Constructions of Efficiently Implementable Boolean functions Possessing High Nonlinearity and Good Resistance to Algebraic Attacks

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

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

August 21, 2024

Abstract

We describe two new classes of functions which provide the presently best known trade-offs between low computational complexity, nonlinearity and (fast) algebraic immunity. The nonlinearity and (fast) algebraic immunity of the new functions substantially improve upon those properties of all previously known efficiently implementable functions. Appropriately chosen functions from the two new classes provide excellent solutions to the problem of designing filtering functions for use in the nonlinear filter model of stream ciphers, or in any other stream ciphers using Boolean functions for ensuring confusion. In particular, for $n \leq 20$, we show that there are functions in our first family whose implementation efficiences are significantly lower than all previously known functions achieving a comparable combination of nonlinearity and (fast) algebraic immunity. Given positive integers ℓ and δ, it is possible to choose a function from our second family whose linear bias is provably at most $2^{-\ell}$, fast algebraic immunity is at least δ (based on conjecture which is well supported by experimental results), and which can be implemented in time and space which is linear in ℓ and δ. Further, the functions in our second family are built using homomorphic friendly operations, making these functions well suited for the application of transciphering.
Keywords: Boolean function, stream cipher, nonlinearity, algebraic immunity, efficient implementation.

1 Introduction

Many cryptosystems, such as stream ciphers, use Boolean functions for providing what C. Shannon called confusion in [32]. Concretely, confusion has been specified in the nineties into a series of cryptographic criteria (see *e.g.* [6]).

The nonlinear filter model is a several decades old model for stream ciphers. This model consists of two components, namely a state machine which maintains and updates a state, and a filtering function which is a Boolean function that is applied to a subset of the bits of the state. The state machine

*Corresponding author.

typically updates the state using a linear feedback function, while the filtering function is chosen to be a nonlinear function. The sequence of outputs of the filtering function on the successive states of the state machine constitutes the keystream produced by the stream cipher. A basic requirement on the Boolean function is that it is balanced so that there is no statistical bias in the keystream produced by the stream cipher and so between the plain-text and the cipher-text.

Extensive research has shown several approaches to cryptanalysing the filter model of stream ciphers. The two main approaches are fast correlation attacks and algebraic attacks of various types. Necessary properties of the Boolean filtering function have been identified for thwarting such attacks. These properties are a high nonlinearity and a high algebraic resistance. While these are security properties, from the point of view of implementation, a Boolean function is required to be efficiently implementable for the resulting stream cipher to be useful for real life applications, which requires it to be faster than any block cipher in counter mode for example. (Depending upon the application, the implementation may be required in hardware or software.)

The design challenge for a Boolean function to be used in the filter model of stream ciphers is the following. Construct a large family (if possible an infinite one) of Boolean functions all of which are balanced and achieve a good combination of high nonlinearity and high algebraic resistance and further are efficient to implement. In [7], this design challenge was referred to as "the big single-output Boolean problem" (similarly, in the domain of Boolean functions for stream ciphers, to the "big APN problem" in the domain of vectorial functions).

There are several known constructions of families of Boolean functions which achieve some, but not all of the above properties. We discuss these families in details in Section 3. For the present, we briefly mention some of these families. The Carlet-Feng functions [9] are balanced, achieve optimal algebraic immunity (and also almost optimal fast algebraic immunity) and high nonlinearity, but are not efficient to implement. The hidden weight bit (HWB) function [4] is very efficient to implement and in [33] it was shown that the HWB function has good algebraic immunity, but the nonlinearity is too low. Subsequently, a sequence of works [34, 7, 25] have generalised the HWB function to improve the nonlinearity while retaining the properties of good algebraic immunity and being efficient to implement. The trade-offs achieved by these works are not completely satisfactory.

In this paper, we revisit the above mentioned design problem for Boolean functions. We describe two new families of functions as solutions to the problem. Functions from both the families are very efficient to implement and achieve a good combination of high nonlinearity and high algebraic immunity. Below we provide a top-level overview of the two families.

Our first family of functions builds on the HWB function. To improve the nonlinearity, we introduce post-processing and pre-processing steps. For the post-processing step, we first extend the HWB function to a vectorial function by extracting a few bits and then apply a highly nonlinear function to these bits. The number of extracted bits is small (in fact, a constant) and so it is feasible to apply a highly nonlinear function to these bits without affecting the efficiency of implementation. For the pre-processing step, we design a novel bijection from n-bit strings to n-bit strings. The bijection is constructed by a combination of changing the representation from $\mathbb{F}_2^n$ to $\mathbb{Z}_{2^n}$ (which is directly implemented in computers and is then very fast), partitioning $\mathbb{Z}_{2^n}$ into a number of intervals, and using simple arithmetic operations over $\mathbb{Z}_{2^n}$. The net effect of applying both the pre and post processing steps is a significant improvement of both nonlinearity and algebraic resistance over HWB without compromising on the issue of efficient implementation. Our experimental results show that for all $n \leq 20$, both nonlinearity and algebraic resistance of suitably chosen n-variable functions from the new family are better than the corresponding values of n-variable functions from all previously known families [34, 7, 25] that are efficient to implement.

Our second family builds on the well known Maiorana-McFarland class of bent functions which 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 be $\langle \pi(\mathbf{X}), \mathbf{Y} \rangle \oplus h(\mathbf{X})$, where π is a bijection from m-bit strings to m-bit strings and h is any m-variable Boolean function. For odd n, we use Maiorana-McFarland bent functions to instantiate a previously proposed [31] construction of highly nonlinear balanced functions with optimal algebraic degree. For even $n = 2m$, we modify a construction due to Dobbertin [15] to construct highly nonlinear balanced function with optimal algebraic degree. For both odd and even n, we obtain provable assurance of high nonlinearity. In fact, the nonlinearity is substantially higher than the nonlinearity of the Carlet-Feng functions. Our main novelty is in the choice of h and π. It is well-known that the bentness of Maiorana-McFarland functions (defined as above) does not depend on the choice of h nor on that of permutation π. Similarly, the nonlinearities of functions that we construct do not depend either on the choices of h and π. We show that the resistance to algebraic attacks does depend on the choice of h and π. So we put forward the suggestion that h be chosen as the majority function since it achieves maximum algebraic immunity [13]. For the choice of π we show how to build a simple bijection based on the HWB function. This combination of majority and the HWB function results in good a (fast) algebraic immunity of the constructed functions. Based on experimental results, we conjecture that for $n \geq 6$, the algebraic immunity of an n-variable function in our second family is at least $\lfloor n/3 \rfloor$ and the fast algebraic immunity is equal to $1 + \lfloor n/2 \rfloor$. We show that given a pair of positive integers (ℓ, δ), it is possible to choose a function such that the linear bias is provably at most $2^{-\ell}$ and the fast algebraic immunity is (conjectured to be) at least δ. Further, the function can be computed very efficiently in time linear in ℓ and δ. This provides both an excellent theoretical as well as a practical solution to the design problem for Boolean functions to be used in the filter model of stream ciphers, or in other stream ciphers using Boolen functions to ensure confusion. Further, the functions in our second family are built using homomorphic friendly operations and the resulting stream ciphers can be used for transciphering [28].

The paper is organised as follows. In Section 2 we describe the preliminaries. The relevant previous constructions are discussed in Section 3. The post-processing step of the first family of functions is described in Section 4, while the pre-processing step and the family itself are described in Section 5. The second family of functions is described in Section 6. Finally, Section 7 concludes the paper.

Remark 1 *We report a number of experimental results. We used simple (and non-optimised) C code to construct the functions and compute their nonlinearities and algebraic degrees. For computing algebraic immunity we used the Boolean function library*[1] *of the SageMath software. For computing fast algebraic immunity, we used a program written by Simon Fischer which was kindly provided to us by Deng Tang.*

2 Preliminaries

In this section, we introduce the notation and provide the definitions of the properties of Boolean functions that we consider in this work. For further details and more elaborate discussion on these issues we refer to [6].

The cardinality of a finite set S will be denoted by $\#S$. For a prime power q, $\mathbb{F}_q$ denotes the finite field of order q consisting of q elements. In particular, $\mathbb{F}_2$ denotes the finite field of two elements. For a positive integer n, $\mathbb{F}_2^n$ is the vector space of dimension n over $\mathbb{F}_2$. The addition operation over both $\mathbb{F}_2$ and $\mathbb{F}_2^n$ will be denoted by $\oplus$. Elements of $\mathbb{F}_2^n$ are considered to be n-bit binary strings.

[1] `https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/boolean_function.html#sage.crypto.boolean_function.BooleanFunction.annihilator`

For an n-bit binary string $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathsf{wt}(\mathbf{x}) = \#\{i : x_i = 1\}$. Given two strings $\mathbf{x}$ and $\mathbf{y}$ of the same length, the distance between them, denoted $d(\mathbf{x}, \mathbf{y})$, is defined to be the number of places where $\mathbf{x}$ and $\mathbf{y}$ are unequal. Given $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_2^n$, their inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ is defined to be $\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 \oplus \cdots \oplus x_n y_n$. For an n-bit string $\mathbf{x}$, by $\mathsf{int}(\mathbf{x})$ we denote the unique integer $i \in \{0, \ldots, 2^n - 1\}$ whose n-bit binary representation is $\mathbf{x}$. Conversely, for $0 \leq i \leq 2^n - 1$, by $\mathsf{bin}_n(i)$ we denote the binary string given by the n-bit binary representation of i. The n-bit all-zero and all-one strings will be denoted as $\mathbf{0}_n$ and $\mathbf{1}_n$ respectively. For $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_2^n$, we say $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for $i = 1, \ldots, n$.

An n-variable Boolean function f is a map $f : \mathbb{F}_2^n \to \mathbb{F}_2$. By $\mathsf{supp}(f)$ we denote the set $\{\mathbf{x} \in \mathbb{F}_2^n : f(\mathbf{x}) = 1\}$. The *weight* of f, denoted $\mathsf{wt}(f)$, is the size of $\mathsf{supp}(f)$, i.e. $\mathsf{wt}(f) = \#\mathsf{supp}(f)$. An n-variable function f is said to be *balanced* if $\mathsf{wt}(f) = 2^{n-1}$. An n-variable function f is uniquely represented by a binary string $f_0 \cdots f_{2^n-1}$, where for $i \in \{0, \ldots, 2^n - 1\}$, $f_i = f(\mathsf{bin}_n(i))$. Such a string representation of f is also called the *truth table representation* of f.

An n-variable function f can be written as a multivariate polynomial in $\mathbb{F}_2[X_1, \ldots, X_n]/(X_1^2 \oplus X_1, \ldots, X_n^2 \oplus X_n)$ as follows. Let $\mathbf{X} = (X_1, \ldots, X_n)$. Then

$$f(X_1, \ldots, X_n) \quad = \quad \bigoplus_{\boldsymbol{\alpha} \in \mathbb{F}_2^n} a_{\boldsymbol{\alpha}} \mathbf{X}^{\boldsymbol{\alpha}}, \tag{1}$$

where $a_{\boldsymbol{\alpha}} \in \mathbb{F}_2$, and for $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, $\mathbf{X}^{\boldsymbol{\alpha}} = X_1^{\alpha_1} \cdots X_n^{\alpha_n}$. The representation given by (1) is called the *algebraic normal form (ANF) representation* of f. The algebraic degree (or simply the degree) of f is defined to be $\mathsf{deg}(f) = \max\{\mathsf{wt}(\boldsymbol{\alpha}) : a_{\boldsymbol{\alpha}} = 1\}$. Functions of degree at most 1 are said to be affine functions. Affine functions having $a_{\mathbf{0}_n} = 0$ are said to be linear functions. It is known that if f is balanced, then $\mathsf{deg}(f) \leq n - 1$. A balanced function f with $\mathsf{deg}(f) = n - 1$ is said to have optimal degree.

The following equations relate the coefficients $a_{\boldsymbol{\alpha}}$ in the ANF of f to the truth table representation of f (see for example Pages 49 and 50 of [6]). For $\mathbf{x}, \boldsymbol{\alpha} \in \mathbb{F}_2^n$,

$$f(\mathbf{x}) = \bigoplus_{\boldsymbol{\beta} \leq \mathbf{x}} a_{\boldsymbol{\beta}} \quad \text{and} \quad a_{\boldsymbol{\alpha}} = \bigoplus_{\mathbf{z} \leq \boldsymbol{\alpha}} f(\mathbf{z}). \tag{2}$$

Nonlinearity. For two n-variable functions f and g, the distance between them is denoted by $d(f, g)$ and is defined to be the distance between their truth table representations. The *nonlinearity* of an n-variable function f is denoted by $\mathsf{nl}(f)$ and is defined to be $\mathsf{nl}(f) = \min d(f, g)$, where the minimum is over all n-variable affine functions g.

