

The Nonlinear Filter Model of Stream Cipher Redivivus

Claude Carlet^{1,2} and Palash Sarkar^{*3}

¹LAGA Laboratory, University of Paris 8, 93526 Saint-Denis, France

²University of Bergen, Norway

³Indian Statistical Institute, 203, B.T. Road, Kolkata, India 700108

Emails: claude.carlet@gmail.com, palash@isical.ac.in

February 12, 2025

Abstract

The nonlinear filter model is an old and well understood approach to the design of secure 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 tap positions and the filter function, which gives great 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 $2^{\lceil \log_2 \lfloor n/2 \rfloor \rceil}$, 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 efficient implementable functions which provide the required level of resistance to fast algebraic and fast correlation attacks. Previously known efficiently implementable functions have too large linear bias making them inappropriate even with a large number of variables. 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 $1743.5, 2771.5, \text{ and } 5607.5$ 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 very efficient to implement, particularly 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 cryptanalyzing 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 [54] (which was efficient on another model, the nonlinear combiner model), a long line of papers [53, 45, 30, 3, 22, 36, 37, 20, 14, 38, 13, 39, 21, 55, 61, 60, 40, 41] explored various avenues for mounting correlation attacks. Surveys of some of the older attacks appear in [11, 12, 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 [24] and was soon followed by the publication of the fast algebraic attack [23]. 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 [18], and some others are fast to compute but possess insufficient nonlinearity [56, 58, 16, 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 nonlinear filter model. The inversion attack [28] 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 [28, 29, 34, 47, 59]. 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 positions of the LFSR that are fed as input to the filtering function. We note that such recommendations implicitly assume that the number of variables of the filtering function is much smaller 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 [33, 5, 27]). 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 [10, 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 [26] proposed) that use is not clear, after the invention of fast algebraic attacks. The introduction of Chapter 6 of [15] 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 [57] 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 [15] (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 the identity permutation. It is known [25] 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.

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 [28]) certain kinds of information leakage which was identified in [3]. On the negative side, filter functions of the form $W + g(\mathbf{Z})$ were shown to be susceptible to inversion attack [28]. We prevent this attack by choosing the gap between the tap positions to be sufficiently large; this countermeasure was already proposed in [28]. More generally, we show that due to the fact that we use filter functions on a large number of variables, the various kinds of state guessing attacks [29, 34, 47, 59] do not apply to our proposals.

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 well¹ with famous ciphers such as Trivium [10] 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 [51, 52, 31] 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, 19] on implementation of Maiorana-McFarland type functions. These works, however, do not cover the implementation of the functions that we consider, the reason being that these 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 [10], Mickey [4], Grain-128a [1], SNOW [27] 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* correlation of 2^{-60} , while at the same security level, for Sosemanuk [5] the best known [40] linear approximation has correlation $2^{-20.84}$, and it is not known whether there are approximations

¹Following [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.

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 [55] 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 [54] 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 no way to 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 [10, 4, 1, 27, 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 welcome further analysis and investigation of other avenues of 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. The earlier version of our proposal also had the same provable guarantees against the same classes of attacks as the present version. So while achieving provable resistance against certain well known classes of attacks removes some avenues of attacks, security of the proposal against all classes of attacks remains a conjecture.

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 [15].

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 $\mathbf{x} = (x_1, \dots, x_n)$ be an n -bit binary string. The weight of \mathbf{x} is $\text{wt}(\mathbf{x}) = \#\{i : x_i = 1\}$. By $\mathbf{0}_n$ and $\mathbf{1}_n$ we will denote the all-zero and all-one strings of length n respectively. Let $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$ 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 $\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 \oplus \dots \oplus x_n y_n$; and we define $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for $i = 1, \dots, n$.

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

Algebraic normal form. Let f be an n -variable function. The *algebraic normal form (ANF) representation* of f is the following: $f(X_1, \dots, X_n) = \bigoplus_{\alpha \in \mathbb{F}_2^n} a_\alpha \mathbf{X}^\alpha$, where $\mathbf{X} = (X_1, \dots, X_n)$; for $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_2^n$, \mathbf{X}^α denotes $X_1^{\alpha_1} \dots X_n^{\alpha_n}$; and $a_\alpha \in \mathbb{F}_2$. The (algebraic) degree of f is $\text{deg}(f) = \max\{\text{wt}(\alpha) : a_\alpha = 1\}$. 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 $\text{deg}(f) \leq n - 1$.

We have the following relations between the coefficients a_α in the ANF of f and the values of f (see for example Section 2.2 of [15]). For $\mathbf{x}, \alpha \in \mathbb{F}_2^n$,

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

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 $\text{nl}(f) = \min d(f, g)$, where the minimum is over all n -variable affine functions g .

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

A function f such that $W_f(\alpha) = \pm 2^{n/2}$ for all $\alpha \in \mathbb{F}_2^n$ 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 $\text{LB}(f) = 1/2 - \text{nl}(f)/2^n$.

Algebraic resistance. Let f be an n -variable function. The *algebraic immunity* of f is defined [24, 44] as follows: $\text{Al}(f) = \min_{g \neq 0} \{\text{deg}(g) : \text{either } gf = 0, \text{ or } g(f \oplus 1) = 0\}$. It is known [24] that $\text{Al}(f) \leq \lceil n/2 \rceil$.

The fast algebraic attack (FAA) was introduced in [23]. 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 [23] that there exists functions g and h with $\text{deg}(g) = e$ and $\text{deg}(h) \leq d$ such that $gf = h$. Based on this result, we provide the following definition. For each $e \in \{1, \dots, \text{Al}(f) - 1\}$, let $d \leq n - 1 - e$ be the maximum integer such that there do not exist n -variable functions g and h with $\text{deg}(g) = e$, $\text{deg}(h) = d$ and $gf = h$. We call the list of all such pairs (e, d) to be the *FAA-profile* of f .

The *fast algebraic immunity (FAI)* of f is a combined measure of resistance to both algebraic and fast algebraic attacks: $\text{FAI}(f) = \min(2\text{Al}(f), \min_{g \neq 0} \{\text{deg}(g) + \text{deg}(fg) : 1 \leq \text{deg}(g) < \text{Al}(f)\})$.

