2\kappa} \rfloor$ are the design parameters, as are the tap positions encoded by pos. The parameter ν is determined by posX, while the parameter δ is determined by pos. The primitive connection polynomial $\tau(x)$ of the LFSR is also a parameter, as is the number of non-zero terms in $\tau(x)$. For a concrete choice targeted at a specific security level, the parameters need to be appropriately chosen so as to provide resistance against known classes of attacks. Next we provide an overview of such attacks and determine the conditions on the parameters which provide resistance to the attacks.

We consider two basic *attack parameters*, namely the number N of keystream bits that is required for a successful attack, and the time complexity T of the attack. To ensure security at level κ any attack should require $T > 2^{\kappa}$. The condition $L \ge 2\kappa$ ensures that the size of the state is at least twice the size of the key. This prevents certain (theoretical) time/memory trade-off attacks [34]. We assume that at most 2^B keystream bits are generated from a single key and IV pair. So N is at most 2^B, as otherwise the attack cannot be mounted. Further, a basic condition is that $N \le T$, i.e. some operation is performed on each keystream bit that is required. So for κ-bit security using 2^B keystream bits, we have $B \leq \kappa$.

Below we perform a concrete analysis of known classes of attacks so as to be able to determine conditions on the parameters which ensure resistance to such attacks. For the concrete analysis, we ignore small constants appearing in the expressions for N and T (i.e. we consider these constants to be 1) and focus only on the exponential components of these quantities.

Linear complexity attack. The degree of $(\mathsf{Maj}, \mathsf{rev})$-$\mathsf{MM}_{2m+1}$ is $d = 2^{\lfloor \log_2 m \rfloor}$. If L is a prime, then the linear complexity of the generated keystream sequence is $\binom{L}{d}$ (see [49]) and so at least these many keystream bits are required to determine the linear complexity. So if

$$\alpha = \binom{L}{2^{\lfloor \log_2 m \rfloor}} > 2^B, \tag{18}$$

then the linear complexity attack is not applicable.

Anderson leakage. An interesting method for exploiting leakage by the filtering function was introduced in [3]. In this approach, the focus is on knowing how much information is leaked by the filtering function *even if the input LFSR sequence is replaced by a purely random bit sequence.* Examples where such leakage can be observed were provided in [3] for the case of filtering functions on a few variables. We call such leakage to be Anderson leakage. Avoiding Anderson leakage amounts to showing that if the input sequence (to the filtering function) is purely random then the keystream sequence is also purely random. Theorem 2 in [27] provides a sufficient condition to ensure this property. The theorem states that if the n-variable filtering function is of the form $Z_1 + g(Z_2, \ldots, Z_n)$, then there is no Anderson leakage. Since the filtering function f (built from MM_{2m+1}) that we propose is of the stated form, it does not exhibit Anderson leakage.

State guessing attacks. These are guess-then-determine attacks. The idea is to guess some bits and then use the obtained keystream to verify the guess. The first attack of this type was described in [27] and is called the inversion attack. The inversion attack is specifically applicable to filtering functions of the form $W \oplus g(\mathbf{Z})$. Since the filtering function that we use is indeed of this form, we need to consider the resistance of the stream cipher to the inversion attack.

Suppose at some point of keystream generation, the state is $\mathbf{s}^{(0)} = (s_{L-1}, \ldots, s_0)$ and the keystream bit generated from $\mathbf{s}^{(0)}$ is w_0. From the definition of the filtering function, the bit w_0 is generated from the bits $(s_{L-1}, \ldots, s_{L-2\kappa})$, and we can write $w_0 = s_{L-2\kappa} \oplus g(s_{L-1}, \ldots, s_{L-2\kappa+1})$, where g is some function (defined from MM_{2m} and is non-degenerate on $2m$ variables). Recall that the tap position for X_1 is $L-1$ and the tap position for Y_m is $L-2\kappa+1$. The gap between the tap position of X_1 and the tap position of Y_m (including both of these tap positions) is $\chi = 2\kappa - 1$. Note that $s_{L-2\kappa} = w_0 \oplus g(s_{L-1}, \ldots, s_{L-2\kappa+1})$. Extending this equation in the backward direction, we obtain the following equations.

$$\begin{array}{rcl} s_{L-2\kappa} & = & w_0 \oplus g(s_{L-1}, \ldots, s_{L-2\kappa+1}) \\ s_{L-2\kappa-1} & = & w_{-1} \oplus g(s_{L-2}, \ldots, s_{L-2\kappa}) \\ s_{L-2\kappa-2} & = & w_{-2} \oplus g(s_{L-3}, \ldots, s_{L-2\kappa-1}) \\ \cdots & \cdot & \cdots \end{array}$$

The sequence $w_0, w_{-1}, w_{-2}, \ldots$ is known. If we guess the values of $s_{L-1}, \ldots, s_{L-2\kappa+1}$ (a total of χ bits), then we obtain $s_{L-2\kappa}$ from the first equation, using the obtained value of $s_{L-2\kappa}$ in the second equation, we obtain the value of $s_{L-2\kappa-1}$, using the obtained value of $s_{L-2\kappa-1}$ in the third equation, we obtain the value of $s_{L-2\kappa-2}$, and so on. In other words, by guessing the values of χ of the state bits, we can obtain values of $L-\chi$ previous state bits, giving us the values of all L bits of the state at a point $L-\chi$ steps in the backward direction. Once the complete state is known, the original guess can be verified by generating the keystream in the forward direction and matching with the obtained keystream. The method requires 2^{χ} guesses, and for each guess about Lm operations are required. By our choice of the tap positions for $\mathbf{X}$ and $\mathbf{Y}$, the value of χ is $2\kappa-1$. So the number of guesses required to successfully mount the attack is far greater than 2^{κ}. This proves the resistance of the stream cipher to the inversion attack. Note that the inversion attack is prevented due to χ being sufficiently large. This was one of the countermeasures to the inversion attack that was already proposed in [27].

The inversion attack was later extended to the generalised inversion attack, the filter state guessing attack, and the generalised filter state guessing attack [27, 28, 33, 47, 58]. We briefly explain the idea behind this line of attacks. Let $n = 2m+1$ be the number of variables of the filtering function. At any point of time, each of the input bits to the filtering function can be written as a linear function of the L state bits of the LFSR obtained after the initialisation phase. So knowing the values of the n input bits to the filtering function at any point of time provides n linear equations in L variables. If the values of the input bits to the filtering function at c points of time are known, where $nc \geq L$, then one obtains a system of L linear equations in L variables and hence can solve this system to obtain the initial state of the LFSR. Of course, one does not know the values of the input bits to the filtering function at any point of time. So a guessing strategy is used.