The Walsh transform of an n-variable function f is a map $W_f : \mathbb{F}_2^n \to \mathbb{Z}$, where for $\boldsymbol{\alpha} \in \mathbb{F}_2^n$,

$$W_f(\boldsymbol{\alpha}) \quad = \quad \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) \oplus \langle \boldsymbol{\alpha}, \mathbf{x} \rangle}.$$

The function f is balanced if and only if $W_f(\mathbf{0}_n) = 0$. The nonlinearity of a function f is given by its Walsh transform as follows.

$$\mathsf{nl}(f) \quad = \quad 2^{n-1} - \frac{1}{2} \max_{\boldsymbol{\alpha} \in \mathbb{F}_2^n} |W_f(\boldsymbol{\alpha})|.$$

A function f such that $W_f(\boldsymbol{\alpha}) = \pm 2^{n/2}$ for all $\boldsymbol{\alpha} \in \mathbb{F}_2^n$ is said to be a bent function [30]. 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.

By $\mathsf{LLB}(f)$ we will denote the logarithm (to base two) of the linear bias of the function f. We would like to underline that we work with the logarithm of the linear bias rather than the linear bias itself. This is because for cryptographic applications, the linear bias is likely to be a small number and for small numbers it is more convenient to work with their logarithms than the numbers themselves.

$$\mathsf{LLB}(f) \quad = \quad \log_2\left(\frac{1}{2} - \frac{\mathsf{nl}(f)}{2^n}\right). \tag{3}$$

For a positive integer n, the covering radius bound CR_n is defined as follows.

$$\mathsf{CR}_n \quad = \quad 2^{n-1} - \lfloor 2^{n/2-1} \rfloor. \tag{4}$$

For an n-variable function f, we have $\mathsf{nl}(f) \leq \mathsf{CR}_n$, where equality holds for bent functions. By LCRB_n we will denote the following quantity.

$$\mathsf{LCRB}_n \quad = \quad \log_2\left(\frac{1}{2} - \frac{\mathsf{CR}_n}{2}\right). \tag{5}$$

Algebraic resistance. The *algebraic immunity* of a function f, denoted by $\mathsf{AI}(f)$, is defined in the following manner [12, 27].

$$\mathsf{AI}(f) \quad = \quad \min_{g \neq 0}\{\mathsf{deg}(g) : \text{ either } gf = 0, \text{ or } g(f \oplus 1) = 0\}. \tag{6}$$

For an n-variable function f, it is known [12] that $\mathsf{AI}(f) \leq \lceil n/2 \rceil$. So a function f has optimal AI if $\mathsf{AI}(f) = \lceil n/2 \rceil$. It was proved in [14] that a random n-variable function almost surely has AI at least $\lfloor n/2 - \log n \rfloor$.

Algebraic immunity quantifies the resistance of a function to algebraic attacks. In practice, it is also required to provide resistance to fast algebraic attack (FAA) [11]. Given an n-variable function f, let g be an n-variable function of degree e such that gf has degree d. If for small e, d is not too high then the function f is susceptible to an FAA. It is known [11] that for $e + d \geq n$, there exists functions g and h with $\mathsf{deg}(g) = e$ and $\mathsf{deg}(h) \leq d$ such that $gf = h$. Based on this observation, we provide the following definition. For each $e \in \{1, \ldots, \mathsf{AI}(f) - 1\}$, let $d \leq n - 1 - e$ be the maximum integer such that there do not exist n-variable functions g and h with $\mathsf{deg}(g) = e$, $\mathsf{deg}(h) = d$ and $gf = h$. We call the list of all such pairs (e, d) as the *FAA-profile* of f.

A combined measure of resistance offered by a function f to both algebraic and fast algebraic attacks is captured by the following notion called *fast algebraic immunity (FAI)*.

$$\mathsf{FAI}(f) \quad = \quad \min\left(2\mathsf{AI}(f), \min_{g \neq 0}\{\mathsf{deg}(g) + \mathsf{deg}(fg) : 1 \leq \mathsf{deg}(g) < \mathsf{AI}(f)\}\right). \tag{7}$$

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

If $\mathsf{AI}(f) = \lceil n/2 \rceil$ and for each pair (e, d) in the FAA-profile of f, $e + d = n - 1$, then f is said to have perfect algebraic immunity (PAI) [23]. We introduce a relaxed version of the notion of optimal AI and PAI. We say that a function f has almost optimal AI if $\mathsf{AI}(f) \geq \lfloor n/2 \rfloor$ and f is said to have almost perfect FAA-profile if for each pair (e, d) in the FAA-profile of f, $e + d \geq n - 2$.

Implementation efficiency. The complexity of implementing a Boolean function is measured with respect to space and time. For example, a truth table representation of an n-variable Boolean function requires 2^n bits and can be computed at a single point in $O(1)$ time (assuming that a look-up into the truth table requires constant time which need not be true if n is large). More generally, we say that a Boolean function has an (S,T)-implementation if it can be implemented using S bits/gates and can be computed using T bit operations. In an asymptotic sense, we say that an infinite family of Boolean functions has an efficient implementation if any n-variable function in the family has an (S,T)-implementation where both S and T are bounded above by polynomials in n. From a concrete point of view, on the other hand, we will be interested in further details of the implementation.

Vectorial functions. For positive integers n and m, an (n,m)-vectorial Boolean function (also called an S-box) F is a map $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. If $m = 1$, then we get back a Boolean function. An (n,m)-vectorial Boolean function F can be written as $F = (f_1, \ldots, f_m)$, where each f_i, $i = 1, \ldots, m$, is an n-variable Boolean function. The f_i's are said to be the coordinate functions of F. For $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \mathbb{F}_2^m$, let $F_{\boldsymbol{\alpha}} = \langle \boldsymbol{\alpha}, (f_1, \ldots, f_m) \rangle = \alpha_1 f_1 \oplus \cdots \oplus \alpha_m f_m$. Then $F_{\boldsymbol{\alpha}}$ is an n-variable Boolean function, and the $F_{\boldsymbol{\alpha}}$'s are called the component functions of F. For $n \geq m$, an (n,m)-vectorial function F is said to be balanced if for each $\boldsymbol{\beta} \in \mathbb{F}_2^m$, $\#F^{-1}(\boldsymbol{\beta}) = 2^{n-m}$. Equivalently, it is known that (see e.g. [6]) F is balanced if and only if all non-zero component functions of F are balanced.

Let F be an (n,m)-vectorial Boolean function and g be an m-variable Boolean function. The composition $g \circ F$ is an n-variable Boolean function given by $(g \circ F)(X_1, \ldots, X_n) = g(F(X_1, \ldots, X_n)) = g(f_1, \ldots, f_m)$. The Walsh transform of $f \circ F$ is the following [18]. For $\boldsymbol{\beta} \in \mathbb{F}_2^n$,

$$W_{f \circ F}(\boldsymbol{\beta}) \quad = \quad \frac{1}{2^m} \sum_{\boldsymbol{\alpha} \in \mathbb{F}_2^m} W_f(\boldsymbol{\alpha}) W_{F_{\boldsymbol{\alpha}}}(\boldsymbol{\beta}). \tag{8}$$

The following simple result can be proved directly by counting pre-images and also follows from (8).

Proposition 1 *Let n and m be positive integers with $n \geq m$, and let F be a balanced (n,m)-vectorial function. Let f be an m-variable Boolean function. Then $f \circ F$ is balanced if and only if f is balanced.*

3 Relevant Previous Constructions

In this section, we briefly outline some previous relevant constructions.

Carlet-Feng (CF) functions. Any polynomial $a(x) = a_0 \oplus a_1 x \oplus \cdots + a_{n-1} x^{n-1} \in \mathbb{F}_2[x]$ is uniquely determined by the coefficient vector $\mathbf{a} = (a_{n-1}, \ldots, a_0) \in \mathbb{F}_2^n$. So the elements of $\mathbb{F}_2^n$ can be considered to be polynomials in $\mathbb{F}_2[x]$ of degree at most $n-1$. Let $\tau(x)$ be a primitive polynomial of degree n over $\mathbb{F}_2$. An n-variable CF-function is defined by its support which is the following set of polynomials of degrees at most $n-1$:

$$\{0, 1, x \bmod \tau(x), x^2 \bmod \tau(x), \ldots, x^{2^{n-1}-2} \bmod \tau(x)\}.$$

It was shown in [9] that such a Boolean function is balanced, has degree $n-1$ and AI $\lceil n/2 \rceil$. (This class of functions was earlier considered in [16] for showing the tightness of bounds on the algebraic immunity of vectorial functions and the nonlinearity was earlier studied in [3].) Further, it was shown in [23] that when n is one more than a power of two, such functions possess PAI. A lower bound on the nonlinearity of such functions was proved in [9]. For concrete values of n, the actual nonlinearities are

much higher than the lower bound. Further, the nonlinearity depends on the choice of the primitive polynomial $\tau(x)$. We computed the nonlinearities of CF functions for certain values of n. The primitive polynomials that we used are given in Appendix A.

A drawback of the CF functions is that these are not very efficient to implement. Evaluating the value of a CF function on a particular input $a(x)$ amounts to computing i such that $a(x) \equiv x^i \bmod p(x)$. This is the discrete logarithm problem in $\mathbb{F}_{2^n}$. A truth table implementation of CF-functions requires $O(2^n)$ bits. Using polynomial space the discrete logarithm problem can be solved in asymptotically sub-exponential time. However, for concrete instances, the function is less slow to compute than it could seem: for particular choices of n, it may be possible to use the Pohlig-Hellman algorithm to obtain a faster algorithm. If $2^n - 1$ is the product of small factors (this is the case of $n = 18$ and $n = 20$ for instance), it is possible to compute one output bit per cycle with 40,000 transistors, as indicated in [5].

Hidden weight bit (HWB) functions. For $n \geq 1$, let $\mathsf{HWB}_n : \{0,1\}^n \to \{0,1\}$ be the hidden weight bit function [4] defined as follows. For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$,

$$\mathsf{HWB}_n(\mathbf{x}) \quad = \quad x_{\mathsf{wt}(\mathbf{x})}, \tag{9}$$

where we assume that $x_0 = 0$. The HWB functions are clearly efficiently implementable. Cryptographic properties of HWB functions were studied in [33]. It was shown that the AI of HWB_n is at least $\lfloor n/3 \rfloor + 1$ and for n in the set $\{6, \ldots, 13\}$, the actual AI is either the lower bound or one more than the lower bound. For n in the set $\{6, \ldots, 13\}$, the FAA-profiles were reported in [33] and turned out to be significantly away from the profile of a PAI function.

The nonlinearity of HWB_n was shown to be $2^{n-1} - 2\binom{n-2}{\lceil (n-2)/2 \rceil}$. This value is quite low. So even though HWB functions are efficiently implementable, they do not possess sufficiently high nonlinearity for cryptographic applications. Concatenations of HWB functions have been studied in [34] producing functions with higher nonlinearities than the HWB functions, but still not high enough for use in practical systems.

Binary decision diagrams (BDD) have been used to propose attacks on stream ciphers [21, 22]. A positive feature of HWB functions is that these functions have high BDD complexity [4, 1, 20].

Generalised HWB (GHWB) functions. A generalisation of HWB functions was introduced in [7] with the goal of improving their nonlinearity and algebraic immunity while retaining the efficiency of implementation. The concrete results for $n = 13, 14, 15$ and 16 presented in [7] show that the AI of GHWB is almost optimal and is greater than the AI of HWB. There is also improvement in nonlinearity. This improvement, however, is not substantial and the obtained nonlinearities of GHWB functions are still significantly lower than that of the CF functions.

Cyclic weightwise functions. Another generalisation of the HWB function was made in [25]. Let $g_0, \ldots, g_n$ be n-variable functions. Using these $n + 1$ functions, an n-variable weightwise function f is constructed as follows: for $\mathbf{x} \in \mathbb{F}_2^n$, $f(\mathbf{x}) = g_w(\mathbf{x})$, where $w = \mathsf{wt}(\mathbf{x})$. The function f is uniquely defined by the sequence of functions $(g_0, \ldots, g_n)$. Note that the function g_w is applied only to strings of weight w. In particular g_0 is applied only to the string $\mathbf{0}_n$.

Since implementing $n + 1$ functions may be difficult in practice, the notion of *cyclic weightwise* functions was introduced in [25], where the functions g_i's are defined from a single n-variable function g as follows: $g_0 = g_1 = g$, and for $i \in \{2, \ldots, n\}$, g_i is defined to be $g_i(\mathbf{x}) = g(\mathbf{x} \ggg (i-1))$, where $\ggg$ is the cyclic right shift operator. The resulting function f is called a cyclic weightwise function, which we denote as $f = \mathsf{CW}_n(g)$. Lower bounds on the nonlinearities of $\mathsf{CW}_n(g)$ was obtained

in [25] for the case when g is linear and for a particular quadratic function g. For the choice of $g(x_1, \ldots, x_n) = x_1 \oplus \left(\bigoplus_{i=1}^{\lfloor (n-1)/2 \rfloor} x_{2i}x_{2i+1}\right)$, actual nonlinearities, degrees and algebraic immunities of $\mathsf{CW}_n(g)$ were provided in [25]. These functions achieve both the highest nonlinearities and the highest algebraic immunities among all the functions presented in [25]. Later we compare these functions to the functions that are obtained from the new constructions that we propose.