So $\text{FAI}(f) = \min(2\text{AI}(f), \min\{e + d + 1\})$, where the second minimum is taken over all pairs (e, d) in the FAA-profile of f . Further, for any function f , $1 + \text{AI}(f) \leq \text{FAI}(f) \leq 2\text{AI}(f)$.

Majority function. For $n \geq 1$, let $\text{Maj} : \{0, 1\}^n \rightarrow \{0, 1\}$ be the majority function defined in the following manner. For $\mathbf{x} \in \{0, 1\}^n$, $\text{Maj}(\mathbf{x}) = 1$ if and only if $\text{wt}(\mathbf{x}) \geq \lceil n/2 \rceil$. It is known [25] that Maj_n has the maximum possible AI of $\lceil n/2 \rceil$.

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 π_1, \dots, π_m be the coordinate functions of π . Let $\mathbf{X} = (X_1, \dots, X_m)$ and $\mathbf{Y} = (Y_1, \dots, Y_m)$. For $m \geq 1$, MM_{2m} is defined to be the following.

$$\begin{aligned} \text{MM}_{2m}(\mathbf{X}, \mathbf{Y}) &= \langle \pi(\mathbf{X}), \mathbf{Y} \rangle \oplus h(\mathbf{X}) \\ &= \pi_1(\mathbf{X})Y_1 \oplus \dots \oplus \pi_m(\mathbf{X})Y_m \oplus h(\mathbf{X}). \end{aligned} \quad (2)$$

Since MM_{2m} is bent, $\text{nl}(\text{MM}_{2m}) = 2^{2m-1} - 2^{m-1}$, and $\text{LB}(\text{MM}_{2m}) = 2^{-m-1}$. 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 $m \geq 1$, $\text{deg}(\text{MM}_{2m}) = \max(\text{deg}(\pi_1) + 1, \dots, \text{deg}(\pi_m) + 1, \text{deg}(h))$.

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

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

$$\text{AI}(\text{MM}_{2m}) \geq \text{wt}(\omega^*) + \text{AI}(\langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})).$$

Suppose that π is a bit permutation, i.e. $\pi(X_1, \dots, X_m) = (X_{\rho(1)}, \dots, X_{\rho(m)})$, for some permutation ρ of $\{1, \dots, m\}$. Then

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

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

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

Since the right hand side of (3) is equal to 0, for $\omega \in \mathbb{F}_2^m$, the coefficient of \mathbf{Y}^ω 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 ω^* with $\text{wt}(\omega^*) = w$ and $g_{\omega^*}(\mathbf{X}) \neq 0$. In (3), equating the coefficient of \mathbf{Y}^{ω^*} to 0, we have

$$0 = g_{\omega^*}(\mathbf{X}) \left(h(\mathbf{X}) \oplus \left(\bigoplus_{i \in \text{supp}(\omega^*)} \pi_i(\mathbf{X}) \right) \right)$$

$$= g_{\omega^*}(\mathbf{X})(\langle \omega, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})).$$

So $g_{\omega^*}(\mathbf{X})$ is an annihilator for $\langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})$. Consequently, $\deg(g) \geq \text{wt}(\omega^*) + \deg(g_{\omega^*}) \geq \text{wt}(\omega^*) + \text{Al}(\langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}))$.

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

Suppose now that π is a bit permutation. Then $\ell(\mathbf{X}) = \langle \omega^*, \pi(\mathbf{X}) \rangle$ is a linear function. If $w = 0$, then $\langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) = h(\mathbf{X})$, and so we have the result. So suppose $w > 0$. From Lemma 1 of [17], we have $\text{Al}(\ell(\mathbf{X}) \oplus h(\mathbf{X})) \geq \text{Al}(h(\mathbf{X})) - 1$. So

$$\begin{aligned} \text{Al}(\text{MM}_{2m}) &\geq \text{wt}(\omega^*) + \text{Al}(\ell(\mathbf{X}) \oplus h(\mathbf{X})) \\ &\geq w + \text{Al}(h(\mathbf{X})) - 1 \geq \text{Al}(h(\mathbf{X})). \end{aligned}$$

□

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

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

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

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

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

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. $1 \oplus \text{MM}_{2m}$), then $(1 \oplus W)g(\mathbf{Z})$ is an annihilator for MM_{2m+1} (resp. $1 \oplus \text{MM}_{2m+1}$). This shows the upper bound. Next we consider the lower bound. Suppose $g(W, \mathbf{Z}) \neq 0$ is an annihilator for $\text{MM}_{2m+1}(W, \mathbf{Z})$. We write $g(W, \mathbf{Z})$ as $g(W, \mathbf{Z}) = Wg_1(\mathbf{Z}) + g_0(\mathbf{Z})$. Noting that $\text{MM}_{2m+1}(W, \mathbf{Z}) = W \oplus \text{MM}_{2m}(\mathbf{X}, \mathbf{Y})$, we obtain

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

So $g_0(\mathbf{Z})\text{MM}_{2m}(\mathbf{Z}) = 0$ and $g_0(\mathbf{Z}) \oplus g_1(\mathbf{Z})(1 \oplus \text{MM}_{2m}(\mathbf{Z})) = 0$. If g_0 is non-zero, then g_0 is an annihilator for MM_{2m} and so $\deg(g) \geq \deg(g_0) \geq \text{Al}(\text{MM}_{2m})$. 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 $1 \oplus \text{MM}_{2m}$, and so $\deg(g) \geq 1 + \deg(g_1) \geq 1 + \text{Al}(\text{MM}_{2m})$. Consequently, in both cases $\deg(g) \geq \text{Al}(\text{MM}_{2m})$.

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

From Theorem 1, choosing π to be any bit 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 reverse permutation, i.e. $\pi(X_1, \dots, X_m) = (X_m, \dots, X_1)$.

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. There are other functions which achieve maximum algebraic immunity [17] 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.