Let as above the state of the LFSR at some point be $\mathbf{s}^{(0)}$ and the subsequent states of the LFSR be denoted as $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(c-1)}$. For $i = 0, \ldots, c-1$, let $\mathbf{t}^{(i)} = \mathsf{proj}(\mathbf{s}^{(i)})$ and $w_i = f_{2m+1}(\mathbf{t}^{(i)})$. Recall from the definition of δ that for $0 \leq i < j \leq c-1$, $\mathbf{t}^{(i)}$ and $\mathbf{t}^{(j)}$ have an overlap of at most δ bits. So in the c n-bit strings $\mathbf{t}^{(0)}, \ldots, \mathbf{t}^{(c-1)}$ there are at least $nc - \delta c(c-1)/2$ unknown bits and hence there are at least $2^{nc-\delta c(c-1)/2}$ possibilities for these c strings. Since f_{2m+1} is balanced, knowledge of w_0 reduces the number of possibilities for $\mathbf{t}^{(0)}$ by a factor of half, and more generally, the knowledge of $w_0, \ldots, w_{c-1}$ reduces the number of possibilities for $\mathbf{t}^{(0)}, \ldots, \mathbf{t}^{(c-1)}$ by a factor of 2^c. So at least $2^{(n-1)c-\delta c(c-1)/2}$ possibilities remain for the c strings $\mathbf{t}^{(0)}, \ldots, \mathbf{t}^{(c-1)}$. For each of these possibilities, we obtain a system of L equations in L variables, the solution of which provides the value of initial state of the LFSR. The correctness of the obtained value of the state can be determined by generating the keystream from the obtained value and matching with the actual keystream. Solving the system of L linear equations requires L^3 arithmetic operations, and generating the keystream from the obtained state require about Ln operations. So the complexity of the attack is at least $2^{(n-1)c-\delta c(c-1)/2}(L^3+Ln)$ operations. Suppose the following condition holds.

$$2^{(n-1)c-\delta c(c-1)/2}(L^3+Ln) \quad > \quad 2^{\kappa} \text{ where } c \text{ is the least positive integer such that } nc \geq L. \qquad (19)$$

Then no form of the state guessing attack succeeds at the κ-bit security level.

Remark 2 *In our concrete proposals given below in Table 1, the value of c in* (19) *comes out to be 2, and so* (19) *becomes $2^{2(n-1)-\delta}(L^3+Ln) > 2^{\kappa}$. Since the values of $n = 2m+1$ in Table 1 are quite close to κ, ensuring $2^{2(n-1)-\delta}(L^3+Ln) > 2^{\kappa}$ does not require δ to be too small. In fact, the values of δ (and corresponding values of L, m and κ) given in Table 4 ensure that this inequality holds for all our proposals.*

Applying the attack requires knowing the pre-image sets of 0 and 1 of the function f_{2m+1}. These need to be stored separately, and depending upon the value of w_i, the appropriate pre-image set is to be used. So the storage required is 2^{2m+1}. For the concrete proposals that we put forward later, the value of $n = 2m + 1$ is quite close to that of the security level κ (see Table 1). As a result, the memory requirement will be prohitively high to apply the attack.

The complexity of the above attack depends on the value of δ, which is the maximum overlap of tap positions between shifts of the state vector. The recommendations for tap positions in the literature [27, 28, 33, 47, 58] aim to reduce this overlap. A commonly used recommendation is full positive difference set, i.e. the absolute values of the differences between the tap positions should be distinct. For a filtering function with n variables, this recommendation results in the gap between the first and the last tap positions to be more than $1 + 2 + \cdots + (n - 1) = n(n - 1)/2$ (since the successive differences must be distinct). So the size of the LFSR is at least quadratic in n. From efficiency considerations, this forces the value of n to be small. By not following this recommendation, we have done away with the condition that L must be at least quadratic in n. Instead, by considering the fundamental requirement behind state guessing attacks, we identified (19) as the condition to resist such attacks. This allows us to choose n to be quite large and close to κ. Of course, this is possible due to the fact that we have an efficient method for implementing the filtering functions on such large values of n.

Fast correlation attacks. The basic correlation attack [53] is applicable to the combiner generator model. Applying this attack to the filter generator model results in going through all the possible 2^L states of the LFSR. Since by our choice $L \geq 2\kappa$, the basic correlation attack does not defeat the κ-bit security level. (An early correlation attack [52] on the nonlinear filter generator finds an equivalent representation of the stream cipher when the filtering function is not known; since we assume that the filtering function is known, this attack is not relevant to our context.)

Fast correlation attacks do not require exhaustive search on the states of the LFSR. There is a large literature on fast correlation attacks including older papers such as [45, 35, 36, 19, 13, 37, 12, 20] as well as more recent papers such as [54, 60, 59, 39, 40]. See [10, 11, 43, 2] for surveys of the area. In the following, we evaluate security against some representative fast correlation attacks and show how to choose the values of the design parameters so as to resist these attack. Recall from Proposition 2 that for MM_{2m+1}, and hence for f_{2m+1}, the linear bias $\varepsilon = 2^{-m-1}$.

Type of attack. For attacks based on low-weight parity-check equations [45, 13], the number of keystream bits required is about $N = (2\varepsilon)^{-2(d-2)/(d-1)} \cdot 2^{L/(d-1)}$, the pre-computation step requires about $N^{d-2}/(d-2)!$ operations, and the (online) time for decoding is about $(2\varepsilon)^{-2d(d-2)/(d-1)} \cdot 2^{L/(d-1)}$, where $d \geq 3$ is the number of non-zero terms in (some multiple of the) LFSR connection polynomial, and ε is the linear bias.
Evaluation against $\mathcal{S}(L, m)$. We have $\varepsilon = 2^{-m-1}$ and so $N = 2^{(2(m+2)(d-2)+L)/(d-1)}$. We set $N = 2^B$ and so $(d - 1)(B - 2(m + 2)) = L - 2(m + 2)$. If $B = 2(m + 2)$, then $L = 2(m + 2)$. We choose L and m to ensure that $L > 2(m + 2)$, and so $B \neq 2(m + 2)$. In this case, we solve for d to obtain $d = 1 + (L - 2(m + 2))/(B - 2(m + 2))$. If $B < 2(m + 2)$, then $d < 1$ which violates the condition $d \geq 3$ and the attack does not work. So let us consider $B > 2(m + 2)$. In this case, substituting the obtained expression for d in the expression for the decoding time, we find the decoding time to be $2^{(2(m+2)L+B^2)/(B-2(m+2))}$. So the following condition ensures κ-bit security.

$$L > (2m + 2) \text{ and either } B < 2(m + 2) \text{ or } \kappa < \frac{2(m + 2)L + B^2}{B - 2(m + 2)}. \tag{20}$$

Note that (20) ensures κ-bit security *for all values of* $d \geq 3$. In particular, it does not matter whether the feedback polynomial of the LFSR is sparse, or whether it has a sparse multiple.