Inverse map. Let $\rho(x) \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n. Then for any nonzero polynomial $a(x) \in \mathbb{F}_2[x]$ of degree at most $n-1$, there is a polynomial $b(x)$ also of degree at most $n-1$ such that $a(x)b(x) \equiv 1 \bmod \rho(x)$, i.e. $b(x) = a(x)^{-1} \bmod \rho(x)$. As in the case of the CF functions, we identify polynomials in $\mathbb{F}_2[x]$ of degrees at most $n-1$ with the elements of $\mathbb{F}_2^n$. We can then define an (n,n)-vectorial function $\mathsf{inv} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ as follows: $\mathsf{inv}(\mathbf{0}_n) = \mathbf{0}_n$ and for any $a(x) \in \mathbb{F}_2[x]$ of degree at most $n-1$, $\mathsf{inv}(a(x)) = a(x)^{-1} \bmod \rho(x)$. This is the well known inverse map which was introduced to cryptography in [29]. A nonzero component function of inv is an n-variable Boolean function. Such functions are balanced and have degrees equal to $n-1$. Further, it is known [29, 10] that the nonlinearity of any non-zero component function is at least $2^{n-1} - 2^{n/2}$. The AI of such a function, however, is not good. It was shown in [17], that the AI is equal to $\lceil 2\sqrt{n} \rceil - 2$. From an implementation point of view, computing $a(x)^{-1} \bmod \rho(x)$ requires about $O(n^3)$ bit operations.

4 Construction of λ-HWB Functions

The HWB function is efficient to implement. Its major drawback, however, is its low nonlinearity. One possible way to improve the cryptographic properties of the HWB function is to perform some post-processing of its output. Recall that the HWB function produces a single bit of output. It is not meaningful to perform any post-processing on a single bit. So as a first step, we consider a vectorial version of the HWB function which produces more than one bit of output. Let r be the number of bits that are to be produced. The question then is how should these r bits be extracted. On an input $\mathbf{x} = (x_1, \ldots, x_n)$, the HWB function produces as output x_i, where i is the weight of $(x_1, \ldots, x_n)$. To extract r bits, we extract a window of r bits of $\mathbf{x}$ centered at x_i. This creates a difficulty if indices of the window fall outside the range $\{1, \ldots, n\}$. There are two ways to tackle this situation, namely the null and the cyclic boundary conditions which we define as follows. Let $\mathbf{x} = (x_1, \ldots, x_n)$ and suppose i is an integer which is not in $\{1, \ldots, n\}$. Under the *null boundary condition*, we define x_i to be 0, while under the *cyclic boundary condition*, we define x_i to be equal to x_j, where j is the unique integer in $\{1, \ldots, n\}$ such that $i \equiv j \bmod n$. From experimental results we find that the nonlinearities of the functions obtained using the cyclic boundary condition is more than the nonlinearities of the functions obtained using the null boundary condition (see Remark 4 later). In view of this, we do not formally introduce the construction using the null boundary condition.

Given positive integers n and r with $r \leq n$, we define an (n,r)-vectorial function $\mathsf{HWB}_{n,r}$ as follows. For $\mathbf{x} \in \mathbb{F}_2^n$, let $w = \mathsf{wt}(\mathsf{x})$. Let $\ell = w - \lfloor r/2 \rfloor$ if r is odd and let $\ell = w - r/2 + 1$ if r is even. Then

$$\mathsf{HWB}_{n,r} \quad = \quad (x_\ell, x_{\ell+1}, \ldots, x_{\ell+r-1}) \quad \text{with cyclic boundary condition.} \tag{10}$$

Note that $\mathsf{HWB}_{n,1} = \mathsf{HWB}_n$. We have the following result regarding the balancedness of $\mathsf{HWB}_{n,r}$.

Proposition 2 *Let n and r be positive integers with $1 \leq r \leq n$. Then $\mathsf{HWB}_{n,r}$ is balanced.*

Proof: Let $\boldsymbol{\beta} \in \mathbb{F}_2^r$. We count the number of preimages of $\boldsymbol{\beta}$ under $\mathsf{HWB}_{n,r}$. For $\mathbf{x} \in \mathbb{F}_2^n$ with $w = \mathsf{wt}(\mathbf{x})$, suppose $\mathsf{HWB}_{n,r}(\mathbf{x}) = \boldsymbol{\beta}$. Then $(x_\ell, x_{\ell+1}, \ldots, x_{\ell+r-1}) = \boldsymbol{\beta}$, where $\ell = w - \lfloor r/2 \rfloor$ if r is odd

and let $\ell = w - r/2 + 1$ if r is even. Let $k = \mathsf{wt}(\boldsymbol{\beta})$. Then $\#\{i \in \{1,\ldots,n\} \setminus \{\ell,\ldots,\ell+r-1\} : x_i = 1\} = w - k$. So the number of $\mathbf{x}$'s such that $\mathsf{wt}(\mathbf{x}) = w$ and $(x_\ell, x_{\ell+1}, \ldots, x_{\ell+r-1}) = \boldsymbol{\beta}$ is equal to $\binom{n-r}{w-k}$. Consequently, the number of preimages of $\boldsymbol{\beta}$ under $\mathsf{HWB}_{n,r}$ is $\sum_{w=0}^{n}\binom{n-r}{w-k} = 2^{n-r}$, since $n - k \geq n - r$. $\square$

Let λ be an r-variable Boolean function. We define an n-variable Boolean function λ-$\mathsf{HWB}_{n,r}$ in the following manner.

$$\lambda\text{-}\mathsf{HWB}_{n,r} \quad = \quad \lambda \circ \mathsf{HWB}_{n,r}. \tag{11}$$

So for $\mathbf{x} \in \mathbb{F}_2^n$, λ-$\mathsf{HWB}_{n,r}(\mathbf{x}) = \lambda(\mathsf{HWB}_{n,r}(\mathbf{x}))$.

Remark 2 *Let $\ell_1 = 1 - \lfloor r/2 \rfloor$ if r is odd and let $\ell_1 = 1 - r/2 + 1$ if r is even. Define an n-variable function g, where for $(x_1,\ldots,x_n) \in \mathbb{F}_2^n$, $g(x_1,\ldots,x_n) = \lambda(x_\ell, x_{\ell+1},\ldots,x_{\ell+r-1})$ with cyclic boundary condition. Let $g_0, g_1, \ldots, g_n$ be n-variable functions where $g_0 = g_1 = g$ and for $i \in \{2,\ldots,n\}$, $g_i(x_1,\ldots,x_n) = g((x_1,\ldots,x_n) \lll (i-1))$, where $\lll$ is the cyclic left shift operator. Then λ-$\mathsf{HWB}_{n,r}$ is a weightwise function defined by the sequence of functions $(g_0, g_1, \ldots, g_n)$. Note that the notion of cyclic weightwise functions is defined using right cyclic shifts, whereas λ-$\mathsf{HWB}_{n,r}$ is obtained from g using left cyclic shifts*[2].

Proposition 3 *Let λ be an r-variable Boolean function. Then λ-$\mathsf{HWB}_{n,r}$ is balanced if and only if λ is balanced.*

Proof: Proposition 2 shows that $\mathsf{HWB}_{n,r}$ is a balanced (n,r)-vectorial function. From Proposition 1 we have that the composition of a balanced (n,r)-vectorial function and an r-variable Boolean function λ is balanced if and only if λ is balanced. $\square$

Let $\pi_1,\ldots,\pi_n$ be permutations of $\{1,\ldots,n\}$ and for $i = 1,\ldots,n$, let $P_i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be defined as $P_i(x_1,\ldots,x_n) = (x_{\pi(1)},\ldots,x_{\pi(n)})$. Let g be an n-variable Boolean function and f be another n-variable Boolean function defined using g and $P_1,\ldots,P_n$ in the following manner: $f(\mathbf{x}) = g(P_w(\mathbf{x}))$, where $w = \mathsf{wt}(\mathbf{x})$. Proposition 4 of [25] shows that f is balanced if and only if g is balanced. Proposition 3 can be seen as a corollary of Proposition 4 of [25]. On the other hand, Proposition 4 of [25] itself can be seen as a corollary of Proposition 1 in the following manner. Given $P_1,\ldots,P_n$, define a bijection $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ by $S(\mathbf{x}) = P_w(\mathbf{x})$, where $w = \mathsf{wt}(\mathbf{x})$. Then $f = g \circ S$, and by Proposition 1, f is balanced if and only if g is balanced.

Proposition 4 *For any r-variable function λ, $(1 \oplus \lambda)$-$\mathsf{HWB}_{n,r} = 1 \oplus \lambda$-$\mathsf{HWB}_{n,r}$. More generally, for any invertible affine transformation $A : \mathbb{F}_2^r \to \mathbb{F}_2^r$, $\mathsf{nl}(\lambda \circ A \circ \mathsf{HWB}_{n,r}) = \mathsf{nl}(\lambda \circ \mathsf{HWB}_{n,r})$.*

The nonlinearity of λ-$\mathsf{HWB}_{n,r}$ is determined by the Walsh transform of λ-$\mathsf{HWB}_{n,r}$. In principle, using (8), the Walsh transform of λ-$\mathsf{HWB}_{n,r}$ can be determined from the Walsh transforms of λ and $\mathsf{HWB}_{n,r}$. So in principle, using (8), the nonlinearity of λ-$\mathsf{HWB}_{n,r}$ can be determined from the Walsh transforms of λ and $\mathsf{HWB}_{n,r}$. The form of (8), however, does not provide any easy method to identify conditions on the Walsh transform of λ such that the nonlinearity of λ-$\mathsf{HWB}_{n,r}$ is high.

Remark 3 *Lower bounds on the nonlinearties of certain cyclic weightwise functions have been obtained in [25]. These lower bounds also provide lower bounds on the nonlinearities of λ-$\mathsf{HWB}_{n,r}$ in the case where λ is a linear function, or when λ is a particular quadratic function. As mentioned in [25], the lower bounds obtained in [25] are not tight and the actual nonlinearities are higher.*

[2]We were unaware of the paper [25] when we obtained the function λ-$\mathsf{HWB}_{n,r}$. It is only later that we realised that λ-$\mathsf{HWB}_{n,r}$ is a special case of (left) cyclic weightwise functions.

Our goal is to choose λ such that λ-$\mathsf{HWB}_{n,r}$ has high nonlinearity. As discussed above, the expression for the Walsh transform of λ-$\mathsf{HWB}_{n,r}$ given by (8) does not provide any guidance. Further, as discussed in Remark 3 the analysis in [25] provides loose lower bounds for some very special choices of λ. Faced with this scenario, we decided to search for choices of λ to determine the set of λ's having the highest possible nonlinearity. Since we are interested in balanced functions, using Proposition 3, we focused only on balanced λ's. Algorithm 1 describes our search strategy. It takes as input n, r and a list $\mathcal{S}$ of r-variable balanced functions and produces as output a set of functions λ such that the corresponding λ-$\mathsf{HWB}_{n,r}$ function has algebraic degree $n-1$ and as such, has maximal nonlinearity among all visited functions.

Algorithm 1: The search procedure for λ-$\mathsf{HWB}_{n,r}$.

Input: n, r and $\mathcal{S}$, where $\mathcal{S}$ is a subset of the set of all balanced r-variable functions

Output: A list $\mathcal{L}$ of r-variable functions such that for any $\lambda \in \mathcal{L}$, λ-$\mathsf{HWB}_{n,r}$ is balanced, has degree $n-1$ and $\lambda \in \mathrm{argmax}_{\mu \in \mathcal{S}}\, \mathsf{nl}(\mu\text{-}\mathsf{HWB}_{n,r})$

```
1  maxnl ← 0; L ← ∅
2  for λ ∈ S do
3      let f = λ-HWB_{n,r}
4      compute nl(f) and deg(f)
5      if deg(f) = n − 1 and maxnl < nl(f) then
6          maxnl ← nl(f); L ← {λ}
7      else
8          if deg(f) = n − 1 and maxnl = nl(f) then
9              L ← L ∪ {λ}
10 return L
```

Proposition 5 *For positive integers n and r with $1 \leq r \leq n$ and $\mathcal{S}$ a subset of balanced r-variable functions, let $\mathcal{L}$ be returned by Algorithm 1 on input n, r and $\mathcal{S}$. Then for any $\lambda \in \mathcal{L}$, λ-$\mathsf{HWB}_{n,r}$ is a balanced n-variable function having degree $n-1$. The time taken by Algorithm 1 is $O(\#\mathcal{S}\, n2^n)$.*

Proof: Suppose $\mathcal{L}$ is the output of Algorithm 1. From Proposition 3 it follows that any $\lambda \in \mathcal{L}$ is balanced. From the algorithm, it directly follows that the degree is $n-1$.

For each λ in $\mathcal{S}$, the algorithm constructs the n-variable function λ-$\mathsf{HWB}_{n,r}$ and computes its nonlinearity and degree. So the time for each λ is $O(n2^n)$, and the total time is $O(\#\mathcal{S}\, n2^n)$. □

If $\mathcal{S}$ is the set of all balanced r-variable functions, then the time required by Algorithm 1 is $O(\binom{2^r}{2^{r-1}} n2^n)$.

For $r = 2, 3$ and 4, and for $n = 13, \ldots, 20$, we have run Algorithm 1 with $\mathcal{S}$ to be the set of all r-variable balanced Boolean functions. (Note that for $r = 2$ the only balanced functions are the non-constant affine functions.) A summary of our observations of these executions of Algorithm 1 are as follows.