Remark 1 *Theorem 1 assures us of the lower bound on the (fast) algebraic immunity of MM_{2m} when π is any bit permutation. All the $m!$ bit permutations π result in functions with the same implementation efficiency. However, some choices of π can lead to an attack which is not an algebraic attack. In a previous version, we had used π as the identity permutation which lead to an attack [6]. See Section 4.1 for a description of how the choice of π as the identity function leads to an attack.*

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

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

Proof: The degree of Maj_m is $2^{\lceil \log_2 m \rceil}$ (see [25]). 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 $\text{AI}((\text{Maj}, \text{rev})\text{-MM}_n) \geq \lceil n/4 \rceil$, and if n is odd then $\text{AI}((\text{Maj}, \text{rev})\text{-MM}_n) \geq \lceil (n-1)/4 \rceil$.*

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

We computed the algebraic immunity of $(\text{Maj}, \text{rev})\text{-MM}_n$ for small values of n and observed that for $n \geq 4$, $\text{AI}((\text{Maj}, \text{rev})\text{-MM}_n) = 1 + \lfloor n/4 \rfloor$. This suggests that the lower bound in Proposition 4 is exact if $n \not\equiv 0 \pmod{4}$ and if $n \equiv 0 \pmod{4}$, the actual algebraic immunity is one more than the lower bound. As mentioned in Section 2, for any Boolean function f , $\text{FAI}(f) \geq 1 + \text{AI}(f)$. Our experiments suggest that for $(\text{Maj}, \text{rev})\text{-MM}_n$, the fast algebraic immunity is actually one more than the algebraic immunity.

So $(\text{Maj}, \text{rev})\text{-MM}_n$ 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.

4 Concrete Proposals

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 $(\text{Maj}, \text{rev})\text{-MM}_{2m+1}$, i.e. where h is chosen to be Maj_m and π is chosen to be rev .

An LFSR is a linear map, so the all zero-state will lead to all-zero output of the LFSR. MM_{2m+1} maps the all-zero string to 0, and so the keystream generated from the all zero-state of the LFSR will be the all-zero keystream. To avoid this scenario, we define the function $f_{2m+1} : \{0, 1\}^{2m+1} \rightarrow \{0, 1\}$ to be

$$f_{2m+1}(W, \mathbf{X}, \mathbf{Y}) = 1 \oplus \text{MM}_{2m+1}(W, \mathbf{X}, \mathbf{Y}). \quad (5)$$

The cryptographic properties of f_{2m+1} are exactly the same as those of MM_{2m+1} .

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 $L \geq 2\kappa$. Suppose that the state (s_{L-1}, \dots, s_0) of the LFSR leads to the next state (s_L, \dots, s_1) , i.e. the state is right shifted once and the next bit s_L becomes the leftmost bit. The initial state of the LFSR is constructed as $(k_{\kappa-1}, \dots, k_0, v_{\kappa-1}, \dots, v_0, b_1, \dots, b_{L-2\kappa})$, where $(k_{\kappa-1}, \dots, k_0)$ is the key, $(v_{\kappa-1}, \dots, v_0)$ is the IV, and $(b_1, \dots, b_{L-2\kappa}) \neq \mathbf{0}_{L-2\kappa}$ is an arbitrary constant bit string.

We next specify the tap positions, i.e. which state bits of the LFSR are fed into the filtering function. We assume that $2m < \kappa$, which holds for all the concrete choices that we specify later. From the state (s_{L-1}, \dots, s_0) of the LFSR,

- the value of W in $f_{2m+1}(W, \mathbf{X}, \mathbf{Y})$ is the state bit $s_{L-\kappa+2m}$,
- the values of the variables in \mathbf{X} are the state bits $s_{L-\kappa+2(m-1)}, s_{L-\kappa+2(m-2)} \dots, s_{L-\kappa+2}, s_{L-\kappa}$,
- and the values of the variables in \mathbf{Y} are the state bits $s_{L-2\kappa+2(m-1)}, s_{L-2\kappa+2(m-2)} \dots, s_{L-2\kappa+2}, s_{L-2\kappa}$.

So if we write the state vector $\mathbf{s} = (s_{L-1}, \dots, s_0)$ as \mathbf{s}_1 , \mathbf{s}_2 and \mathbf{s}_3 , where \mathbf{s}_1 and \mathbf{s}_2 are κ -bit strings and \mathbf{s}_3 is a $(L - 2\kappa)$ -bit string, then the tap positions for W and \mathbf{X} are obtained by starting from the rightmost bit of \mathbf{s}_1 and moving in steps of two to the left for $m + 1$ steps, and the tap positions for \mathbf{Y} are obtained by starting from the rightmost bit of \mathbf{s}_2 and moving in steps of two to the left for m steps. Note that the tap positions are divided into two clusters, the first cluster consisting of $m + 1$ tap positions separated from each other by a step of two, the second cluster consisting of m tap positions which are also separated from each other by a step of two, and there is a gap of $\kappa - 2m$ positions between the two clusters. Our choice of tap positions does not follow the recommendation given in the literature [28, 29, 34, 47, 59]. Nonetheless, we later argue that the design can resist state guessing attacks. This is due to the fact that we use Boolean functions on many more variables than what has been proposed in the literature. Let M be the length of the interval between the first tap position for \mathbf{X} and the last tap position for \mathbf{Y} , including the end points. Then $M = \kappa + 2m$. The quantity M will be later required to quantify resistance to inversion attacks [28].

We assume that the stream cipher goes through an initialisation phase of 2κ steps, where no keystream bit is produced, and instead the keystream bit is fed back to the LFSR in the following manner. Let μ be a positive integer. Suppose $\mathbf{s} = (s_{L-1}, \dots, s_0)$ is a state of the LFSR at some step (which is less than 2κ) during the initialisation phase. From \mathbf{s} , let the LFSR feedback bit be c and the output of the filtering function be b . Then the next state of the LFSR is $\mathbf{s}' = (s'_{L-1}, \dots, s'_0)$, where $s'_{L-1} = c \oplus b$, and for $i = L - 2, L - 3, \dots, 0$, $s'_i = s_{i+1} \oplus b$ if $i \geq L - 2\kappa$ and $i \equiv 0 \pmod{\mu}$, and $s'_i = s_{i+1}$, otherwise. Note that the output of the filtering function is injected into the state at $\lceil 2\kappa/\mu \rceil$ points requiring $\lceil 2\kappa/\mu \rceil$ XOR gates. We propose $\mu = \lfloor \sqrt{2\kappa} \rfloor$ so that the feedback during the initialisation phase does not require too many XOR gates. After the initialisation phase the leftmost 2κ bits of the LFSR are complex nonlinear functions of the key and the IV. Keystream generation starts after the initialisation phase.