Type of attack. For attacks based on general decoding, [10] identifies the key idea to be from [19]. The attack in [19] requires N to be about $\varepsilon^{-2} \cdot 2^{(L-k)/w}$ under the condition $N \gg (2\varepsilon)^{-2w}$, and the time complexity of the decoding step is about $2^k \cdot (2\varepsilon)^{-2w}$, where ε is the linear bias of the filtering function, and $k \in \{1, \ldots, L\}$ and $w \geq 2$ are algorithm parameters.
Evaluation against $\mathcal{S}(L, m)$. Again we have $\varepsilon = 2^{-m-1}$. Setting $N = 2^B$, we obtain $k = L + 2w(m+1) - wB$. Setting T to be the time complexity of the decoding step, we obtain $\log_2 T = L + 2w(2m+3) - wB$. Since $L \geq 2\kappa$, ensuring $2(2m+3) \geq B$ is (more than) sufficient to ensure κ-bit security, *irrespective of the values of* w *and* k. We record this condition as follows.

$$B \quad \leq \quad 2(2m+3). \tag{21}$$

Type of attack. The attacks in [37, 12] apply specifically to the filter generator model. These two attacks are essentially the same when the filtering function is plateaued (which is the case for f_{2m+1}). For the attack, N is about $2^{(L-k)/w}$, and T is about $2^k \cdot F^w$, where w and k are as in the attack in [19] (see above) and F is the size of the support of the Walsh transform of the filtering function.
Evaluation against $\mathcal{S}(L, m)$. The size of the support of the Walsh transform of MM_{2m+1} is equal to the size of the support of the Walsh transform of MM_{2m}. Since MM_{2m} is bent, it follows that the size of the support of MM_{2m} is 2^{2m}. So $F = 2^{2m}$. Setting $N = 2^B$, we obtain $k = L - wB$. Substituting k in the expression for T, we obtain $\log_2 T = L + w(2m - B)$. Since $L \geq 2\kappa$ and $w \geq 2$, we have $\log_2 T = L + w(2m - B) \geq 2\kappa + 2(2m - B)$. So if $2\kappa + 2(2m - B) > \kappa$, or equivalently, $\kappa + 4m > 2B$, then κ-bit security is achieved against the attack. We record this condition as follows.

$$B \quad < \quad (\kappa + 4m)/2. \tag{22}$$

Other fast correlation attacks. We next consider some of the more recent attacks. The attack in [54] is based on using $M > 1$ linear approximations. For this attack, both N and T are about $2^{L-\beta}$, where β is an algorithm parameter. A necessary condition for the attack to succeed is that $M > 2^\beta$. For the attack to succeed at the κ-bit security level, i.e. $T \leq 2^\kappa$, it is required to have $\beta \geq L - \kappa$. From the bound on M, it follows that more than $2^{L-\kappa}$ linear approximations with sufficiently high correlations are required. For Grain-128a [1], L equals 128 and about $2^{26.58}$ (i.e. M is about $2^{26.58}$) linear approximations with absolute correlations at least $2^{-54.2381}$ were identified in [54]. For $\kappa = 128$, Table 1 recommends $L = 257$ and $m = 58$. To apply the attack in [54] to $\mathcal{S}(257, 59)$, more than 2^{129} linear approximations with sufficiently high correlations are required. The linear bias of the filtering function itself is 2^{-60}. Finding multiple linear approximations requires combining keystream bits which further lowers the linear bias. So there is no approximation with linear bias greater than 2^{-60}. Further, going through the details of the attack in [1], we could not identify any method to obtain more than 2^{129} linear approximations. So there does not seem to be any way to apply the attack in [54] to $\mathcal{S}(257, 59)$.

Subsequent works (such as [60, 59, 39, 40]) on fast correlation attacks use vectorial decoding technique along with multiple linear approximations, use of the BKW algorithm [7], and multivariate correlation attack. The attacks are quite complex and there are no simple closed form expressions for the values of N and T. The stream ciphers to which these attacks are applied are Grain-128a and Sosemanuk [5]. From the above discussion, we already know that the linear bias of $\mathcal{S}(257, 59)$ is substantially lower than that of Grain-128a. For Sosemanuk, the best known [39] linear approximation has correlation $2^{-20.84}$, which is far greater than the linear bias 2^{-60} for $\mathcal{S}(257, 59)$. The very low linear bias of $\mathcal{S}(257, 59)$ and in general of the other stream cipher proposals in Table 1 make the attacks in [60, 59, 39, 40] inapplicable at the stated security levels.

(Fast) Algebraic attacks. For a filtering function f having $\mathsf{AI}(f) = a$, an algebraic attack requires about $N = \sum_{i=0}^{a} \binom{L}{i}$ keystream bits and has time complexity T to be about N^{ω}, where ω is the exponent of matrix multiplication (see Section 3.1.5 of [14]). We assume $\omega = 2.8$. For our proposal, the filtering function is f_{2m+1} whose algebraic resistance is the same as that of $(\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_{2m+1}$). Let $a = (f_{2m+1}) = \mathsf{AI}((\mathsf{Maj}, \mathsf{rev})\text{-}\mathsf{MM}_{2m+1})$. From Proposition 4, we assume that $a = \lceil m/2 \rceil$, i.e. the actual algebraic immunity of f_{2m+1} is equal to the lower bound on the algebraic immunity. Let

$$\beta = \left(\sum_{i=0}^{\lceil m/2 \rceil} \binom{L}{i} \right)^{2.8}. \tag{23}$$

So choosing L and m such that $T = \beta > 2^{\kappa}$ prevents algebraic attacks at the κ-bit security level.

For any (e, d) for which there are functions g and h of degrees e and d respectively such that $gf_{2m+1} = h$, the Berlekamp-Massey step in a fast algebraic attack on the filter generator model has time complexity $\mathcal{O}(ED \log D)$, where $E = \sum_{i=0}^{e} \binom{L}{i}$ and $D = \sum_{i=0}^{d} \binom{L}{i}$ (see [31] and Section 3.1.5 of [14]). This complexity dominates the overall time complexity of a fast algebraic attack. Ignoring the logarithm term, we take T to be equal to ED. The number N of keystream bits required is about $2E$. The maximum value of e is one less than the algebraic immunity a. From Proposition 4, we again assume as above that $a = \lceil m/2 \rceil$. Recall that for any Boolean function, its fast algebraic immunity is at least one more than its algebraic immunity, i.e. $\mathsf{FAI}(f_{2m+1}) \geq a + 1$. For any functions g and h of degrees e and d respectively such that $gf_{2m+1} = h$, we have $e + d \geq \mathsf{FAI}(f_{2m+1}) \geq a + 1$, and we assume that the lower bound is the actual value of $e + d$, and so $d = a + 1 - e$. Define

$$\gamma = \min_{\substack{1 \leq e \leq a-1, \\ d = a+1-e}} \left(\sum_{i=0}^{e} \binom{L}{i} \right) \left(\sum_{i=0}^{d} \binom{L}{i} \right). \tag{24}$$

So choosing L and m such that $T = \gamma > 2^{\kappa}$ prevents fast algebraic attacks at the κ-bit security level.