1. For $n = 13, \ldots, 20$ and $r = 2$, for the λ's produced by Algorithm 1, the nonlinearities of λ-$\mathsf{HWB}_{n,2}$ are equal to the nonlinearities of the corresponding HWB_n. This though is not true in general. For example, for $n = 8$, taking $\lambda(X_1, X_2) = X_1 \oplus X_2$, the nonlinearity of λ-$\mathsf{HWB}_{8,2}$ is 92, while the nonlinearity of HWB_8 is 88.

2. For a fixed value of n, the nonlinearity of λ-$\mathsf{HWB}_{n,r}$ with λ produced by Algorithm 1 increases with the value of r.

For $r = 5$, the number of balanced r-variable functions is equal to $\binom{32}{16} \approx 2^{29.163}$. So if in Algorithm 1 we put $\mathcal{S}$ to be the set of all 5-variable balanced functions, then the time taken will be proportional to $n2^{n+29.163}$. On the computing resources available to use, for $n = 13$ this computation is barely feasible while it is out of our reach for $n = 20$. Accordingly, we decided to take $\mathcal{S}$ to be a proper subset of 5-variable balanced functions. The first condition that we imposed is to consider only functions having degree 4. This, however, does not significantly reduce the size of $\mathcal{S}$. Next we imposed the condition that along with degree 4, the functions should have nonlinearity 12, which is the maximum possible nonlinearity among all 5-variable balanced functions. This condition is motivated by our finding that for $r = 3$ and $r = 4$, the λ's which are returned by Algorithm 1 have the maximum possible nonlinearity among all balanced r-variable functions. (At present, however, this is only an observation from the experimental results, and we do not have a proof.) The number of 5-variable functions having degree 4 and nonlinearity 12 is $1666560 \approx 2^{20.668}$. With $\#\mathcal{S} = 1666560$, it becomes feasible to run Algorithm 1 for $n = 13, \ldots, 20$ on our computers. The nonlinearities that are obtained are higher than the nonlinearities obtained for $r = 2, 3$ and 4. The following proposition states the results that we obtained.

Proposition 6 *Let $r = 5$. For $n = 13, \ldots, 20$, the maximum nonlinearities, along with the corresponding λ's and $1 \oplus \lambda$'s, achieved by balanced λ-$\mathsf{HWB}_{n,r}$ functions having degree $n - 1$, where λ runs over all 5-variable balanced functions having degree 4 and nonlinearity 12, are as follows.*

- $n = 13$, $\mathsf{nl}(\lambda\text{-}\mathsf{HWB}_{n,r}) = 3780$, *where* $\lambda, 1 \oplus \lambda \in \{\lambda_{5,1}, \lambda_{5,2}\}$.
- $n = 14$, $\mathsf{nl}(\lambda\text{-}\mathsf{HWB}_{n,r}) = 7572$, *where* $\lambda, 1 \oplus \lambda \in \{\lambda_{5,3}, \lambda_{5,4}\}$.
- $n = 15$, $\mathsf{nl}(\lambda\text{-}\mathsf{HWB}_{n,r}) = 15236$, *where* $\lambda, 1 \oplus \lambda \in \{\lambda_{5,1}, \lambda_{5,2}\}$.
- $n = 16$, $\mathsf{nl}(\lambda\text{-}\mathsf{HWB}_{n,r}) = 30526$, *where* $\lambda, 1 \oplus \lambda \in \{\lambda_{5,5}, \lambda_{5,6}\}$.
- $n = 17$, $\mathsf{nl}(\lambda\text{-}\mathsf{HWB}_{n,r}) = 61284$, *where* $\lambda, 1 \oplus \lambda \in \{\lambda_{5,1}, \lambda_{5,2}\}$.
- $n = 18$, $\mathsf{nl}(\lambda\text{-}\mathsf{HWB}_{n,r}) = 122758$, *where* $\lambda, 1 \oplus \lambda \in \{\lambda_{5,7}, \lambda_{5,8}\}$.
- $n = 19$, $\mathsf{nl}(\lambda\text{-}\mathsf{HWB}_{n,r}) = 246368$, *where* $\lambda, 1 \oplus \lambda \in \{\lambda_{5,9}, \lambda_{5,10}\}$.
- $n = 20$, $\mathsf{nl}(\lambda\text{-}\mathsf{HWB}_{n,r}) = 493476$, *where* $\lambda, 1 \oplus \lambda \in \{\lambda_{5,11}, \lambda_{5,12}\}$.

In the above, $\lambda_{5,i}$, $i = 1, \ldots, 12$, given by their 32-bit string representations are the following. (The ANFs of these functions are given in Appendix B.)

$$\lambda_{5,1} = 10111111010100010001101000001110, \quad \lambda_{5,2} = 10101000011010110100111001001110$$
$$\lambda_{5,3} = 10010011011000111011010111010000, \quad \lambda_{5,4} = 10000100110100111010100111110100$$
$$\lambda_{5,5} = 10101011011010110001101100011000, \quad \lambda_{5,6} = 10000101111110111000101000001110$$
$$\lambda_{5,7} = 11100001010111110000101001001110, \quad \lambda_{5,8} = 10101011001100100001111000011110$$
$$\lambda_{5,9} = 10001001010111110010110011101000, \quad \lambda_{5,10} = 10101011100100111011010100000110$$
$$\lambda_{5,11} = 01100010101011111110000111000100, \quad \lambda_{5,12} = 01100000110010110101011101001110$$

Proof: We have run Algorithm 1 with $r = 5$ and $\mathcal{S}$ to be the set of all balanced r-variable functions having degree 4 and nonlinearity 12. The stated result lists the outputs of Algorithm 1 for $n = 13, \ldots, 20$. □

Remark 4 *For null boundary condition, to obtain a balanced function it is not required that λ be balanced. Let λ-$\mathsf{HWB}_{n,r}^{(\mathsf{n})}$ denote the function constructed using the null boundary condition. For $r = 3, 4$ and $n = 13, \ldots, 20$, we constructed all possible n-variable functions λ-$\mathsf{HWB}_{n,r}^{(\mathsf{n})}$ using the null boundary condition corresponding to all the 2^{2^r} possible r-variable functions λ, computed their Walsh transforms to determine whether they are balanced and obtained their nonlinearities. The best nonlinearities obtained by this procedure turned out to be less than the best nonlinearities obtained for λ-$\mathsf{HWB}_{n,r}$. For $r = 5$, we let λ vary over all 5-variable balanced functions having degree 4 and nonlinearity 12. For each such λ, we constructed the corresponding λ-$\mathsf{HWB}_{n,r}^{(\mathsf{n})}$ using the null boundary condition, computed its Walsh transform and determined its nonlinearity and whether it is balanced. Again for $r = 5$, this experiment resulted in functions whose nonlinearities are less than the functions obtained for $r = 5$ using cyclic boundary condition. Since for $r = 3, 4$ and 5, the best nonlinearities obtained using the null boundary condition are less than the best nonlinearities obtained using the cyclic boundary condition, we do not report the results of our experiments for the functions obtained using the null boundary condition.*

We computed the algebraic immunities of the functions given by Proposition 6. The results are given in the following proposition.

Proposition 7 *For $r = 5$ and $n = 13, \ldots, 18$, the algebraic immunities of the functions described in Proposition 6 are the following.*

1. $\mathsf{AI}(\lambda_{5,1}\text{-}\mathsf{HWB}_{13,5}) = \mathsf{AI}(\lambda_{5,2}\text{-}\mathsf{HWB}_{13,5}) = 6$.
2. $\mathsf{AI}(\lambda_{5,3}\text{-}\mathsf{HWB}_{14,5}) = \mathsf{AI}(\lambda_{5,4}\text{-}\mathsf{HWB}_{14,5}) = 7$.
3. $\mathsf{AI}(\lambda_{5,1}\text{-}\mathsf{HWB}_{15,5}) = \mathsf{AI}(\lambda_{5,2}\text{-}\mathsf{HWB}_{15,5}) = 7$.
4. $\mathsf{AI}(\lambda_{5,5}\text{-}\mathsf{HWB}_{16,5}) = \mathsf{AI}(\lambda_{5,6}\text{-}\mathsf{HWB}_{16,5}) = 7$.
5. $\mathsf{AI}(\lambda_{5,1}\text{-}\mathsf{HWB}_{17,5}) = \mathsf{AI}(\lambda_{5,2}\text{-}\mathsf{HWB}_{17,5}) = 8$.
6. $\mathsf{AI}(\lambda_{5,7}\text{-}\mathsf{HWB}_{18,5}) = \mathsf{AI}(\lambda_{5,8}\text{-}\mathsf{HWB}_{18,5}) = 8$.

We compare properties of λ-$\mathsf{HWB}_{n,r}$ with the properties of HWB, GHWB and the highest nonlinearities reported in [25]. For λ-$\mathsf{HWB}_{n,r}$ the highest nonlinearities are achieved for $r = 5$ and hence we do not report the nonlinearties achieved for $r = 2, 3$ and 4. The nonlinearities are compared in Table 1. In the table, for each function f, along with $\mathsf{nl}(f)$ we also provide the value of $\mathsf{LLB}(f)$. In Table 2, we compare the degrees and algebraic immunities of λ-$\mathsf{HWB}_{n,r}$ functions with those of HWB, GHWB and Table 7 of [25]. Each entry of Table 2 is of the form (d, a), where d is the degree and a is the algebraic immunity.

Based on Tables 1 and 2, we make the following observations.

1. The nonlinearities of the functions given by Proposition 6 are higher than those of GHWB, but lower than those reported in Table 7 of [25].
2. The algebraic immunities of the functions given by Proposition 6 are at least the optimum minus one.
3. The algebraic immunities of the functions given by Proposition 6 are never less than those of GHWB. Compared to Table 7 of [25], the algebraic immunities of the functions given by Proposition 6 are equal for $n = 13$ and $n = 16$, but higher for $n = 14$ and $n = 15$.

	HWB [33]		GHWB [7]		Table 7 of [25]		Proposition 6	
n	nl	LLB	nl	LLB	nl	LLB	nl	LLB
13	3172	−3.15	3284	−3.33	3862	−5.13	3780	−4.70
14	6344	−3.15	6668	−3.43	7816	−5.45	7572	−4.72
15	12952	−3.26	14604	−4.20	15748	−5.69	15236	−4.84
16	25904	−3.26	29128	−4.17	31616	−5.83	30526	−4.87
17	52666	−3.35	–	–	–	–	61284	−4.95
18	105332	−3.35	–	–	–	–	122758	−4.98
19	213524	−3.43	–	–	–	–	246368	−5.05
20	427048	−3.43	–	–	–	–	493476	−5.09

Table 1: Comparison of nonlinearities achieved by λ-$\mathsf{HWB}_{n,r}$ with the nonlinearities of HWB, GHWB and Table 7 of [25].

n	HWB	GHWB	Table 7 of [25]	Proposition 7
13	(12,5)	(12,6)	(12,6)	(12,6)
14	(13,5)	(13,6)	(12,6)	(13,7)
15	(14,6)	(14,7)	(14,6)	(14,7)
16	(15,6)	(15,7)	(14,7)	(15,7)
17	–	–	–	(16,8)
18	–	–	–	(17,8)

Table 2: Comparison of degrees and algebraic immunities of λ-$\mathsf{HWB}_{n,r}$ functions with those of HWB, GHWB and Table 7 of [25].

4. The degrees of the functions given by Proposition 6 are always optimal, while the degrees of the functions reported in Table 7 of [25] are optimal for $n = 13$ and $n = 15$ and one less than the optimal for $n = 14$ and $n = 16$.

To summarise, compared to the functions reported in Table 7 of [25], the functions given by Proposition 6 represent a different trade-off, i.e. while the nonlinearities are lower, the algebraic immunities and the degrees are the same or higher.

5 Construction of Interval λ-HWB Functions

The function λ-$\mathsf{HWB}_{n,r}$ improves the properties of the HWB function by first extending the HWB function to a vectorial function and then applying λ to the output of the vectorial function. This constitutes a post-processing of the output of the HWB vectorial function. The functions that are obtained using this approach provide a different trade-off from the functions reported in Table 7 of [25], while the nonlinearities are lower, the degrees and the algebraic immunities are the same or higher. To further improve the nonlinearity, we consider a pre-processing of the input to λ-$\mathsf{HWB}_{n,r}$. In more details, we construct a nonlinear bijection $\phi : \mathbb{F}_2^n \to \mathbb{F}_2^n$, so that before applying λ-$\mathsf{HWB}_{n,r}$ to an input $\mathbf{x} \in \mathbb{F}_2^n$, we first apply ϕ to $\mathbf{x}$ to obtain $\mathbf{y}$ and then apply λ-$\mathsf{HWB}_{n,r}$ to $\mathbf{y}$.