Remark 2 *The above described tapping positions and initialisation method is in response to an attack [6] on an earlier proposal which used a different set of tap positions and a different initialisation method. See Section 4.1 for a description of the earlier proposal and how it was exploited in the attack in [6].*

The parameters L and m are the *design parameters* of the stream cipher. The goal is to choose these parameters so as to resist well known classes of known plaintext attacks which result in recovering the key. 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 $L \geq 2\kappa$ 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 [35]. 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 $N \leq T$, 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 $B \leq \kappa$. In the analysis below, our goal is to determine values of L and m such that the stream cipher is not vulnerable to the known types of attacks. For this 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 (Maj, rev)-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

$$\alpha = \binom{L}{2^{\lfloor \log_2 m \rfloor}} > 2^B, \quad (6)$$

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 [28] provides a sufficient condition to ensure this property. The theorem states that if the n -variable filtering function is of the form $Z_1 + g(Z_2, \dots, Z_n)$, 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 [28] and is called the inversion attack. The inversion attack is specifically applicable to filter functions of the form $X \oplus g(\mathbf{Z})$. Since the filter function that we use is indeed of this form, we need to consider the resistance of the stream cipher to the inversion attack. Consider the bit sequence generated by the LFSR to be s_0, s_1, s_2, \dots and the corresponding keystream bits be k_0, k_1, k_2, \dots . Due to the placement of the tap positions, the i -th keystream bit k_i is determined by the bits $s_i, s_{i-1}, \dots, s_{i-M}$ (see above for the definition of M). Further, we have $k_i = s_i \oplus g(s_{i-1}, \dots, s_{i-M})$, where g is defined from MM $_{2m}$

as a function of M variables which is degenerate on the $\kappa - m$ variables. So we have $s_i = k_i \oplus g(s_{i-1}, \dots, s_{i-M})$, i.e. the i -th state bit is determined by the known keystream bit k_i and the values of the bits s_{i-1}, \dots, s_{i-M} . Once s_i is obtained, s_{i+1} is obtained as $s_{i+1} = k_{i+1} \oplus g(s_i, \dots, s_{i-M+1})$, similarly, from s_{i+1} , one obtains s_{i+2} and so on. In other words, if for some i , the values of s_{i-1}, \dots, s_{i-M} are known, then the LFSR sequence can be computed in the forward direction. The attack proceeds by guessing the values of s_{i-1}, \dots, s_{i-M} , computing the LFSR sequence for L steps to determine the state of the LFSR and then using this state to generate a keystream, if the generated keystream matches with the obtained keystream, then the guess is correct. Since there are M bits whose values have to be guessed, the method requires 2^M guesses and about Lm operations for each guess. For $\mathcal{S}(L, m)$, the value of M is $\kappa + m$. So the inversion attack requires more than $2^{\kappa+2m}$ operations on $\mathcal{S}(L, m)$. This proves the resistance of $\mathcal{S}(L, m)$ to the inversion attack. Note that the inversion attack is prevented due to M being sufficiently large. This was one of the countermeasures to inversion attack that was already proposed in [28].

The inversion attack was later extended to the generalised inversion attack, the filter state guessing attack, and the generalised filter state guessing attack [28, 29, 34, 47, 59]. As one of the countermeasures to these attacks, some design criteria, such as full positive difference set, for choosing tap positions have been suggested. Choosing the tap positions to be consecutive has been mentioned to be the worst possible choice [34]. Our choice of tap positions consists of two clusters where the tap positions in each cluster are consecutive. So based on the recommendations in the literature, our choice of tap positions is close to the worst possible design choice. Nonetheless, we argue that $\mathcal{S}(L, m)$ resists all kinds of state guessing attacks. The reason is that we use a filtering function on many more variables than has been considered in the literature. All the recommendations in the literature implicitly assume that the number of variables of the filtering function is much smaller than the target security level κ . Since we use filtering functions with number of variables close to κ , the recommendations on tap positions in the literature do not apply to our case.

Let n 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 initial state bits of the LFSR. So knowing the values of the n input bits at any point of time provide n linear equations in L variables. If the values of the input bits 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 input bits to the filtering function at any point of time. There are 2^n possible values of these input bits. The keystream bits k_1, k_2, \dots , i.e. the outputs of the filtering function are known. Since the filtering function is balanced, observing one bit of the keystream, say k_i , rules out half of the possible values of the corresponding input bits to the filtering function, i.e. there are 2^{n-1} possible values of the input bits that give rise to the observed keystream bit. So guessing one of these 2^{n-1} values provides n linear equations in L variables. An input bit to the filtering function at time point i may also be an input bit to the filtering function at a few other time points. For $\mathcal{S}(L, m)$ the two sets of input bits to the filtering function at two successive time points i and $i + 1$ have $2m - 1$ bits in common. Suppose we have guessed the values of the input bits for time point i . Then the number of possible values for the input bits at time point $i + 1$ is 4. Observing k_{i+1} and again using the fact that the filtering function is balanced, the number of possible values for the input bits at time point $i + 1$ reduces to 2. So the observed keystream bits k_i and k_{i+1} together restrict the number of possibilities of values of the input bits at time point i and at time point $i + 1$ to be 2^n . By guessing one of these 2^n values, we obtain a system of $2n$ linear equations in L variables. Extending this idea, we see that by guessing one of 2^{n+c-1} values of the c consecutive values of the input bits at time points $i, i + 1, \dots, i + c - 1$, we obtain a system of nc equations in L variables. If $n + c > L$, then we can

solve the linear system and obtain the initial state of the LFSR. Solving the linear system requires L^3 operations. Since a linear system has to be solved for each guess, the complexity of the attack is $2^{n+c-1}L^3$. Suppose the following condition holds.

$$2^{n+c-1}L^3 > 2^\kappa \text{ where } c \text{ is the least positive integer such that } n+c > L. \quad (7)$$

Then no form of the state guessing attack succeeds against $\mathcal{S}(L, m)$ at the κ -bit security level.