4.3 Concrete Choices of L, m and pos

Given κ, we obtained representative values of L and m. The procedure we followed to obtain L and m is to choose the value of L to be the first prime number greater than 2κ, and then for the chosen value of L, choose m to be the least integer such that $\beta, \gamma > 2^{\kappa}$. This ensures security against (fast) algebraic attacks considered above. Next, using κ and the corresponding values of L and m, we computed the maximum value of B such that $B \leq \kappa$, $\alpha > 2^{B}$, and (20), (21) and (22) hold. In each of the cases that we considered, it turns out that this maximum value of B is in fact κ. So for the chosen values of L and m, the corresponding $\mathcal{S}(L, m)$ ensures κ-bit security against the above analysed correlation attacks even when the adversary has access to 2^{κ} keystream bits. We note, however, that generating 2^{κ} keystream bits from a single key and IV pair is meaningless from a practical point of view. Instead, we set $B = 64$, i.e. *from a single key and IV pair at most* 2^{64} *keystream bits are to be generated.* Table 1 shows the values of κ, L, m and other parameters of the filtering function. We note the following points regarding the entries in Table 1.

1. The number of variables of the filtering function f_{2m+1} is $2m + 1$ (and not m).
2. For each κ, the value of L is the first prime number greater than 2κ. Our choice of a prime number for the value of L is to ensure that the linear complexity of the generated keystream is indeed equal to α.
3. The table provides representative values of L and m. It is possible to have other pairs of values for (L, m) which provide κ-bit security. Different values of L and m lead to different sizes of the

κ	L	m	$2m+1$	deg	LB	AI	FAI
80	163	37	75	32	2^{-38}	19	20
128	257	59	119	32	2^{-60}	30	31
160	331	71	143	64	2^{-72}	36	37
192	389	87	175	64	2^{-88}	44	45
224	449	101	203	64	2^{-102}	51	52
256	521	115	231	64	2^{-116}	58	59

Table 1: Values of L and m which provide κ-bit security against the attacks analysed in this section when the attacker has access to at most 2^B keystream bits.

L	prim poly
163	$x^{163} \oplus x^7 \oplus x^6 \oplus x^3 \oplus 1$
257	$x^{257} \oplus x^7 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$
331	$x^{331} \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$
389	$x^{389} \oplus x^7 \oplus x^6 \oplus x^3 \oplus x^2 \oplus x \oplus 1$
449	$x^{449} \oplus x^9 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus x \oplus 1$
521	$x^{521} \oplus x^9 \oplus x^6 \oplus x^5 \oplus x^3 \oplus x \oplus 1$

Table 2: Examples of primitive polynomials for the values of L shown in Table 1.

circuits implementing the corresponding stream ciphers. For a fixed value of κ, a practical designer may need to consider various values of (L, m) to determine which pair provides the smallest circuit. In Section 5, we provide gate count estimates for the stream ciphers corresponding to the values of L and m in the table.

4. For each entry in the table, our calculation shows that $\beta \gg \gamma$. For example, for $\mathcal{S}(257, 59)$, we obtained $\beta = 2^{364.38}$ while $\gamma = 2^{130.12}$. This is not surprising, since the fast algebraic attack is known to be much more faster than the basic algebraic attack.
5. The feedback polynomial for the LFSR has to be a primitive polynomial of degree L. The above analysis of correlation attacks shows that for the obtained values of L and m, it does not matter whether the polynomial is sparse, or whether it has a sparse multiple. So from an efficiency point of view, one may choose a low weight primitive polynomial. Examples of primitive polynomials for the values of L in Table 1 are shown in Table 2.

Concrete choices of the tap positions. Given the values of L and m (and also κ), we applied the procedure for choosing the tap positions encoded by pos as outlined in Section 4.1. The leftmost κ bits of pos is the string posX. The next $\kappa - 1$ bits of pos (i.e. the positions pos$[L-\kappa-1, \ldots, L-2\kappa+1]$) encode the tap positions for the variables in $\mathbf{Y}$. We define posY to be the κ-bit string formed by appending a 0 to the $(\kappa-1)$-bit segment pos$[L-\kappa-1, \ldots, L-2\kappa+1]$ of pos. So both posX and posY are κ-bit strings and are uniquely defined from pos. Further, given posX and posY it is possible to uniquely construct pos (by concatenating posX and posY, appending $L-2m$ zeros, and setting the bit in position $L-2\kappa$ to be 1, to encode the tap position for W). So providing posX and posY is equivalent to providing pos. In Table 3 we provide the values of posX and posY encoded as hexadecimal strings. For example, the entry for posX corresponding to $\mathcal{S}(128, 59)$ is be352$\cdots$ which encodes the binary string 10111110001101010010$\cdots$. In Table 4 we provide the corresponding values of ν and δ. Recall that ν is obtained from posX as in (16) and δ is obtained from pos as in (17).

The number of variables of the filtering function is $n = 2m+1$. For each of the obtained values of L and n, and the corresponding value of δ we checked whether (19) holds. In each case we found that (19) indeed holds. So the stream cipher proposals ensure security against state guessing attacks.

κ	posX and posY
$\mathcal{S}(80, 37)$	d569a664f500763506c3 ff0149d4c640e9846cf2
$\mathcal{S}(128, 59)$	be352d9ca349432b80b38ac54e5164c9 d2ece08cbb5566d608a69b19e4a91418
$\mathcal{S}(160, 71)$	ea4308e1229305d185450cfa26b0dcac68c4ab7d 1dbb5a438e7e55904cc04406bf0670ad728462b0
$\mathcal{S}(192, 87)$	a0265ea181b73a460fb50d8482590e584d15869de343957e c6b218be600d6183c074d00fde24e1c308ebb06cebab0f84
$\mathcal{S}(224, 101)$	e9507d49d4f4609a710d8d291eb466430af5668b03ec424c18417d86 d288451f8f0554a46615f4448afa34aab8673d0647044afcd4682ec4
$\mathcal{S}(256, 115)$	c1ec835120741f290154b122618c625f0a9e77c5172cac84ae564390b2e91fda 5865fda7830eca37d0c2045994e9c83b1c55e13f1966c220809bc019d37f0054

Table 3: The strings posX and posY corresponding to the values of L and m for κ which are shown in Table 1. The first row corresponds to posX and the second row corresponds to posY.