Let $\mathbb{Z}_{2^n}$ be the set of integers modulo 2^n. To improve the nonlinearity, we need to construct ϕ such that it is both fast and highly nonlinear. There exist maps which are very fast to compute since they are directly implemented in computers, and which allow to change the structure of $\mathbb{F}_{2^n}$ into that of the residue class ring $\mathbb{Z}_{2^n}$ and vice versa (conversions between the representations $\mathbb{F}_{2^n}$ and $\mathbb{Z}_{2^n}$ are done using the functions $\mathsf{int}(\mathbf{x})$ and $\mathsf{bin}_n(i)$, as described in Section 2.) The fact that each structure

is complex with respect to the other is used in the so-called ARX cryptosystems. We shall build our preprocessing on these functions.

The core of our construction of ϕ is based on the idea of partitioning $\mathbb{Z}_{2^n}$ into intervals. We first describe this partitioning strategy.

> *Partition of* $\mathbb{Z}_{2^n}$: Let $n \geq 2$ and $s < n$ be a positive integer. Let $0 \leq w_0, \ldots, w_{2^s-1} \leq 2^n - 1$ be integers such that $w_{k+1} = w_k + 2^{n-s} \bmod 2^n$. For $0 \leq k \leq 2^s - 1$, let $I_k = \{w_k, w_k + 1, \ldots, w_k + 2^{n-s} - 1\}$ where the elements of the set I_k are computed modulo 2^n.

Proposition 8 *The collection of sets* $\{I_k\}$ *with* $k = 0, \ldots, 2^s - 1$ *forms a partition of* $\mathbb{Z}_{2^n}$.

Proof: Note that the number of I_k's is 2^s, and each I_k is a subset of $\mathbb{Z}_{2^n}$ containing 2^{n-s} elements. So to show the result it is sufficient to show that for $0 \leq k < \ell \leq 2^s - 1$, I_k and I_ℓ are disjoint. From the definition of the w_k's, we have $w_\ell = w_k + (\ell - k)2^{n-s} \bmod 2^n$. Suppose that I_k and I_ℓ have a non-empty intersection. Then there are integers a and b with $0 \leq a, b \leq 2^{n-s} - 1$ such that $w_k + a \equiv w_\ell + b \bmod 2^n$, i.e. $(\ell - k)2^{n-s} + (b - a) \equiv 0 \bmod 2^n$. Note that $1 \leq \ell - k \leq 2^s - 1$ and so $2^{n-s} \leq (\ell - k)2^{n-s} \leq 2^n - 2^{n-s}$. Further, $-2^{n-s} + 1 \leq b - a \leq 2^{n-s} - 1$. So $1 \leq (\ell - k)2^{n-s} + (b - a) \leq 2^n - 1$. Consequently, $(\ell - k)2^{n-s} + (b - a) \not\equiv 0 \bmod 2^n$, which is a contradiction. □

Proposition 9 *For* $n \geq 2$, $w_0 \in \mathbb{Z}_{2^n}$ *and positive integer* $s < n$, *define* $\mathcal{I}_{n,w_0,s} : \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^s}$ *as follows.*

$$\mathcal{I}_{n,w_0,s}(i) = \begin{cases} \left\lfloor \frac{i - w_0}{2^{n-s}} \right\rfloor & \text{if } i \geq w_0, \\ \left\lfloor \frac{i + 2^n - w_0}{2^{n-s}} \right\rfloor & \text{if } i < w_0. \end{cases} \tag{12}$$

Let $k = \mathcal{I}_{n,w_0,s}(i)$. *Then* $w_k = w_0 + k2^{n-s} \bmod 2^n$ *and* k *is the unique integer such that* i *is in* $I_k = \{w_k, w_k + 1, \ldots, w_k + 2^{n-s} - 1\}$.

Using the collection of intervals $\{I_k\}$, we define a bijection $\mathcal{B}$ of $\mathbb{Z}_{2^n}$. The idea is the following. Let $i \in \mathbb{Z}_{2^n}$. Then i is in one of the intervals I_k, and from i, the value of k can be found using Proposition 9. Suppose then that $i = w_k + a$, for some $a \in \mathbb{Z}_{2^{n-s}}$. Let $b = (2k + 1)a \bmod 2^{n-s}$. Since $2k + 1$ is odd, the map $a \mapsto (2k + 1)a \bmod 2^{n-s}$ is a bijection of $\mathbb{Z}_{2^{n-s}}$. So $b \in \mathbb{Z}_{2^{n-s}}$. Let $j = w_k + b$. We set $\mathcal{B}(i)$ to be equal to j. In the following result we provide a more formal description of the bijection $\mathcal{B}$.

Proposition 10 *For* $n \geq 2$, *positive integer* $s < n$ *and* $w_0 \in \mathbb{Z}_{2^n}$, *define* $\mathcal{B}_{n,w_0,s} : \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^n}$ *as follows. For* $i \in \mathbb{Z}_{2^n}$, *the value of* $\mathcal{B}_{n,w_0,s}(i)$ *is determined by the following sequence of steps.*

1. $k \leftarrow \mathcal{I}_{n,w_0,s}(i)$;
2. $w_k \leftarrow w_0 + k2^{n-s} \bmod 2^n$;
3. $a \leftarrow (i - w_k) \bmod 2^n$;
4. $b \leftarrow a(2k + 1) \bmod 2^{n-s}$;
5. $j \leftarrow b + w_k \bmod 2^n$;
6. *set* $\mathcal{B}_{n,w_0,s}(i)$ *to be equal to* j.

The map $\mathcal{B}_{n,w_0,s}$ *defined above is a bijection.*

Proof: From Proposition 9, k is the unique integer such that i is in I_k. Since w_0 is given, w_k is uniquely determined by k and hence w_k is uniquely determined by i. So $a = i - w_k \bmod 2^n$ is an element of $\mathbb{Z}_{2^{n-s}}$, which is uniquely determined by i. Since $2k + 1$ is odd, the map $a \mapsto a(2k + 1) \bmod 2^{n-s}$ is a bijection from $\mathbb{Z}_{2^{n-s}}$ to itself. So b is in $\mathbb{Z}_{2^{n-s}}$ and is uniquely determined by a. Since $j = b + w_k \bmod 2^n$,

b is uniquely determined by a, a itself is uniquely determined by i, and w_k is uniquely determined by i, it follows that j is also uniquely determined by i. This shows that $\mathcal{B}_{n,w_0,s}$ is an injection and hence a bijection. □

Given $\mathbf{x} = (x_1, x_2, \ldots, x_{n-1}, x_n) \in \mathbb{F}_2^n$, let $\mathsf{reverse}(\mathbf{x})$ denote the string $(x_n, x_{n-1}, \ldots, x_2, x_1)$, i.e. $\mathsf{reverse}(\mathbf{x})$ reverses the string $\mathbf{x}$. Using $\mathcal{B}$ and $\mathsf{reverse}$, we define a bijection ϕ from $\mathbb{F}_2^n$ to itself. The idea is the following. Given $\mathbf{x} \in \mathbb{F}_2^n$, change the representation to $i \in \mathbb{Z}_{2^n}$. Let $j = \mathcal{B}(i)$. Change the representation of j from $\mathbb{Z}_{2^n}$ to $\mathbb{F}_{2^n}$, use $\mathsf{reverse}$, and then change the representation back to $\mathbb{Z}_{2^n}$. Apply $\mathcal{B}$ once again and change the representation to $\mathbb{F}_{2^n}$ and produce as the output of ϕ. The description is made precise in the following result.

Proposition 11 *Given $n \geq 2$, positive integer $s < n$ and $w_0 \in \mathbb{Z}_{2^n}$, define a map $\phi_{n,w_0,s} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ as follows. For $\mathbf{x} \in \mathbb{F}_2^n$, the following defines $\phi_{n,w_0,s}(\mathbf{x})$.*

$i \leftarrow \mathsf{int}(\mathbf{x})$; $j \leftarrow \mathcal{B}_{n,w_0,s}(i)$; $\mathbf{y} \leftarrow \mathsf{bin}_n(j)$;
$\mathbf{w} \leftarrow \mathsf{reverse}(\mathbf{y})$;
$i \leftarrow \mathsf{int}(\mathbf{w})$; $j \leftarrow \mathcal{B}_{n,w_0,s}(i)$; $\mathbf{z} \leftarrow \mathsf{bin}_n(j)$;
set $\phi_{n,w_0,s}(\mathbf{x})$ to be equal to $\mathbf{z}$.

The map $\phi_{n,w_0,s}$ described above is a bijection.

Using $\phi_{n,w_0,s}$ and λ-$\mathsf{HWB}_{n,r}$ we define a Boolean function $\mathsf{IntHWB}_{n,w_0,s,\lambda} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ as follows.

$$\mathsf{IntHWB}_{n,w_0,s,\lambda} \quad = \quad \lambda\text{-}\mathsf{HWB}_{n,r} \circ \phi_{n,w_0,s} = \lambda \circ \mathsf{HWB}_{n,r} \circ \phi_{n,w_0,s}. \tag{13}$$

So for $\mathbf{x} \in \mathbb{F}_2^n$,

$$\mathsf{IntHWB}_{n,w_0,s,\lambda}(\mathbf{x}) \quad = \quad \lambda(\mathsf{HWB}_{n,r}(\phi_{n,w_0,s}(\mathbf{x}))). \tag{14}$$

One may note that the application of $\phi_{n,w_0,s}$ to $\mathbf{x}$ corresponds to a pre-processing of the input to λ-$\mathsf{HWB}_{n,r}$.

The parameters to the map $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ are the integers $w_0 \in \mathbb{Z}_{2^n}$, $s < n$ and the r-variable function λ. The number of bits required to store w_0 is n and the number of bits required to store s is $\lceil \log_2 n \rceil$. Assuming that λ is stored in its truth table representation, $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ requires $n + \lceil \log_2 n \rceil + 2^r$ bits to be stored. If $r \ll n$, and in particular if r is constant, then $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ has a very efficient space representation. Computing $\phi_{n,w_0,s}$ requires a few simple arithmetic operations, and computing λ-$\mathsf{HWB}_{n,r}$ requires computing the weight of a string and an application of λ. So the time complexity of $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ is also very efficient. In other words, $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ is a very efficiently implementable function.

Proposition 12 *Let $n \geq 2$, $w_0 \in \mathbb{Z}_{2^n}$, $s, r < n$ be positive integers, and λ be an r-variable function. Then $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ is balanced if and only if λ is balanced.*

Proof: Since $\phi_{n,w_0,s}$ is a bijection, $\phi_{n,w_0,s} \circ \lambda$-$\mathsf{HWB}_{n,r}$ is balanced if and only if λ-$\mathsf{HWB}_{n,r}$ if and only if λ is balanced. □

The requirement is to choose w_0, s and λ in a manner so that $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ has high nonlinearity. Since $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ is constructed using the composition operator, using (8) the Walsh transform of $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ can be expressed in terms of the Walsh transforms of $\phi_{n,w_0,s}$, $\mathsf{HWB}_{n,r}$ and λ. The resulting expression, however, does not provide guidance on how to choose the parameters of $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ to ensure high nonlinearity. Further, we are also not aware of any other analytical method for ensuring

that $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ has high nonlinearity. In view of this, we decided to search for appropriate parameters so that $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ has high nonlinearity. Letting $w_0 \in \mathbb{Z}_{2^n}$, $s \leq \lfloor n/2 \rfloor$ and λ to be a balanced r-variable function make the size of the parameter space $O(n2^n\binom{2^r}{2^{r-1}})$. For each selection of parameters in this space, it is required to construct the function $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ and compute its nonlinearity. This requires $O(n2^n)$ time. So the total time for the search becomes $O(n^2 2^{2n}\binom{2^r}{2^{r-1}})$. This is computationally infeasible. So we decided to fix $r = 5$ and consider the functions $\lambda_{5,i}$ corresponding to the values of n given by Proposition 6. This reduces the search time to $O(n^2 2^{2n})$. For $n = 13, \ldots, 20$ we were able to carry out this search. The search algorithm is given in Algorithm 2.

Algorithm 2: The search procedure for $\mathsf{IntHWB}_{n,w_0,s,\lambda}$.

Input: n, $\mathcal{L}$, where $\mathcal{L}$ is the list of $\lambda_{5,i}$ corresponding to n as given in Proposition 6
Output: A list $\mathcal{P}$ of triplets (λ, s, w_0).

```
1  maxnl ← 0; P ← ∅
2  for λ ∈ L do
3      for s in {1, ..., ⌊n/2⌋} do
4          for w_0 in Z_{2^n} do
5              let f = IntHWB_{n,w_0,s,λ}
6              compute nl(f) and deg(f)
7              if deg(f) = n − 1 and maxnl < nl(f) then
8                  maxnl ← nl(f); P ← {(λ, s, w_0)}
9              else
10                 if deg(f) = n − 1 and maxnl = nl(f) then
11                     P ← P ∪ {(λ, s, w_0)}
12 return P
```

Proposition 13 *Suppose $\mathcal{P}$ is returned by Algorithm 2 on input n. For each $(\lambda, s, w_0) \in \mathcal{P}$, the function $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ is a balanced n-variable function having degree equal to $n-1$.*

The results of running Algorithm 2 for $n = 13, \ldots, 20$ are stated in the following proposition.