In the above analysis, we saw that the tap positions of $\mathcal{S}(L, m)$ (two clusters, with consecutive positions in each cluster) result in the fact that once the values of the input bits to the i -th position is guessed, there are only two possible values for the input bits to the $(i+1)$ -st position. This is due to the fact of the huge overlap between the sets of input bits in the i -th and $(i+1)$ -st positions. The recommendations for tap positions in the literature [28, 29, 34, 47, 59] 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 $1+2+\dots+(n-1) = n(n-1)/2$ (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 (7) as the condition to resist such attacks. This allows use 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 [54] 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 [53] 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, 36, 37, 20, 14, 38, 13, 21] as well as more recent papers such as [55, 61, 60, 40, 41]. See [11, 12, 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, 14], 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.

$$L > (2m + 2) \text{ and either } B < 2(m + 2) \text{ or } \kappa < \frac{2(m + 2)L + B^2}{B - 2(m + 2)}. \quad (8)$$

Note that (8) 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, [11] identifies the key idea to be from [20]. The attack in [20] 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 $k \in \{1, \dots, L\}$ 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.

$$B \leq 2(2m + 3). \quad (9)$$

Type of attack. The attacks in [38, 13] 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 [20] (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.

$$B < (\kappa + 4m)/2. \quad (10)$$

Other fast correlation attacks. We next consider some of the more recent attacks. The attack in [55] 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 [55]. For $\kappa = 128$, Table 1 recommends $L = 257$ and $m = 58$. To apply the attack in [55] 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 [55] to $\mathcal{S}(257, 59)$.

Subsequent works (such as [61, 60, 40, 41]) 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], which is another cipher providing 128-bit security. 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 [40] 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 [61, 60, 40, 41] inapplicable at the stated security levels.

(Fast) Algebraic attacks. For a filtering function f having $\text{Al}(f) = a$, 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 [15]). We assume $\omega = 2.8$. From Proposition 4, we conservatively assume that $\text{Al}((\text{Maj, rev})\text{-MM}_{2m+1}) = \lceil m/2 \rceil$, i.e. the algebraic immunity is equal to its lower bound. Let

$$\beta = \left(\sum_{i=0}^{\lceil m/2 \rceil} \binom{L}{i} \right)^{2.8}. \quad (11)$$

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

For any (e, d) in the FAA-profile of $(\text{Maj, rev})\text{-MM}_{2m}$, 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 [32] and Section 3.1.5 of [15]). This complexity dominates the overall time complexity of a fast algebraic attack and ignoring the logarithm term, and we take T to be equal to ED . The maximum value of e is one less than the algebraic immunity. The number N of keystream bits required is about $2E$. From Proposition 4, we again assume that $\text{Al}((\text{Maj, rev})\text{-MM}_{2m+1}) = \lceil (2m+1)/4 \rceil$. Recall that for any Boolean function, its fast algebraic immunity is at least one more than its algebraic immunity, and so $\text{FAI}((\text{Maj, rev})\text{-MM}_{2m+1}) \geq 1 + \text{Al}((\text{Maj, rev})\text{-MM}_{2m+1}) = 1 + \lceil (2m+1)/4 \rceil$. For the concrete analysis, we use the lower bound $1 + \lceil m/2 \rceil$ as the actual value of $\text{FAI}((\text{Maj, rev})\text{-MM}_{2m+1})$. Define

$$\gamma = \min_{\substack{1 \leq e \leq \lceil m/2 \rceil, \\ d = a - e}} \left(\sum_{i=0}^e \binom{L}{i} \right) \left(\sum_{i=0}^d \binom{L}{i} \right). \quad (12)$$

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

Concrete choices of L and m . Given κ , we obtained representative values of L and m . The procedure we followed 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. The number of variables of the filtering function is $n = 2m + 1$. For each of the obtained values of L and n , we checked whether (7) holds. In each case we found that (7) indeed holds. So the obtained values of L and m ensure security against state guessing attacks. 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 (8), (9) and (10) 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.

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.

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 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 expected 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.

4.1 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}, \dots, 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 $s_{L-\kappa+m-1}, \dots, s_{L-\kappa}$, and the values of the variables in \mathbf{Y} were chosen to be the state bits $s_{L-2\kappa+m-1}, \dots, s_{L-2\kappa}$.
3. In the previous version, during the initialisation phase, the feedback from the filter function was fed back into the state by XORing it with the next bit of the LFSR.

Let s_0, s_1, \dots be the bit sequence produced by the LFSR and let k_0, k_1, \dots be the keystream. Then

$$\begin{aligned}
k_i &= f(s_i, \dots, s_{i-L+1}) \\
&= s_{i-\kappa+m} \oplus \langle \pi(s_{i-\kappa+m-1}, \dots, s_{i-\kappa+1}, s_{i-\kappa}), (s_{i-2\kappa+m-1}, \dots, s_{i-2\kappa+1}, s_{i-2\kappa}) \rangle \\
&\quad \oplus \text{Maj}_m(s_{i-\kappa+m-1}, \dots, s_{i-\kappa+1}, s_{i-\kappa}),
\end{aligned}$$

where f is the filter function. The first two of the above mentioned aspects of the previous proposal created the following undesirable situation.

$$\begin{aligned}
k_i \oplus k_{i+1} &= s_{i-\kappa+m-1} s_{i-2\kappa+m-1} \oplus s_{i+1-\kappa} s_{i+1-2\kappa} \oplus \\
&\quad \text{Maj}_m(s_{i-\kappa+m-1}, \dots, s_{i-\kappa+1}, s_{i-\kappa}) \oplus \text{Maj}_m(s_{i-\kappa+m}, \dots, s_{i-\kappa+2}, s_{i-\kappa+1}).
\end{aligned}$$

In other words, since π was chosen to be the identity permutation, and two successive keystream bits are obtained from a single shift of the LFSR sequence, $m-1$ of the terms in the inner product cancelled out when k_i and k_{i+1} are XORed together. This feature was 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 $k_i \oplus k_{i+1}$, this leads to an efficient key recovery attack.