$\mathcal{S}(L, m)$	ν	δ
$\mathcal{S}(80, 37)$	16	36
$\mathcal{S}(128, 59)$	26	57
$\mathcal{S}(160, 71)$	30	69
$\mathcal{S}(192, 87)$	39	86
$\mathcal{S}(224, 101)$	45	96
$\mathcal{S}(256, 115)$	51	112

Table 4: Values of ν and δ corresponding to the strings posX and pos (built from posX and posY) shown in Table 3.

4.4 Differential Attacks

Choosing the filtering function to protect against certain known classes of attacks does not however, protect against all possible attacks. There are attacks which can succeed even if the filtering function is properly chosen. One such attack on a previous version of the proposal was described in [6]. To describe the attack and the design modification to resist it we need to mention the aspects of the design which were different in the previous proposal. There are three such aspects.

1. In the definition of MM_{2m}, in the earlier version we had chosen π to be the identity permutation, whereas in the present version we choose π to be the bit reversal permutation.

2. In the previous version, the tap positions were chosen as follows. From the state $(s_{L-1}, \ldots, s_0)$ of the LFSR, the value of W in $f_{2m+1}(W, \mathbf{X}, \mathbf{Y})$ was chosen to be the state bit $s_{L-\kappa+m}$, the values of the variables in $\mathbf{X}$ were chosen to be the state bits $s_{L-\kappa+m-1}, \ldots, s_{L-\kappa}$, and the values of the variables in $\mathbf{Y}$ were chosen to be the state bits $s_{L-2\kappa+m-1}, \ldots, s_{L-2\kappa}$.

3. In the previous version, during the initialisation phase, the feedback from the filtering function was fed back into the state by XORing it with the next bit of the LFSR, i.e. it was fed back to only one position of the state.

Suppose the state after the initialisation phase is $(s_{L-1}, \ldots, s_0)$, and for $t \geq 1$, let $s_t = \mathsf{nb}(s_{t-1}, \ldots, s_{t-L})$. Let the keystream bit generated from the state $(s_t, s_{t-1}, \ldots, s_{t-L+1})$ be w_t. Using the definition of π to be the identity permutation and the tap positions as mentioned above, we have

$$\begin{aligned} w_t &= s_{t-\kappa+m} \oplus \langle \pi(s_{t-\kappa+m-1}, \ldots, s_{t-\kappa+1}, s_{t-\kappa}), (s_{t-2\kappa+m-1}, \ldots, s_{t-2\kappa+1}, s_{t-2\kappa}) \rangle \\ &\quad \oplus \mathsf{Maj}_m(s_{t-\kappa+m-1}, \ldots, s_{t-\kappa+1}, s_{t-\kappa}), \end{aligned}$$

So

$$\begin{aligned} w_t \oplus w_{t+1} &= s_{t-\kappa+m-1}s_{t-2\kappa+m-1} \oplus s_{t+1-\kappa}s_{t+1-2\kappa} \\ &\quad \oplus \mathsf{Maj}_m(s_{t-\kappa+m-1}, \ldots, s_{t-\kappa+1}, s_{t-\kappa}) \oplus \mathsf{Maj}_m(s_{t-\kappa+m}, \ldots, s_{t-\kappa+2}, s_{t-\kappa+1}). \end{aligned}$$

In other words, due to the choice of π as the identity permutation, the selection of tap positions as consecutive positions of the state, and the fact that two successive keystream bits are obtained from a single shift of the LFSR sequence, $m-1$ of the quadratic terms in the inner product cancelled out when w_t and w_{t+1} are XORed together. Further, the inputs to the two calls to Maj have an overlap of size $m-1$. These features were observed in [6] and exploited to mount a differential attack. The idea of the differential attack is to introduce a difference in the top most bit position of the IV. During the initialisation phase, this difference travels unchanged for $\kappa - m$ steps and then produces a difference in the input to the filtering function. The difference in the output of the filtering function is fed back into the leftmost bit of the LFSR. As the LFSR is further shifted, this difference then travels without any further modification creating a high probability truncated differential. Combined with the simplified form of $w_t \oplus w_{t+1}$, this leads to an efficient key recovery attack.

In the present version, we have chosen π to be the bit reversal permutation, and all the tap positions for $\mathbf{X}$ to occur to the left of all the tap positions for $\mathbf{Y}$. As a result, in the XOR of any number of keystream bits, no quadratic term cancels out (see Proposition 5). So, in particular, the above kind of cancellation of quadratic terms does not arise.

Further, due to our design procedure for the tap positions of corresponding to the variables $X_1, \ldots, X_n$, the overlap of inputs in the calls to Maj in the XOR of any two keystream bits is at most ν. From Table 4, we observe that the value of ν is considerably smaller than m. So the complexity of Maj is mostly preserved in the XOR of any number of keystream bits.

Further, in the present version, during the initialisation phase we inject the output of the filtering function at multiple positions of the state, the number of positions being controlled by the parameter μ. So any (controlled) difference in any bit position travels at most μ steps before it is further modified. Since our recommendation is to choose μ to be about $\sqrt{2\kappa}$, in the full initialisation phase consisting of 2κ iterations, a difference will be updated about $\sqrt{2\kappa}$ times. This makes it difficult to control a difference through all the initialisation rounds.

Due to the above modifications, the attack in [6] on the previous proposal does not apply to the modified proposal. In fact, the modifications though inexpensive, substantially improve the differential properties of the keystream and also improve the "robustness" of the initialisation phase. We note, however, unlike our analysis of the some of the other attacks, we do not have any proof that our proposal resists all kinds of differential attacks. We welcome further analysis of our proposals, including finding other avenues of attack.

5 Efficiency of Computing MM_n

In Section 4, we proposed using f_{2m+1} (which is $1 \oplus \mathsf{MM}_{2m+1}$) as the filtering function in the nonlinear filter model. Further, Table 1 provides specific suggestions of values of m for achieving different security levels. In this section, we consider the complexity of implementing MM_{2m+1}. Since MM_{2m+1} is constructed from MM_{2m} using one XOR gate, we need to consider the complexity of implementing MM_{2m}.

The computation of MM_{2m} requires computing h and an inner product of two m-bit strings. The computation of h requires the computation of the weight of an m-bit string and the computation of a threshold function. We discuss basic strategies for implementing these operations which provide estimates of the number of gates required.

Inner product of two m-bit strings. This operation requires m AND gates and $m-1$ XOR gates.