Proposition 14 *For $n = 13, \ldots, 20$ and λ is one of $\lambda_{5,i}$ given by Proposition 6, the maximum nonlinearities achieved by $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ are as follows.*

1. *$n = 13$: for $s = 4$, $w_0 = 254$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,2}}) = 3952$.*
2. *$n = 14$: for $s = 5$, $w_0 = 13090$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,4}}) = 7974$.*
3. *$n = 15$: for $s = 7$, $w_0 = 21272$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,2}}) = 16062$.*
4. *$n = 16$:*
   for $s = 4$, $w_0 = 16699$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,5}}) = 32290$;
   for $s = 4$, $w_0 = 27429$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,5}}) = 32290$.
5. *$n = 17$: for $s = 4$, $w_0 = 105883$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,1}}) = 64834$.*
6. *$n = 18$: for $s = 5$, $w_0 = 118924$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 130042$.*

7. $n = 19$: *for* $s = 5$, $w_0 = 200085$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,9}}) = 260606$.

8. $n = 20$: *for* $s = 5$, $w_0 = 353518$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,12}}) = 522046$.

5.1 Resistance to Algebraic Attacks

For $n = 13, \ldots, 19$, the algebraic immunities of the functions given in Proposition 14 could be computed on our servers, but for $n = 20$, the process exited abnormally and did not return the value of AI. The values of AI for $n = 13, \ldots, 19$ are stated in the following proposition.

Proposition 15 *The algebraic immunities of the functions in Proposition 14 are as follows.*

1. $n = 13$: *for* $s = 4$, $w_0 = 254$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,2}}) = 6$.

2. $n = 14$: *for* $s = 5$, $w_0 = 13090$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,4}}) = 7$.

3. $n = 15$: *for* $s = 7$, $w_0 = 21272$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,2}}) = 7$.

4. $n = 16$:
   for $s = 4$, $w_0 = 16699$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,5}}) = 8$;
   for $s = 4$, $w_0 = 27429$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,5}}) = 8$.

5. $n = 17$: *for* $s = 4$, $w_0 = 105883$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,1}}) = 9$.

6. $n = 18$: *for* $s = 5$, $w_0 = 118924$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 9$.

7. $n = 19$: *for* $s = 5$, $w_0 = 200085$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,9}}) = 9$.

Note that except for $n = 13, 15$ and 19, in all other cases the algebraic immunities are optimal, and for $n = 13, 15$ and 19, the algebraic immunities are one less than the optimal. We conjecture that the value of AI for the function in Proposition 14 for $n = 20$ is 10. This is based on our further study of algebraic immunities as discussed below.

To further understand the algebraic immunities of the functions in the class $\mathsf{IntHWB}_{n,w_0,s,\lambda}$, we conducted some more experiments. For $n = 13, \ldots, 19$, we fixed λ and s as in Proposition 15 and for 100 randomly chosen values of w_0, we constructed the function $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ and computed its nonlinearity and algebraic immunity. For $n = 14, 16$ and 18, in all the 100 cases the algebraic immunities came out to be $n/2$, i.e. optimal. For $n = 13, 15, 17$ and 19, in all the 100 cases the algebraic immunities came out to be either $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$. Letting a_1 and a_2 to be the number of cases where the algebraic immunities came out to be $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ respectively, we obtained $(a_1, a_2) = (70, 30), (65, 35), (73, 27), (62, 38)$ for $n = 13, 15, 17$ and 19 respectively. So the experiments provide evidence that for even n functions in the class $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ have optimal algebraic immunity, while for odd n, functions in the class $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ have either optimal or almost optimal algebraic immunity, with optimal algebraic immunity occuring for about 30% or more of the cases.

For $n = 17$, the function in Proposition 14 has optimal algebraic immunity. For $n = 13, 15$ and 19, the functions in Proposition 14 have algebraic immunity one less than the optimal. From the results of our above mentioned experiments with 100 random values of w_0, we provide examples of functions for $n = 13, 15$ and 19 with optimal algebraic immunity.

Example 1

- $n = 13$: *for* $s = 3$, $w_0 = 3204$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 3950$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 7$.

- $n = 15$*: for* $s = 4$, $w_0 = 51$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 16036$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 8$.
- $n = 19$*: for* $s = 5$, $w_0 = 471438$, $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,9}}) = 260502$, $\mathsf{AI}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 10$.

Note that for $n = 13$, the nonlinearity of the above example is 3950, while the maximum nonlinearity reported in Proposition 14 is 3952. For $n = 15$, the nonlinearity of the above example is 16036, while the maximum nonlinearity reported in Proposition 14 is 16062. For $n = 19$, the nonlinearity of the above example is 260502, while the maximum nonlinearity reported in Proposition 14 is 260606. So for $n = 13, 15$ and 19, optimal AI can be obtained with a small decrease in nonlinearity.

To assess the resistance of the class of functions to fast algebraic attacks, we computed the FAA-profile for the functions given in Proposition 14 for $n = 13, 14, 15$ and 16 and also for the functions in Example 1. These are given below.

- $n = 13$, $s = 4$, $w_0 = 254$:
  FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,2}}$: $(1, 11), (2, 9), (3, 9), (4, 7), (5, 7)$.
- $n = 13$, $s = 3$, $w_0 = 3204$:
  FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}$: $(1, 10), (2, 9), (3, 9), (4, 7), (5, 7), (6, 6)$.
- $n = 14$, $s = 5$, $w_0 = 13090$:
  FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,4}}$: $(1, 11), (2, 11), (3, 10), (4, 8), (5, 7), (6, 7)$.
- $n = 15$, $s = 7$, $w_0 = 21272$:
  FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,2}}$): $(1, 13), (2, 11), (3, 11), (4, 9), (5, 9), (6, 7)$.
- $n = 15$, $s = 4$, $w_0 = 51$:
  FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}$: $(1, 13), (2, 11), (3, 10), (4, 9), (5, 8), (6, 7), (7, 7)$.
- $n = 16$, $s = 4$, $w_0 = 16699$:
  FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,5}}$: $(1, 13), (2, 12), (3, 11), (4, 10), (5, 9), (6, 8), (7, 7)$.
- $n = 16$, $s = 4$, $w_0 = 27429$:
  FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,5}}$: $(1, 13), (2, 12), (3, 11), (4, 10), (5, 9), (6, 8), (7, 7)$.

We find that almost perfect FAA-profile is achieved in all cases. Consequently, for all such functions f, $\mathsf{FAI}(f) \geq n - 1$. This indicates good resistance of these functions to fast algebraic attacks.

For $n = 17, \ldots, 20$, due to high memory requirement, it was not possible to compute the complete FAA-profiles for the functions in Proposition 14 and Example 1. Below we provide the partial FAA-profiles that could be computed.

- $n = 17$, $s = 4$, $w_0 = 105883$:
  partial FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,1}}$: $(1, 14), (2, 14), (3, 13), (4, 12), (5, 11)$.
- $n = 18$, $s = 5$, $w_0 = 118924$:
  partial FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}$: $(1, 15), (2, 15), (3, 13), (4, 12)$.
- $n = 19$, $s = 5$, $w_0 = 200085$:
  partial FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,9}}$: $(1, 16), (2, 15), (3, 14)$.
- $n = 19$, $s = 5$, $w_0 = 471438$:
  partial FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,9}}$: $(1, 17), (2, 15), (3, 15)$.

	Table 7 of [25]		Proposition 14		CF [9]		c.r. (4)	
n	nl	LLB	nl	LLB	nl	LLB	CR_n	LCRB_n
13	3862	−5.13	3952	−5.83	3988	−6.25	4051	−7.51
14	7816	−5.45	7974	−6.23	8072	−7.09	8128	−8.00
15	15748	−5.69	16062	−6.67	16212	−7.57	16294	−8.51
16	31616	−5.83	32290	−7.10	32530	−8.11	32640	−9.00
17	–	–	64834	−7.54	65210	−8.65	65355	−9.50
18	–	–	130042	−7.99	130594	−9.10	130816	−10.00
19	–	–	260606	−8.41	261294	−9.27	261782	−10.50
20	–	–	522046	−8.87	523234	−9.96	523776	−11.00

Table 3: Comparison of nonlinearities achieved by $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ with Table 7 of [25], the CF functions and the covering radius bound.

n	Table 7 of [25]	Proposition 15	CF [9]
13	(12,6)	(12,6)	(12,7)
14	(12,6)	(13,7)	(13,7)
15	(14,6)	(14,7)	(14,8)
16	(14,7)	(15,8)	(15,8)
17	–	(16,9)	(16,9)
18	–	(17,9)	(17,9)
19	–	(18,9)	(18,10)

Table 4: Comparison of degrees and algebraic immunities of $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ functions with Table 7 of [25] and the CF functions.

- $n = 20$, $s = 5$, $w_0 = 353518$:
  partial FAA-profile for $\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,12}}$: $(1, 17), (2, 16), (3, 15)$.

We observe that in all cases for (e, d) in the above partial FAA-profiles, the relation $e + d \geq n - 2$ holds and we conjecture that for any of these functions f, the relation $\mathsf{FAI}(f) \geq n - 1$ hold.

Remark 5 *From the experimental results we observe that for all the n-variable functions f of the type* IntHWB*, for which we were able to compute the algebraic immunities and the FAA-profiles, we have* $\mathsf{AI}(f) \geq \lfloor n/2 \rfloor$*, and* $\mathsf{FAI}(f) \geq n - 1$*. Further,* $\mathsf{AI}(f) = \lceil n/2 \rceil$ *in several of the cases. This suggests that functions of the type* IntHWB *provide good resistance to algebraic and fast algebraic attacks.*

5.2 Comparison

From Table 1, we note that the nonlinearities reported in Table 7 of [25] are the highest. So we compare the nonlinearities reported in Table 7 of [25] with those of $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ and with the nonlinearites of the CF functions as well as to the values of the covering radius bound. We constructed the CF functions using the primitive polynomials in Appendix A and then computed their nonlinearities. For $n = 13$, the nonlinearity of the CF function that we obtained is higher than the nonlinearity reported in [7]. This is not surprising since the actual function and hence the value of the nonlinearity depends upon the actual primitive polynomial that was used.

The comparison of nonlinearities is shown in Table 3. The comparison of degrees and algebraic immunities are shown in Table 4. Each entry of Table 4 is of the form (d, a), where d is the degree and a is the algebraic immunity. Based on Tables 3 and 4 we make the following observations.

1. The nonlinearities of the IntHWB functions reported in Proposition 14 are higher than the nonlinearities reported in Table 7 of [25]. For $n = 13$, the algebraic immunity of the IntHWB function given by Proposition 15 is equal to the algebraic immunity of the function reported in Table 7 of [25]. For $n \geq 14$, the algebraic immunities of the IntHWB functions reported in Proposition 15 are higher than the algebraic immunities reported in Table 7 of [25]. In view of improvements in both nonlinearities and algebraic immunities, we do not consider the functions reported in Table 7 of [25] any further.

2. For both the CF functions as well as IntHWB, the value of LLB decreases as n increases. The resistance against linear/correlation attacks is determined by the value of LLB rather than the actual value of the nonlinearity. So increasing n provides better resistance to correlation attacks.

3. The nonlinearities of IntHWB functions are lower than those of CF functions. The LLB's of IntHWB functions are about 1.5 bits more than the LLB's of CF functions. Suppose a target value of LLB is fixed and the value is achieved by CF functions for a particular value of n. By choosing a higher value of n, the same value of LLB can be also be achieved by IntHWB functions. To take a concrete example, suppose it is desired that the value of LLB should be at most -8.00. If CF functions are to be used, then the value of n should be at least 16, while if IntHWB functions are to be used, then the value of n should be at least 19. This may seem like a disadvantage for IntHWB functions. Note, however, that IntHWB functions are efficiently implementable. Implementing the 19-variable IntHWB function from Proposition 14 requires a space complexity of $19+5+32 = 56$ bits, whereas thousands of bits will be required to implement a 16-variable CF function. So IntHWB functions provide the option of very cheaply increasing the value of n to achieve a target value of LLB. An additional advantage of using a higher value of n is the increased resistance to algebraic attacks. In the above example, for $n = 16$, CF functions have optimal algebraic immunity of 8, while for $n = 19$ the IntHWB function given in Proposition 14 has algebraic immunity 9 (see Proposition 15).

5.3 Higher Values of n

It becomes very time consuming to run Algorithm 2 for n greater than 20. To obtain an idea of the nonlinearity achieved by $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ for higher values of n we conducted some experiments. We fixed $s = 5$ and $\lambda = \lambda_{5,7}$ and constructed $\mathsf{IntHWB}_{n,w_0,s,\lambda}$ for a number of random choices of w_0. For $n = 21, \ldots, 24$, we chose 10000 values for w_0, while for $n = 25, \ldots, 30$, we chose 1000 values for w_0. For each $n = 21, \ldots, 30$, in the following example, we report the maximum nonlinearity that was achieved.

Example 2

- $n = 21$, $s = 5$, $w_0 = 1948971$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 1045280$.
- $n = 22$, $s = 5$, $w_0 = 223972$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 2092280$.
- $n = 23$, $s = 5$, $w_0 = 2179192$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 4187200$.
- $n = 24$, $s = 5$, $w_0 = 11878200$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 8378102$.
- $n = 25$, $s = 5$, $w_0 = 17211712$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 16761306$.
- $n = 26$, $s = 5$, $w_0 = 45478445$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 33530292$.
- $n = 27$, $s = 5$, $w_0 = 67070690$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 67070690$.