In the present version, we have chosen π to be the bit reversal permutation. Also, we have chosen the tap positions to differ by two positions, and so one should look at $k_i \oplus k_{i+2}$ rather than $k_i \oplus k_{i+1}$. (Considering $k_i \oplus k_{i+1}$ results in quadratic terms forming a bent function on $4m$ variables.) With π as the bit reversal permutation, no term of k_i cancels with any term of k_{i+2} . So the simplification obtained when π is the identity permutation cannot be achieved when π is the bit reversal permutation. More generally, with π as the bit reversal permutation, in the XOR of any number of key bits, no quadratic term cancels out.

Further, in the present version, during the initialisation phase we inject the output of the filter 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.

The other modification that we have introduced is to choose the tap positions to be two places apart. Since for all the proposals in Table 1, $2m$ is quite close to κ , the tap positions are spread more or less evenly over the entire κ positions used to load the IV. As a result, any difference introduced in any bit

of the IV will appear at the input of the filtering function within the first few initialisation steps. Then as argued above this difference will be updated about $\sqrt{2\kappa}$ times during the whole initialisation phase.

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. So our present proposal is to be considered to be more of an “engineering” solution rather than a mathematical one in response to the attack in [6]. We welcome further analysis of our proposal, including finding other attack avenues.

5 Efficiency of Computing MM_n

In Section 4, we proposed using f_{2m+1} (which is either MM_{2m+1} or $1 \oplus \text{MM}_{2m+1}$) 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 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 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

	$\mathcal{S}(163, 36)$	$\mathcal{S}(257, 59)$	$\mathcal{S}(331, 70)$	$\mathcal{S}(389, 87)$	$\mathcal{S}(449, 100)$	$\mathcal{S}(521, 114)$
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 3: Estimates of the number of NAND gates required to implement $\mathcal{S}(L, m)$ for values of L and m in Table 1.

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.

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

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. Similar assumptions were made in [1] to obtain the gate count estimate for Grain-128a. Further, we also ignore the $\lceil 2\kappa/\mu \rceil$ XOR gates required for the feedback injection from the filter 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 filter 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, 10] 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, 10], 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 [10] there is a difference in the number of units required for a flip-flop: [1] takes a flip-flop to be 8 units, while [10] takes a flip-flop to be 12 units. In Table 3, 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 [10], then the gate count estimate for $\mathcal{S}(163, 36)$ is 2395.5. The gate estimate for Trivium obtained in [10] is 3488. Both Trivium and $\mathcal{S}(163, 36)$ are targeted at the 80-bit security level, and $\mathcal{S}(163, 36)$ is substantially smaller than Trivium. The lower size of $\mathcal{S}(163, 36)$ is due to the smaller state size; $\mathcal{S}(163, 36)$ 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 shows that the circuit sizes compare well with well known 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 Pierrick Méaux for his comments on an earlier version 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. 4, 5, 14, 19
- [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, 13
- [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, 11
- [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, 4, 5

- [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, 4, 5, 15
- [6] Tim Beyne and Michiel Verbauwhede. Cryptanalysis of a nonlinear filter-based stream cipher. Cryptology ePrint Archive, Paper 2025/197, 2025. 5, 9, 11, 16, 17, 18
- [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. 14
- [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>. 18
- [9] Randal E. Bryant. On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication. *IEEE Trans. Computers*, 40(2):205–213, 1991. 27
- [10] 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, 4, 5, 19
- [11] 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, 13, 14
- [12] 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, 13
- [13] 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, 13, 14
- [14] 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, 13
- [15] Claude Carlet. *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press, 2021. 3, 5, 6, 15
- [16] Claude Carlet. A wide class of Boolean functions generalizing the hidden weight bit function. *IEEE Trans. Inf. Theory*, 68(2):1355–1368, 2022. 2
- [17] 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. 8, 9

- [18] 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
- [19] 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. 4
- [20] 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, 13, 14
- [21] 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, 13
- [22] 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
- [23] 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, 6
- [24] 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, 6
- [25] 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, 9
- [26] 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, 25
- [27] 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, 4, 5

- [28] 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, 10, 11, 12, 13
- [29] 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, 10, 12, 13
- [30] 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 Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 472–476. Springer, 1992. 2
- [31] 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. 4
- [32] 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 Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2004. 15
- [33] 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
- [34] 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, 10, 12, 13
- [35] 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. 11
- [36] 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, 13
- [37] 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, 13
- [38] Fredrik Jönsson and Thomas Johansson. A fast correlation attack on LILI-128. *Inf. Process. Lett.*, 81(3):127–132, 2002. 2, 13, 14

- [39] 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
- [40] 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, 4, 13, 14, 15
- [41] Isaac Andrés Canales Martínez and Igor Semaev. Multivariate correlation attacks and the cryptanalysis of lfsr-based stream ciphers. *Des. Codes Cryptogr.*, 92(11):3391–3427, 2024. 2, 13, 14, 15
- [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, 13
- [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. 6
- [45] Willi Meier and Othmar Staffelbach. Fast correlation attacks on certain stream ciphers. *J. Cryptol.*, 1(3):159–176, 1989. 2, 13
- [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. 4
- [47] Enes Pasalic, Samir Hodžić, Samed Bajrić, 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, 10, 12, 13
- [48] Oscar S. Rothaus. On “bent” functions. *J. Comb. Theory, Ser. A*, 20(3):300–305, 1976. 3, 6
- [49] Rainer A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer, 1986. 2, 11
- [50] Palash Sarkar and Subhamoy Maitra. Construction of nonlinear Boolean functions with important cryptographic properties. 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 485–506. Springer, 2000. 27

- [51] 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. 4
- [52] Palash Sarkar and Subhamoy Maitra. Efficient implementation of cryptographically useful 'large' boolean functions. *IEEE Trans. Computers*, 52(4):410–417, 2003. 4
- [53] 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 Cryptographic Techniques, Linz, Austria, April 1985, Proceedings*, volume 219 of *Lecture Notes in Computer Science*, pages 103–110. Springer, 1985. 2, 13
- [54] Thomas Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, 34(1):81–85, 1985. 2, 5, 13
- [55] 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, 13, 14
- [56] 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, 27
- [57] 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
- [58] 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
- [59] 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, 10, 12, 13
- [60] 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, 13, 14, 15
- [61] 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, 13, 14, 15

A Improved Degree and Algebraic Resistance

Dobbertin [26] had proposed a method for modifying normal bent functions to construct highly nonlinear balanced Boolean functions on an even number of variables. The method is recursive and requires

balanced Boolean functions on an odd number of variables. Suppose that for odd n , Bal_n is a balanced function on n variables. Dobbertin's recursive method applied to the Maiorana-McFarland bent functions is the following.