Weight of an m-bit string. A half-adder takes two input bits and outputs two bits which represent sum of the two input bits. A full adder takes three input bits and outputs two bits which represent the sum of the three input bits. We estimate the numbers of half and full adders that are required for computing the weight of an m-bit string $\mathbf{x}$. The algorithm for computing weight that we use is from [8]. If $m = 1$, then no adders are required, if $m = 2$, a half adder computes the weight, and if $m = 3$, a full adder computes the weight. For $m > 3$, write $m = m_1 + m_2 + 1$, where $m_1 + 1$ is the highest power of two that is at most m. The algorithm computes the weight of the first m_1 bits of $\mathbf{x}$, the weight of the next m_2 bits of $\mathbf{x}$, and then adds these two weights together with the last bit of $\mathbf{x}$.

Using the above algorithm, it is easy to show that the computation of the weight of an m-bit string, when $m = 2^r - 1$ with $r \geq 1$, requires $2^r - r - 1$ full adders. From this it easily follows that the number of full adders required to compute the weight of an m-bit string for arbitrary m is $O(m)$. Since a full adder can be implemented using a constant number of NAND gates, the number of NAND gates required to compute the weight of an m-bit string is also $O(m)$. We record this fact in the following result.

Proposition 7 *The computation of* Maj_m *can be done using* $O(m)$ *NAND gates.*

We are interested in the exact counts of full and half adders required for the values of m in Table 1. Let us denote a full adder by [F] and a half adder by [H]. In Table 1, one of the choices is $m = 37$. Writing $37 = 31 + 5 + 1$, it is required to find the weight of one 31-bit string, one 5-bit string and then perform the final addition. Computation of the weight of the 31-bit string requires 26[F]. Writing $5 = 3 + 1 + 1$, the computation of the weight of a 5-bit string requires 3[F] (a full adder to compute the weight of a 3-bit string, and 1[F]+1[H] to add the remaining two bits to this weight). The weight of a 5-bit string is a 3-bit quantity, while the weight of a 31-bit string is a 5-bit quantity. So the final addition of the weights along with the remaining bit requires 3[F]+2[H]. So a total of 31[F]+3[H] is required to compute the weight of a 37-bit string. In a similar manner, it is possible to obtain the numbers of full and half adders required to compute the weights of m-bit strings for the values of m given in Table 1 and these counts are given below.

$m = 37$: 31[F]+3[H]; $m = 59$: 53[F]+1[H]; $m = 71$: 64[F]+3[H];
$m = 87$: 80[F]+2[H]; $m = 101$: 94[F]+3[H]; $m = 115$: 108[F]+2[H].

Threshold function on the weight of an m-bit string. The weight of a 37-bit string is a 6-bit quantity, say $w_5w_4w_3w_2w_1w_0$. It is required to determine whether the value represented by this string is at least 19. This is computed by the Boolean formula $w_5 \vee (w_4 \wedge (w_3 \vee w_2 \vee (w_1 \wedge w_0)))$, requiring 3[OR]+2[AND] gates. In general, the weight of an m-bit string is an ω-bit value, where $\omega = \lceil \log_2 m \rceil$. To compute the threshold function, ω_1 OR and ω_2 AND gates are required for some values of ω_1 and ω_2 satisfying $\omega_1 + \omega_2 \leq \omega$. So the number of gates for computing the threshold function on the weight of an m-bit string requires a logarithmic (in m) number of gates.

	$\mathcal{S}(163, 37)$	$\mathcal{S}(257, 59)$	$\mathcal{S}(331, 70)$	$\mathcal{S}(389, 87)$	$\mathcal{S}(449, 100)$	$\mathcal{S}(521, 114)$
LFSR	1304	2056	2648	3112	3592	4168
f_{2m+1}	439.5	715.5	872.5	1075.5	1262.5	1439.5
total	1743.5	2771.5	3520.5	4187.5	4854.5	5607.5

Table 5: Estimates of the number of NAND gates required to implement $\mathcal{S}(L, m)$ for values of L and m in Table 1.

Circuit size for computing MM_n. The inner product requires $O(m)$ OR and AND gates and the weight computation requires $O(m)$ full adders. So the circuit size for computing MM_n is $O(n)$.

Next we consider concrete estimates. For such estimates, we ignore the at most $\lceil \log_2 m \rceil$ AND and OR gates required for computing the threshold function from the weight, and the gates required to compute the next bit of the LFSR. Further, we also ignore the $\lceil 2\kappa/\mu \rceil$ XOR gates required for the feedback injection from the filtering function during the initialisation phase. This number is not much. For example, for $\kappa = 128$, we have $\mu = \lfloor \sqrt{2\kappa} \rfloor = 16$ and so 16 XOR gates are required to feedback the output of the filtering function during the initialisation phase. We obtain concrete estimates for the two major components, the LFSR and the filtering function.

To obtain concrete estimates, it is convenient to convert the various gate counts into a single unit. Previous works [1, 9] have taken a single NAND gate as the basic unit and translated other gates in terms of this unit. A half-adder can be implemented using 5 NAND gates, while a full adder can be implemented using 9 NAND gates. In [1, 9], a XOR gate was taken to be 2.5 units and an AND gate was taken to be 1.5 units. Between the papers [1] and [9] there is a difference in the number of units required for a flip-flop: [1] takes a flip-flop to be 8 units, while [9] takes a flip-flop to be 12 units. In Table 5, we provide estimates of the circuit sizes of $\mathcal{S}(L, m)$ for the values of (L, m) in Table 1. These estimates consider a flip-flop to be 8 units.

If we consider a flip-flop to be 12 units as done for Trivium [9], then the gate count estimate for $\mathcal{S}(163, 37)$ is 2395.5. The gate estimate for Trivium obtained in [9] is 3488. Both Trivium and $\mathcal{S}(163, 37)$ are targeted at the 80-bit security level, and $\mathcal{S}(163, 37)$ is substantially smaller than Trivium. The lower size of $\mathcal{S}(163, 37)$ is due to the smaller state size; $\mathcal{S}(163, 37)$ uses a 163-bit state, while Trivium uses a 288-bit state.

The gate count estimate for Grain-128a obtained in [1] is 2145.5. Grain-128a is targeted at the 128-bit security level. Comparing to $\mathcal{S}(257, 59)$, which is also targeted at the 128-bit security level, we find the gate count estimate of $\mathcal{S}(257, 59)$ to be 2771.5. The state sizes of both Grain-128a and $\mathcal{S}(257, 59)$ are almost the same. The greater size of $\mathcal{S}(257, 59)$ is due to the greater gate size requirement of the filtering function f_{2m+1} in comparison to the gate size requirement of the nonlinear components of Grain-128a. Even though $\mathcal{S}(257, 59)$ is larger than Grain-128a, its size of about 2771.5 gates is small enough for $\mathcal{S}(257, 59)$ to be considered as a small cipher. Finally we note that $\mathcal{S}(521, 114)$ which is targeted at the 256-bit security level requires about 5607.5 gates. We know of no other 256-bit secure stream cipher which has such a small gate count.