	Example 2		CF [9]		c.r. (4)	
n	nl	LLB	nl	LLB	CR_n	LCRB_n
21	1045280	−9.31	1046846	−10.24	1047852	−11.50
22	2092280	−9.75	2094936	−10.89	2096128	−12.00
23	4187200	−10.21	4190834	−11.24	4192856	−12.50
24	8378102	−10.64	8383446	−11.67	8386560	−13.00
25	16761306	−11.04	16769938	−12.17	16774320	−13.50
26	33530292	−11.44	33545384	−12.86	33550336	−14.00
27	67070690	−11.78	67097318	−13.50	67103072	−14.50
28	134157910	−12.13	134201202	−13.99	134209536	−15.00
29	268332760	−12.35	268409892	−14.36	268423871	−15.50
30	536691884	−12.55	536833704	−14.82	536854528	−16.00

Table 5: Comparison of nonlinearities achieved by the functions in Example 2 with those of CF functions and the covering radius bound.

- $n = 28$, $s = 5$, $w_0 = 95163654$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 134157910$.
- $n = 29$, $s = 5$, $w_0 = 224553125$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 268332760$.
- $n = 30$, $s = 5$, $w_0 = 378168951$: $\mathsf{nl}(\mathsf{IntHWB}_{n,w_0,s,\lambda_{5,7}}) = 536691884$.

In Table 5, we compare the nonlinearities in Example 2 with those of the CF-function. Note that for $n = 21, \ldots, 30$, even though we were able to explore a very limited portion of the parameter space of IntHWB functions, the nonlinearities and the values of LLB that are achieved compare quite well to the corresponding values of the CF functions. In particular, the values of LLB for the IntHWB functions is at most about 2 more than those of the CF functions. As explained in Section 5.2, the main advantage of IntHWB functions is their very efficient implementation. So a target value of LLB can be cheaply achieved by increasing the value of n. While a CF function would achieve the same value of LLB for a smaller value of n, it would be much more efficient to implement an IntHWB function with a higher value of n. Further, based on our experiments for $n = 13$ to $n = 20$ reported in Section 5.1, we conjecture that even for $n > 20$, the IntHWB functions provide good resistance to algebraic attacks (see Remark 5).

6 A Construction with Provably High Nonlinearity

The CF functions enjoy provable guarantees. They are known to have optimal algebraic immunity [9] and good resistance to fast algebraic attacks [23]. Further, a lower bound on the nonlinearity of the CF function is known [9] (though the lower bound is loose and the actual nonlinearity is substantially higher than the lower bound).

In this section, we provide a construction which yields functions whose provable properties are in some sense a dual of those of the CF functions. The functions constructed in this section have provably high nonlinearity. In fact, for a fixed value of n, the nonlinearity of the function constructed in this section is higher than the nonlinearities of the CF function on n variables. On the other hand, we are only able to prove a general lower bound on the algebraic immunity of the functions. Our claim on algebraic immunities of the constructed functions is a conjecture which is based on experimental results. Similar to the functions constructed in Section 5, the crucial advantage of the functions constructed in

this section over the CF functions is their very efficient implementation. Later we discuss this issue in details.

The construction described in this section is obtained by combining several known components. This results in balanced nonlinear functions with maximum degree, provably high nonlinearity and conjectured high algebraic immunity.

6.1 Maiorana-McFarland Bent Functions

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

$$\begin{aligned} \mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y}) &= \langle \pi(\mathbf{X}), \mathbf{Y} \rangle \oplus h(\mathbf{X}) \\ &= \pi_1(\mathbf{X})Y_1 \oplus \cdots \oplus \pi_m(\mathbf{X})Y_m \oplus h(\mathbf{X}). \end{aligned} \tag{15}$$

We extend the definition of MM to all positive integers in the following manner.

$$\begin{aligned} \mathsf{MM}_1(W) &= W, \\ \mathsf{MM}_{2m+1}(W, \mathbf{X}, \mathbf{Y}) &= W \oplus \mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y}), \quad \text{for } m \geq 1. \end{aligned} \tag{16}$$

The nonlinearity of MM_n stated in the following result is well known, while the result on the degree is simple to obtain.

Proposition 16 *For $m \geq 1$,*

1. $\mathsf{nl}(\mathsf{MM}_{2m}) = 2^{2m-1} - 2^{m-1}$, *and* $\mathsf{nl}(\mathsf{MM}_{2m+1}) = 2^{2m} - 2^m$.
2. $\mathsf{deg}(\mathsf{MM}_{2m+1}) = \mathsf{deg}(\mathsf{MM}_{2m}) = \max(\mathsf{deg}(\pi_1) + 1, \ldots, \mathsf{deg}(\pi_m) + 1, \mathsf{deg}(H))$.

Remark 6 *Note that the nonlinearity of MM_n does not depend on the choices of the bijection π and the function h. We exploit this feature later.*

To the best of our knowledge the following result on the algebraic immunity of MM_n is new.

Theorem 1 *Suppose $m \geq 1$.*

1. $\mathsf{AI}(\mathsf{MM}_{2m}) \leq \mathsf{AI}(\mathsf{MM}_{2m+1}) \leq 1 + \mathsf{AI}(\mathsf{MM}_{2m})$.
2. *There is an $\boldsymbol{\omega}^\star \in \mathbb{F}_2^m$ such that* $\mathsf{AI}(\mathsf{MM}_{2m}) \geq \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{AI}\big(\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})\big)$.

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

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

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

this case, g_1 is an annihilator for $1 \oplus \mathsf{MM}_{2m}$, and so $\mathsf{deg}(g) \geq 1 + \mathsf{deg}(g_1) \geq 1 + \mathsf{AI}(\mathsf{MM}_{2m})$. Consequently, in both cases $\mathsf{deg}(g) \geq \mathsf{AI}(\mathsf{MM}_{2m})$.

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

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

$$\begin{aligned} 0 &= g(\mathbf{X}, \mathbf{Y})\mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y}) \\ &= \left(\bigoplus_{\boldsymbol{\omega} \in \mathbb{F}_2^m} \mathbf{Y}^{\boldsymbol{\omega}} g_{\boldsymbol{\omega}}(\mathbf{X}) \right) \big(\pi_1(\mathbf{X}) Y_1 \oplus \cdots \oplus \pi_m(\mathbf{X}) Y_m \oplus h(\mathbf{X}) \big). \end{aligned} \tag{17}$$

Since the right hand side of (17) is equal to 0, for $\boldsymbol{\omega} \in \mathbb{F}_2^m$, the coefficient of $\mathbf{Y}^{\boldsymbol{\omega}}$ in the expansion on the right hand side of (17) must be equal to 0. Since $g(\mathbf{X}, \mathbf{Y}) \neq 0$, let $w \geq 0$ be the minimum integer such that there is an $\boldsymbol{\omega}^\star$ with $\mathsf{wt}(\boldsymbol{\omega}^\star) = w$ and $g_{\boldsymbol{\omega}^\star}(\mathbf{X}) \neq 0$. In (17), equating the coefficient of $\mathbf{Y}^{\boldsymbol{\omega}^\star}$ to 0, we have

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

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

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

6.2 Construction of Balanced Functions

Note that MM_{2m+1} is a balanced function having nonlinearity $2^{2m} - 2^m$ and degree at most m. We next provide a construction of a balanced function on $2m+1$ variables which has nonlinearity $2^{2m} - 2^m$ and degree $2m$. This construction is given in Theorem 10(a) of [31]. The description is in terms of the string representation of Boolean functions. We provide an equivalent description in terms of the algebraic normal form.

For $m \geq 0$, we define Bal_{2m+1}. Let $\lambda_1(U, V)$, $\lambda_2(U, V)$ and $\lambda_3(U, V)$ be the three distinct non-constant linear functions on two variables. Let h_1 be a bent function on $2m-2$ variables, h_2 be a balanced function on $2m-3$ variables having nonlinearity $2^{2m-4} - 2^{m-2}$, and for $m \geq 3$, let h_3 and h_4 be bent functions on $2m-4$ variables.

Definition 1 *For $m = 0, 1$, we define $\mathsf{Bal}_1(X_1) = X_1$ and $\mathsf{Bal}_3(X_1, X_2, X_3) = X_1 \oplus X_2 X_3$ respectively. For $m \geq 2$, define*

$$\begin{aligned} &\mathsf{Bal}_{2m+1}(X_1, \ldots, X_{2m+1}) \\ &\quad = (1 \oplus X_1)\big(h_1(X_2, \ldots, X_{2m-1}) \oplus \lambda_1(X_{2m}, X_{2m+1}) \big) \oplus X_1 f_1(X_2, \ldots, X_{2m+1}), \end{aligned}$$

where $f_1(X_2,\ldots,X_{2m+1}) = (1 \oplus X_2)h_2(X_3,\ldots,X_{2m-1}) + X_2 f_2(X_3,\ldots,X_{2m+1})$.

For $m = 2$, $f_2(X_3,X_4,X_5) = (1 \oplus X_3)\lambda_2(X_4,X_5) \oplus X_3\lambda_3(X_4,X_5)$ *and for* $m \geq 3$,

$$\begin{aligned}
&f_2(X_3,\ldots,X_{2m+1})\\
&\quad = \quad (1 \oplus X_3)\big(h_3(X_4,\ldots,X_{2m-1}) \oplus \lambda_2(X_{2m},X_{2m+1})\big) \oplus X_3 f_3(X_4,\ldots,X_{2m+1}),\ \textit{where}\\
&f_3(X_4,\ldots,X_{2m+1})\\
&\quad = \quad (1 \oplus X_4)\cdots(1 \oplus X_{2m-1})\lambda_2(X_{2m},X_{2m+1})\\
&\qquad \oplus\big(1 \oplus (1 \oplus X_4)\cdots(1 \oplus X_{2m-1})\big)\big(h_4(X_4,\ldots,X_{2m-1}) \oplus \lambda_3(X_{2m},X_{2m+1})\big).
\end{aligned}$$

Proposition 17 (Theorem 10(a) of [31]) *For* $m \geq 1$, Bal_{2m+1} *is a balanced function having degree* $2m$ *and nonlinearity* $2^{2m} - 2^m$. *Consequently,* $\mathsf{LLB}(\mathsf{Bal}_{2m+1}) = -m$.

For concreteness, we make the following choices.

Concrete choices of $\lambda_1, \lambda_2, \lambda_3, h_1, h_2, h_3$ *and* h_4 *in* Bal_{2m+1}, $m \geq 2$:

$$\left.\begin{array}{l}\lambda_1(U,V) = U,\ \lambda_2(V) = V, \lambda_3(U,V) = U \oplus V,\\ h_1 = \mathsf{MM}_{2m-2},\ h_2 = \mathsf{MM}_{2m-3},\ \text{and for } m \geq 3,\ h_3 = h_4 = \mathsf{MM}_{2m-4}.\end{array}\right\} \tag{18}$$

Next we define Bal_{2m} for $m \geq 1$. The construction that we describe is essentially due to Dobbertin [15]. Later we describe the differences between Dobbertin's construction and the construction that we consider in this paper.

Suppose π is the permutation used to define MM_{2m}. Let $\mathbf{a} = (a_1,\ldots,a_m) = \pi^{-1}(\mathbf{0}_m)$. We define $\mathbf{1}_{\mathbf{a}}(\mathbf{X}) = (1 \oplus a_1 \oplus X_1)\cdots(1 \oplus a_1 \oplus X_m)$. For $m \geq 1$, Bal_{2m} is defined as follows. Let $\mathbf{X} = (X_1,\ldots,X_m)$ and $\mathbf{Y} = (Y_1,\ldots,Y_m)$. Then

$$\mathsf{Bal}_{2m}(\mathbf{X},\mathbf{Y}) \quad = \quad \mathsf{MM}_{2m}(\mathbf{X},\mathbf{Y}) \oplus \mathbf{1}_{\mathbf{a}}(\mathbf{X})\mathsf{Bal}_m(\mathbf{Y}). \tag{19}$$

Note that the definition of Bal_{2m} involves the definition of MM_{2m} and several other MM's on smaller number of variables. Each of these MM's has its own π and h. So the construction of Bal_{2m} is parameterised by the bijections π's and the functions h's that are used to construct the various MM's..

Proposition 18 *For* $m \geq 1$, $\mathsf{nl}(\mathsf{Bal}_{2m}) = 2^{2m-1} - 2^m + \mathsf{nl}(\mathsf{Bal}_m)$. *Consequently, for* $n = 2^{n_1}n_2$, *with* $n_1 \geq 1$ *and* n_2 *odd*

$$\begin{aligned}
\mathsf{nl}(\mathsf{Bal}_n) &= 2^{n-1} - 2^{n/2-1} - 2^{n/4-1} + \cdots - 2^{n_2-1} - 2^{\lfloor n_2/2 \rfloor},\\
\mathsf{LLB}(\mathsf{Bal}_n) &= \log_2\left(\frac{2^{n/2-1} + 2^{n/4-1} + \cdots + 2^{n_2-1} + 2^{\lfloor n_2/2 \rfloor}}{2^n}\right).
\end{aligned}$$

Remark 7 *Note that the nonlinearity of* Bal_{2m} *does not depend on the choices of the bijections* π *and the functions* h. *Remark 6 makes a similar observation regarding* MM_{2m}.