Suppose π is the permutation used to define MM_{2m} . Let $\mathbf{a} = (a_1, \dots, a_m) = \pi^{-1}(\mathbf{0}_m)$. We define $\mathbf{1}_{\mathbf{a}}(\mathbf{X}) = (1 \oplus a_1 \oplus X_1) \cdots (1 \oplus a_m \oplus X_m)$. If π is a bit permutation, then $\mathbf{a} = \mathbf{0}_m$ and $\mathbf{1}_{\mathbf{a}}(\mathbf{X})$ is simply $(1 \oplus X_1) \cdots (1 \oplus X_m)$. For $m \geq 1$, Bal_{2m} is defined as follows.

$$\text{Bal}_{2m}(\mathbf{X}, \mathbf{Y}) = \text{MM}_{2m}(\mathbf{X}, \mathbf{Y}) \oplus \mathbf{1}_{\mathbf{a}}(\mathbf{X})\text{Bal}_m(\mathbf{Y}). \quad (13)$$

The recursive definition in (13) terminates when m is odd. It is easy to see that for $m \geq 1$, $\deg(\text{Bal}_{2m}) = m + \deg(\text{Bal}_m)$, and $\text{nl}(\text{Bal}_{2m}) = 2^{2m-1} - 2^m + \text{nl}(\text{Bal}_m)$. The following result provides a lower bound on the algebraic immunity of Bal_{2m} .

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

$$\text{Al}(\text{Bal}_{2m}) \geq \text{wt}(\omega^*) + \text{Al}(\langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \oplus \text{Bal}_m(\omega^*)\mathbf{1}_{\mathbf{a}}(\mathbf{X})).$$

Proof: Suppose $g(\mathbf{X}, \mathbf{Y})$ is an annihilator for $\text{Bal}_{2m}(\mathbf{X}, \mathbf{Y})$, where $g(\mathbf{X}, \mathbf{Y}) = \bigoplus_{\omega \in \mathbb{F}_2^m} \mathbf{Y}^\omega g_\omega(\mathbf{X})$ for some functions $g_\omega(\mathbf{X})$. Also, we write $\text{Bal}_m(\mathbf{Y}) = \bigoplus_{\omega \in \mathbb{F}_2^m} b_\omega \mathbf{Y}^\omega$, where $b_\omega \in \mathbb{F}_2$. We have

$$\begin{aligned} 0 &= g(\mathbf{X}, \mathbf{Y})\text{Bal}_{2m}(\mathbf{X}, \mathbf{Y}) \\ &= \left(\bigoplus_{\omega \in \mathbb{F}_2^m} \mathbf{Y}^\omega g_\omega(\mathbf{X}) \right) \\ &\quad \left(\pi_1(\mathbf{X})Y_1 \oplus \cdots \oplus \pi_m(\mathbf{X})Y_m \oplus h(\mathbf{X}) \oplus \mathbf{1}_{\mathbf{a}}(\mathbf{X}) \bigoplus_{\omega \in \mathbb{F}_2^m} b_\omega \mathbf{Y}^\omega \right). \end{aligned} \quad (14)$$

Since the right hand side of (14) is equal to 0, all coefficients of \mathbf{Y}^ω in the expansion on the right hand side of (14) 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 ω^* with $\text{wt}(\omega^*) = w$ and $g_{\omega^*}(\mathbf{X}) \neq 0$. In (14), equating the coefficient of \mathbf{Y}^{ω^*} to 0, we have

$$\begin{aligned} 0 &= g_{\omega^*}(\mathbf{X}) \left(h(\mathbf{X}) \oplus \left(\bigoplus_{i \in \text{supp}(\omega^*)} \pi_i(\mathbf{X}) \right) \oplus \left(\mathbf{1}_{\mathbf{a}}(\mathbf{X}) \bigoplus_{\omega \leq \omega^*} b_\omega \right) \right) \\ &= g_{\omega^*}(\mathbf{X}) (\langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \oplus \text{Bal}_m(\omega^*)\mathbf{1}_{\mathbf{a}}(\mathbf{X})). \end{aligned}$$

Here we have used $\text{Bal}_m(\omega^*) = \bigoplus_{\omega \leq \omega^*} b_\omega$ (see (1)). So $g_{\omega^*}(\mathbf{X})$ is an annihilator for $\langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \oplus \text{Bal}_m(\omega^*)\mathbf{1}_{\mathbf{a}}(\mathbf{X})$. Consequently, $\deg(g) \geq \text{wt}(\omega^*) + \deg(g_{\omega^*}) \geq \text{wt}(\omega^*) + \text{Al}(\langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \oplus \text{Bal}_m(\omega^*)\mathbf{1}_{\mathbf{a}}(\mathbf{X}))$.

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

The lower bound on $\text{Al}(\text{MM}_{2m})$ is in terms of the AI of the function $f = \langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})$, while the lower bound on $\text{Al}(\text{Bal}_{2m})$ is in terms of the AI of the function $g = \langle \omega^*, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \oplus \text{Bal}_m(\omega^*)\mathbf{1}_{\mathbf{a}}(\mathbf{X})$. Note that f and g differ in at most one bit, which shows that the AI of $\text{Al}(\text{MM}_{2m})$ and $\text{Al}(\text{Bal}_{2m})$ are quite close.

The computation of MM_{2m} requires the computation of several Maiorana-McFarland functions on a smaller number of variables, terminating with the computation of a balanced function on an odd

number of variables. If we instantiate the function h with the majority function, then the computation of MM_{2m} requires several inner product and majority function computations on smaller number of variables. These push up the circuit size of Bal_{2m} in comparison to that of MM_{2m} , but do not lead to any substantial gain in the AI.