We note that it is possible to ramp up security at the cost of a reasonable increase in gate count. For example, from Table 1, at the 128-bit security level, our proposal has $L = 257$ and $m = 59$, resulting in linear bias equal to 2^{-60} and the fast algebraic attack requiring more than $2^{130.12}$ operations. If we increase m from 59 to 63, then the linear bias drops to 2^{-64}, and the fast algebraic attack now requires more than $2^{135.83}$ operations. The gate estimate for $\mathcal{S}(257, 63)$, i.e. the stream cipher with $L = 257$ and $m = 63$, is 2818.5 gates (the LFSR requires 2056 gates, and f_{127} requires 762.5 gates). So the security increases by about 5 bits at the cost of an increase of only 47 gates in the circuit size.

6 Conclusion

We described a construction of balanced Boolean functions which has several provable properties, namely very high nonlinearity, acceptable algebraic resistance, *and* is efficient to implement in hardware. Using such Boolean functions, we proposed concrete constructions of the nonlinear filter model for stream ciphers targeted at different security levels. Gate count estimates for the stream cipher proposals show that the circuit sizes compare well with famous stream ciphers at the 80-bit and the 128-bit security levels, while for higher security levels, we do not know of any stream cipher with lower gate count estimates.

Acknowledgement

We thank Subhadeep Banik, Tim Beyne, Pierrick Méaux, Will Meier, Michiel Verbauwhede, and Bin Zhang for their comments on earlier versions of the paper. Deng Tang provided us with a program written by Simon Fischer which we have used for computing fast algebraic immunity. We thank both of them.

References

[1] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *Int. J. Wirel. Mob. Comput.*, 5(1):48–59, 2011. 5, 6, 19, 25

[2] Martin Ågren, Carl Löndahl, Martin Hell, and Thomas Johansson. A survey on fast correlation attacks. *Cryptogr. Commun.*, 4(3-4):173–202, 2012. 2, 18

[3] Ross J. Anderson. Searching for the optimum correlation attack. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 137–143. Springer, 1994. 2, 4, 16

[4] Steve Babbage and Matthew Dodd. The MICKEY stream ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 191–209. Springer, 2008. 3, 5, 6

[5] Côme Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin, and Hervé Sibert. Sosemanuk, a fast software-oriented stream cipher. In M. Robshaw and O. Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 98–118. Springer, 2008. 2, 5, 6, 19

[6] Tim Beyne and Michiel Verbauwhede. Cryptanalysis of a nonlinear filter-based stream cipher. Cryptology ePrint Archive, Paper 2025/197, 2025. 4, 6, 22, 23

[7] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. 19

[8] Joan Boyar and René Peralta. The exact multiplicative complexity of the Hamming weight function. *Electron. Colloquium Comput. Complex.*, TR05-049, 2005. `https://dblp.org/rec/journals/eccc/ECCC-TR05-049.bib`. 24

[9] Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008. 3, 5, 6, 25

[10] Anne Canteaut. Fast correlation attacks against stream ciphers and related open problems. In *Proceedings of the IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security*, pages 49–54. IEEE, 2005. 2, 18, 19

[11] Anne Canteaut. Fast correlation attack. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed*, pages 450–452. Springer, 2011. 2, 18

[12] Anne Canteaut and Eric Filiol. On the influence of the filtering function on the performance of fast correlation attacks on filter generators. In *Proceedings of the 23rd Symposium on Information Theory in Benelux*, Lecture Notes in Computer Science, May 2000. 2, 18, 19

[13] Anne Canteaut and Michaël Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588. Springer, 2000. 2, 18

[14] Claude Carlet. *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press, 2021. 3, 6, 7, 20

[15] Claude Carlet. A wide class of Boolean functions generalizing the hidden weight bit function. *IEEE Trans. Inf. Theory*, 68(2):1355–1368, 2022. 2

[16] Claude Carlet, Deepak Kumar Dalai, Kishan Chand Gupta, and Subhamoy Maitra. Algebraic immunity for cryptographically significant Boolean functions: Analysis and construction. *IEEE Trans. Inf. Theory*, 52(7):3105–3121, 2006. 9, 10

[17] Claude Carlet and Keqin Feng. An infinite class of balanced functions with optimal algebraic immunity, good immunity to fast algebraic attacks and good nonlinearity. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*, pages 425–440. Springer, 2008. 2

[18] Anupam Chattopadhyay, Subhamoy Maitra, Bimal Mandal, Manmatha Roy, and Deng Tang. Efficient hardware implementation for Maiorana-McFarland type functions. *IACR Cryptol. ePrint Arch.*, page 1970, 2023. 5

[19] Vladimir V. Chepyzhov, Thomas Johansson, and Ben J. M. Smeets. A simple algorithm for fast correlation attacks on stream ciphers. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2000. 2, 18, 19

[20] Philippe Chose, Antoine Joux, and Michel Mitton. Fast correlation attacks: An algorithmic point of view. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 209–221. Springer, 2002. 2, 18

[21] Andrew J. Clark, Jovan Dj. Golic, and Ed Dawson. A comparison of fast correlation attacks. In Dieter Gollmann, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 1996. 2

[22] Nicolas T. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer, 2003. 2, 7

[23] Nicolas T. Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003. 2, 7

[24] Deepak Kumar Dalai, Subhamoy Maitra, and Sumanta Sarkar. Basic theory in construction of Boolean functions with maximum possible annihilator immunity. *Des. Codes Cryptogr.*, 40(1):41–58, 2006. 3, 7

[25] Hans Dobbertin. Construction of bent functions and balanced Boolean functions with high nonlinearity. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 1994. 3

[26] Patrik Ekdahl, Thomas Johansson, Alexander Maximov, and Jing Yang. A new SNOW stream cipher called SNOW-V. *IACR Trans. Symmetric Cryptol.*, 2019(3):1–42, 2019. 2, 5, 6

[27] Jovan Dj. Golic. On the security of nonlinear filter generators. In Dieter Gollmann, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 173–188. Springer, 1996. 2, 4, 16, 17, 18

[28] Jovan Dj. Golic, Andrew J. Clark, and Ed Dawson. Generalized inversion attack on nonlinear filter generators. *IEEE Trans. Computers*, 49(10):1100–1109, 2000. 2, 4, 17, 18

[29] Jovan Dj. Golic and Slobodan V. Petrovic. A generalized correlation attack with a probabilistic constrained edit distance. In Rainer A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 472–476. Springer, 1992. 2

[30] Kishan Chand Gupta and Palash Sarkar. Efficient representation and software implementation of resilient Maiorana-McFarland s-boxes. In Chae Hoon Lim and Moti Yung, editors, *Information*

Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers, volume 3325 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 2004. 5

[31] Philip Hawkes and Gregory G. Rose. Rewriting variables: The complexity of fast algebraic attacks on stream ciphers. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2004. 20

[32] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. In *Proceedings 2006 IEEE International Symposium on Information Theory, ISIT 2006, The Westin Seattle, Seattle, Washington, USA, July 9-14, 2006*, pages 1614–1618. IEEE, 2006. 2

[33] Samir Hodžić, Enes Pasalic, and Yongzhuang Wei. Guess and determine cryptanalysis with variable sampling and its applications. *IET Inf. Secur.*, 13(6):559–569, 2019. 2, 4, 17, 18

[34] Jin Hong and Palash Sarkar. New applications of time memory data tradeoffs. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 353–372. Springer, 2005. 15

[35] Thomas Johansson and Fredrik Jönsson. Fast correlation attacks based on turbo code techniques. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 1999. 2, 18

[36] Thomas Johansson and Fredrik Jönsson. Improved fast correlation attacks on stream ciphers via convolutional codes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 347–362. Springer, 1999. 2, 18

[37] Fredrik Jönsson and Thomas Johansson. A fast correlation attack on LILI-128. *Inf. Process. Lett.*, 81(3):127–132, 2002. 2, 18, 19

[38] Sabine Leveiller, Gilles Zémor, Philippe Guillot, and Joseph Boutros. A new cryptanalytic attack for PN-generators filtered by a Boolean function. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 2002. 2

[39] Sudong Ma, Chenhui Jin, Jie Guan, Ting Cui, and Zhen Shi. Improved fast correlation attack using multiple linear approximations and its application on Sosemanuk. *IEEE Trans. Inf. Theory*, 70(10):7484–7497, 2024. 2, 5, 18, 19

[40] Isaac Andrés Canales Martinez and Igor Semaev. Multivariate correlation attacks and the cryptanalysis of lfsr-based stream ciphers. *Des. Codes Cryptogr.*, 92(11):3391–3427, 2024. 2, 18, 19

[41] Pierrick Méaux. From at least $n/3$ to at most $3\sqrt{n}$: Correcting the algebraic immunity of the hidden weight bit function. Cryptology ePrint Archive, Paper 2025/615, 2025. 10

[42] Pierrick Méaux and Yassine Ozaim. On the cryptographic properties of weightwise affine and weightwise quadratic functions. *Discrete Applied Mathematics*, 355:13–29, 2024. 2

[43] Willi Meier. Fast correlation attacks: Methods and countermeasures. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *Lecture Notes in Computer Science*, pages 55–67. Springer, 2011. 2, 18

[44] Willi Meier, Enes Pasalic, and Claude Carlet. Algebraic attacks and decomposition of Boolean functions. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer, 2004. 7

[45] Willi Meier and Othmar Staffelbach. Fast correlation attacks on certain stream ciphers. *J. Cryptol.*, 1(3):159–176, 1989. 2, 18

[46] Enes Pasalic, Anupam Chattopadhyay, and WeiGuo Zhang. Efficient implementation of generalized Maiorana-McFarland class of cryptographic functions. *J. Cryptogr. Eng.*, 7(4):287–295, 2017. 5

[47] Enes Pasalic, Samir Hodžić, Samed Bajriić, and Yongzhuang Wei. Optimizing the placement of tap positions. In Berna Örs and Bart Preneel, editors, *Cryptography and Information Security in the Balkans - First International Conference, BalkanCryptSec 2014, Istanbul, Turkey, October 16-17, 2014, Revised Selected Papers*, volume 9024 of *Lecture Notes in Computer Science*, pages 15–30. Springer, 2014. 2, 4, 17, 18

[48] Oscar S. Rothaus. On "bent" functions. *J. Comb. Theory, Ser. A*, 20(3):300–305, 1976. 3, 7

[49] Rainer A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer, 1986. 2, 16

[50] Palash Sarkar and Subhamoy Maitra. Efficient implementation of "large" stream cipher systems. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 319–332. Springer, 2001. 5

[51] Palash Sarkar and Subhamoy Maitra. Efficient implementation of cryptographically useful 'large' Boolean functions. *IEEE Trans. Computers*, 52(4):410–417, 2003. 5

[52] Thomas Siegenthaler. Cryptanalysts representation of nonlinearly filtered ML-sequences. In Franz Pichler, editor, *Advances in Cryptology - EUROCRYPT '85, Workshop on the Theory and Application of of Cryptographic Techniques, Linz, Austria, April 1985, Proceedings*, volume 219 of *Lecture Notes in Computer Science*, pages 103–110. Springer, 1985. 2, 18

[53] Thomas Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, 34(1):81–85, 1985. 2, 5, 18

[54] Yosuke Todo, Takanori Isobe, Willi Meier, Kazumaro Aoki, and Bin Zhang. Fast correlation attack revisited - cryptanalysis on full Grain-128a, Grain-128, and Grain-v1. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 129–159. Springer, 2018. 2, 5, 18, 19

[55] Qichun Wang, Claude Carlet, Pantelimon Stanica, and Chik How Tan. Cryptographic properties of the hidden weighted bit function. *Discret. Appl. Math.*, 174:1–10, 2014. 2

[56] Qichun Wang and Thomas Johansson. A note on fast algebraic attacks and higher order nonlinearities. In Xuejia Lai, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 6th International Conference, Inscrypt 2010, Shanghai, China, October 20-24, 2010, Revised Selected Papers*, volume 6584 of *Lecture Notes in Computer Science*, pages 404–414. Springer, 2010. 3

[57] Qichun Wang, Chik How Tan, and Pantelimon Stanica. Concatenations of the hidden weighted bit function and their cryptographic properties. *Adv. Math. Commun.*, 8(2):153–165, 2014. 2

[58] Yongzhuang Wei, Enes Pasalic, and Yupu Hu. Guess and determine attacks on filter generators - revisited. *IEEE Trans. Inf. Theory*, 58(4):2530–2539, 2012. 2, 4, 17, 18

[59] Bin Zhang, Ruitao Liu, Xinxin Gong, and Lin Jiao. Improved fast correlation attacks on the Sosemanuk stream cipher. *IACR Trans. Symmetric Cryptol.*, 2023(4):83–111, 2023. 2, 18, 19

[60] Zhaocun Zhou, Dengguo Feng, and Bin Zhang. Vectorial decoding algorithm for fast correlation attack and its applications to stream cipher Grain-128a. *IACR Trans. Symmetric Cryptol.*, 2022(2):322–350, 2022. 2, 18, 19