With the choice of π to be the identity permutation and h to be Maj_m in (2), the degree of MM_n is $2^{\lceil \log_2 \lfloor n/2 \rfloor \rceil}$ and the algebraic immunity is about $\lfloor n/4 \rfloor$. We discuss ways to improve these parameters.

For $n \geq 1$, let $\text{HWB}_n : \{0, 1\}^n \rightarrow \{0, 1\}$ be the hidden weight bit function [9] defined as follows. For $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{F}_2^n$, $\text{HWB}_n(\mathbf{x}) = 0$ if $\text{wt}(\mathbf{x}) = 0$, and $\text{HWB}_n(\mathbf{x}) = x_{\text{wt}(\mathbf{x})}$, if $\text{wt}(\mathbf{x}) > 0$. In [56], it was shown that the AI of HWB_n is at least $\lfloor n/3 \rfloor + 1$. We extend HWB_n to n -HWBP : $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ as follows. For $(x_1, \dots, x_n) \in \mathbb{F}_2^n$, let $w = \text{wt}(x_1, \dots, x_n)$. Then n -HWBP (x_1, \dots, x_n) is defined to be the following. n -HWBP $(x_1, \dots, x_n) = (0, \dots, 0)$ if $w = 0$, and n -HWBP $(x_1, \dots, x_n) = (x_w, x_{w+1}, \dots, x_n, x_1, \dots, x_{w-1})$, if $\text{wt}(\mathbf{x}) > 0$. It is easy to verify that n -HWBP is a bijection.

Suppose we set π in the definition of MM_{2m} to be m -HWBP (along with choosing h to be Maj_m), and define MM_{2m+1} from MM_{2m} as in (4). This leads to an improvement of algebraic resistance. We conducted experiments to compute the AI and FAI of MM_n for n up to 20. The results are shown in Tables 4 and 5, and suggest that $\lfloor n/3 \rfloor \leq \text{AI}(\text{MM}_n) \leq 1 + \lfloor n/3 \rfloor$, and $\lfloor n/2 \rfloor \leq \text{FAI}(\text{MM}_n) \leq 1 + \lfloor n/2 \rfloor$.

A construction of balanced functions on $2m + 1$ variables which has nonlinearity $2^{2m} - 2^m$ and the maximum possible degree $2m$ is given by Theorem 10(a) of [50]. This construction requires bent functions on smaller number of variables. Suppose we choose these bent functions to MM_{2m} with π built from HWBP and h built from the majority function on appropriate number of variables. For odd n , let us define Bal_n to be such a function. Suppose that the terminating function in (13) is chosen to be Bal_n . The net result is that for any n (either odd or even), it is possible to obtain balanced functions with maximum degree and very high nonlinearity. We have implemented this construction and computed the algebraic immunities and fast algebraic immunities of the resulting functions. The results of our experiments for n up to 20 are shown in Tables 4 and 5, and again we observe that the algebraic immunity is either $\lfloor n/3 \rfloor$, or $1 + \lfloor n/3 \rfloor$, and the fast algebraic immunity is either $\lfloor n/2 \rfloor$ or $1 + \lfloor n/2 \rfloor$.

Even though Bal_n has higher degree and higher algebraic immunities, we did not opt for Bal_n . The reason is the implementation efficiency. The modifications make the functions more complex, which results in more gate requirements. In particular, the implementation of m -HWBP requires $O(m^2)$ gates, whereas the implementation of Maj_m requires $O(m)$ gates. Since implementation efficiency is a primary concern, we choose the simpler option and compensate by increasing the number of variables.

n	$\text{AI}(\text{Maj}_{\lceil n/2 \rceil})$	$\text{AI}(\text{HWB}_{\lceil n/2 \rceil})$	$\text{AI}(\text{MM}_n)$	$\text{AI}(\text{Bal}_n)$
4	1	1	2	2
5	1	1	2	2
6	2	2	3	3
7	2	2	3	3
8	2	2	3	3
9	2	2	3	4
10	3	2	4	4
11	3	2	4	4
12	3	3	5	5
13	3	3	5	5
14	4	3	5	5
15	4	3	6	6
16	4	4	5	5
17	4	4	6	6
18	5	4	7	7
19	5	4	7	7
20	5	4	7	7

Table 4: Algebraic immunities of MM and Bal , where π is chosen to be m -HWBP and h is chosen to be Maj_m .

n	FAA-profile of MM_n	$\text{FAI}(\text{MM}_n)$	FAA-profile of Bal_n	$\text{FAI}(\text{Bal}_n)$
4	(1,1)	3	(1,1)	3
5	(1,1)	3	(1,2)	4
6	(1,2), (2,2)	4	(1,2), (2,2)	4
7	(1,2), (2,2)	4	(1,2), (2,2)	4
8	(1,3), (2,2)	5	(1,3), (2,2)	5
9	(1,3), (2,2)	5	(1,3), (2,3), (3,3)	5
10	(1,4), (2,3), (3,3)	6	(1,4), (2,3), (3,3)	6
11	(1,4), (2,3), (3,3)	6	(1,4), (2,4), (3,3)	6
12	(1,4), (2,4), (3,4), (4,4)	6	(1,5), (2,4), (3,4), (4,4)	7
13	(1,4), (2,4), (3,4), (4,4)	6	(1,5), (2,4), (3,4), (4,4)	7
14	(1,6), (2,5), (3,5), (4,5)	8	(1,6), (2,6), (3,5), (4,5)	8
15	(1,6), (2,5), (3,5), (4,5), (5,5)	8	(1,6), (2,5), (3,5), (4,5), (5,5)	8
16	(1,7), (2,6), (3,5), (4,5)	9	(1,7), (2,6), (3,5), (4,5)	9
17	(1,7), (2,6), (3,5), (4,5), (5,5)	9	(1,7), (2,6), (3,5), (4,5), (5,5)	9
18	(1,8), (2,7), (3,7), (4,6)	–	(1,8), (2,7), (3,7), (4,6)	–
19	(1,8), (2,7), (3,7)	–	(1,8), (2,7), (3,7)	–
20	(1,8), (2,8), (3,8)	–	(1,9), (2,8), (3,8)	–

Table 5: (Partial) FAA-profiles and FAI of MM and Bal , where π is chosen to be m -HWBP and h is chosen to be Maj_m .