Relation to Dobbertin's construction. Dobbertin [15] had proposed a general construction of balanced Boolean functions on an even number of variables. The proposal was to modify a normal bent function on $2m$ variables by inserting a balanced function on m variables on the flat where the bent function is constant. Since Maiorana-McFarland bent functions are normal bent functions, the construction that we considered above is essentially due to Dobbertin. Our reason for choosing Maiorana-McFarland bent functions is that such functions are easy to implement, while it is not clear whether there are

	CF [9]		Bal		c.r. (4)	
n	nl	LLB	nl	LLB	CR_n	LCRB_n
13	3988	−6.25	4032	−7.00	4051	−7.51
14	8072	−7.09	8120	−7.83	8128	−8.00
15	16212	−7.57	16256	−8.00	16294	−8.51
16	32530	−8.11	32628	−8.87	32640	−9.00
17	65210	−8.65	65280	−9.00	65355	−9.50
18	130594	−9.10	130800	−9.91	130816	−10.00
19	261294	−9.27	261632	−10.00	261782	−10.50
20	523234	−9.96	523756	−10.94	523776	−11.00

Table 6: Comparison of nonlinearities of Bal with that of CF functions and the covering radius bound.

efficient methods to implement a general normal bent function. A difference between Dobbertin's construction and ours is in the choice of the function f on an odd number n of variables which terminate the recursive construction. Dobbertin was interested in maximising nonlinearity, and consequently, his suggestion was to choose f having the highest possible nonlinearity among all balanced functions on n variables. Efficieny of implementation was not a goal in Dobbertin's construction. So it is not ensured that the f which has the highest possible nonlinearity is also efficiently implementable. On the other hand, our goal is to ensure efficiency of implementation. Hence, we choose f to be Bal_n which can be implemented quite efficiently. For $n = 1, 3, 5$ and 7, choosing f to be Bal_n also ensures maximum achievable nonlinearity. For odd $n \geq 9$, however, there are balanced functions with nonlinearity higher than Bal_n (see [31] for the first example of such a function for $n = 15$), though the maximum achievable nonlinearity for n-variable balanced functions is not known.

In Table 6 we compare the nonlinearities of CF functions, Bal_n and CR_n for $n = 13, \ldots, 20$. One may note that the nonlinearities of Bal_n are always greater than the nonlinearities of the CF functions. In fact, for even n the gap between $\mathsf{LLB}(\mathsf{Bal}_n)$ and $\mathsf{LLB}(\mathsf{CF}_n)$ increases, whereas the gap between $\mathsf{LLB}(\mathsf{Bal}_n)$ and LCRB_n decreases. In any case, for either even n or odd n, the gap between $\mathsf{LLB}(\mathsf{Bal}_n)$ and $\mathsf{LLB}(\mathsf{LCRB}_n)$ is small. This indicates that the nonlinearities achieved by $\mathsf{LLB}(\mathsf{Bal}_n)$ are quite close to the covering radius bound.

The following result provides the degree of Bal_{2m}.

Proposition 19 *For $m \geq 1$, $\mathsf{deg}(\mathsf{Bal}_{2m}) = 2m - 1$.*

Proof: From the definition of Bal_{2m}, it follows that $\mathsf{deg}(\mathsf{Bal}_{2m}) = m + \mathsf{deg}(\mathsf{Bal}_m)$. The stated result follows by induction on $m \geq 1$ where we need to use the fact from Proposition 17 that if n is odd, then $\mathsf{deg}(\mathsf{Bal}_n) = n - 1$. □

From Propositions 17 and 19, we see that for $n \geq 1$, Bal_n is a balanced function having degree $n - 1$. The nonlinearities of Bal_n are also quite high and in fact higher than the nonlinearities of the CF functions.

Next we consider the algebraic immunity of Bal_{2m}.

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

$$\mathsf{AI}(\mathsf{Bal}_{2m}) \;\geq\; \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{AI}\big(\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X})\rangle \oplus h(\mathbf{X}) \oplus \mathsf{Bal}_m(\boldsymbol{\omega}^\star)\mathbf{1}_{\mathbf{a}}(\mathbf{X})\big).$$

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

$$0 \;=\; g(\mathbf{X}, \mathbf{Y})\mathsf{Bal}_{2m}(\mathbf{X}, \mathbf{Y})$$

$$= \left(\bigoplus_{\boldsymbol{\omega} \in \mathbb{F}_2^m} \mathbf{Y}^{\boldsymbol{\omega}} g_{\boldsymbol{\omega}}(\mathbf{X}) \right) \left(\pi_1(\mathbf{X}) Y_1 \oplus \cdots \oplus \pi_m(\mathbf{X}) Y_m \oplus h(\mathbf{X}) \oplus \mathbf{1}_{\mathbf{a}}(\mathbf{X}) \bigoplus_{\boldsymbol{\omega} \in \mathbb{F}_2^m} b_{\boldsymbol{\omega}} \mathbf{Y}^{\boldsymbol{\omega}} \right). \quad (20)$$

Since the right hand side of (20) is equal to 0, all coefficients of $\mathbf{Y}^{\boldsymbol{\omega}}$ in the expansion on the right hand side of (20) must be equal to 0. Since $g(\mathbf{X}, \mathbf{Y}) \neq 0$, let $w \geq 0$ be the minimum integer such that there is an $\boldsymbol{\omega}^\star$ with $\mathsf{wt}(\boldsymbol{\omega}^\star) = w$ and $g_{\boldsymbol{\omega}^\star}(\mathbf{X}) \neq 0$. In (20), equating the coefficient of $\mathbf{Y}^{\boldsymbol{\omega}^\star}$ to 0, we have

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

Here we have used $\mathsf{Bal}_m(\boldsymbol{\omega}^\star) = \bigoplus_{\boldsymbol{\omega} \leq \boldsymbol{\omega}^\star} b_{\boldsymbol{\omega}}$ (see (2)). So $g_{\boldsymbol{\omega}^\star}(\mathbf{X})$ is an annihilator for $\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \oplus \mathsf{Bal}_m(\boldsymbol{\omega}^\star) \mathbf{1}_{\mathbf{a}}(\mathbf{X})$. Consequently, $\mathsf{deg}(g) \geq \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{deg}(g_{\boldsymbol{\omega}^\star}) \geq \mathsf{wt}(\boldsymbol{\omega}^\star) + \mathsf{AI}(\langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \oplus \mathsf{Bal}_m(\boldsymbol{\omega}^\star) \mathbf{1}_{\mathbf{a}}(\mathbf{X}))$.

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

Theorem 2 provides a lower bound on the AI of Bal_{2m}. We were not able to find any nice lower bound on the AI for Bal_{2m+1}.

Remark 8 *The lower bound on* $\mathsf{AI}(\mathsf{MM}_{2m})$ *is in terms of the AI of the function* $f = \langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X})$, *while the lower bound on* $\mathsf{AI}(\mathsf{Bal}_{2m})$ *is in terms of the AI of the function* $g = \langle \boldsymbol{\omega}^\star, \pi(\mathbf{X}) \rangle \oplus h(\mathbf{X}) \oplus \mathsf{Bal}_m(\boldsymbol{\omega}^\star) \mathbf{1}_{\mathbf{a}}(\mathbf{X})$. *Note that* f *and* g *differ in at most one bit, which shows that the AI of* $\mathsf{AI}(\mathsf{MM}_{2m})$ *and* $\mathsf{AI}(\mathsf{Bal}_{2m})$ *are quite close (as we will later see from experimental results).*

6.3 Choices of π and h in MM

The nonlinearity and degree of MM_{2m} and Bal_{2m} (and also MM_{2m+1} and Bal_{2m+1}) do not depend on the choice of π and h. On the other hand, the following simple result shows that not all choices of h and π provide good algebraic immunity.

Proposition 20 *Suppose in the construction of* MM_{2m}, π *is chosen to be the identity permutation and* h *is chosen to be the constant function zero. Then* $\mathsf{AI}(\mathsf{MM}_{2m}) = 2$ *and* $\mathsf{AI}(\mathsf{Bal}_{2m}) \leq 3$.

Proof: With the stated choice of π and h, $\mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y}) = \langle \mathbf{X}, \mathbf{Y} \rangle$ which is a quadratic function. Hence its algebraic immunity is at most 2. It is easy to argue that the algebraic immunity cannot be 1.

Since $\mathsf{Bal}_{2m}(\mathbf{X}, \mathbf{Y}) = \mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y}) \oplus \mathbf{1}_{\mathbf{0}_m}(\mathbf{X}) \mathsf{Bal}_m(\mathbf{Y})$, it follows that for any annihilator $g(\mathbf{X}, \mathbf{Y})$ of $\mathsf{MM}_{2m}(\mathbf{X}, \mathbf{Y})$ and for any $i \in \{1, \ldots, m\}$, $X_i g(\mathbf{X}, \mathbf{Y})$ is an annihilator of $\mathsf{Bal}_{2m}(\mathbf{X}, \mathbf{Y})$. So $\mathsf{AI}(\mathsf{Bal}_{2m}) \leq 3$. □

From Theorems 1 and 2, the AI of MM_{2m} and Bal_{2m} are lower bounded by the AI of some function determined from h and π. This suggests that properly choosing h and π can result in good AI for Bal_{2m}. A simple function which achieves the maximum possible AI is the majority function [13]. Based on this fact we make the following concrete choice.

> *Concrete choice of* h *in* MM_{2m}: Choose the m-variable function h in the construction of MM_{2m} given by (15) to be Maj_m, which is the m-variable majority function.

There are other functions which achieve maximum algebraic immunity [8] and these could also be chosen to instantiate h. Our choice of Maj_m is arguably the simplest choice of a function achieving maximum algebraic immunity.

We also need to choose π. To this end, for a positive integer $n \geq 1$, we define n-$\mathsf{HWBP} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ as follows. For $(x_1, \ldots, x_n) \in \mathbb{F}_2^n$, let $w = \mathsf{wt}(x_1, \ldots, x_n)$. Then n-$\mathsf{HWBP}(x_1, \ldots, x_n)$ is defined to be the following.

$$n\text{-}\mathsf{HWBP}(x_1, \ldots, x_n) \quad = \quad \begin{cases} (0, \ldots, 0) & \text{if } w = 0, \\ (x_w, x_{w+1}, \ldots, x_n, x_1, \ldots, x_{w-1}) & \text{otherwise.} \end{cases}$$

From the definition, it easily follows that n-HWBP is a bijection from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Note that n-HWBP_1, i.e. the first coordinate function of n-HWBP is the HWB function. In fact, for $i \geq 1$, n-HWBP_i returns the bit x_{w+i-1} (where the subscript is taken modulo n in the set of residues $\{1, \ldots, n\}$). We define the following concrete choice of π.

> *Concrete choice of π in MM_{2m}*: Choose the m-bit to m-bit permutation π in the construction of MM_{2m} given by (15) to be m-HWBP.

Note that m-$\mathsf{HWBP}^{-1}(\mathbf{0}_m) = \mathbf{0}_m$, and so the function $\mathbf{1}_{\mathbf{a}}(\mathbf{X})$ in the construction of Bal_{2m} is simply $\mathbf{1}_{\mathbf{0}_m}(\mathbf{X}) = (1 \oplus X_1) \cdots (1 \oplus X_m)$.

BDD complexity. Binary decision diagrams (BDD) have been used to attack stream ciphers [21]. It is known that the HWB function has high BDD complexity [4, 1, 20]. So an additional advantage of choosing $\pi = m$-HWBP is that with this choice the functions MM and Bal provide good resistance to BDD attacks.

Remark 9 *Since the degree of the HWB_n function is known to be $n - 1$ [33], with the above choice of π, we obtain the degree of MM_{2m} to be m which is the highest possible degree for a bent function.*

Remark 10 *The function Maj_m has algebraic immunity $\lceil m/2 \rceil$. So the choice of $h = \mathsf{Maj}_m$ in the construction of MM_{2m} is heuristically motivated by the consideration that the algebraic immunity of MM_{2m} and Bal_{2m} (see Remark 8) will be at least $\lceil m/2 \rceil$. More generally, we heuristically expect the algebraic immunity of Bal_n to be at least $\lceil n/4 \rceil$.*

With the choices of $h = \mathsf{Maj}_m$ and $\pi = m$-HWBP, the values of algebraic immunity of MM_n and Bal_n for various n are given in Table 7. In the table, for illustrating the point made by Remark 10, we also provide the values of the algebraic immunity of $\mathsf{Maj}_{\lceil n/2 \rceil}$ and $\mathsf{HWB}_{\lceil n/2 \rceil}$. The values in Table 7 go beyond our heuristic expectation stated in Remark 10. In Table 8, we provide the FAA-profiles and FAI of MM_n and Bal_n, for $n = 4, \ldots, 20$. For $4 \leq n \leq 17$, the complete FAA-profiles are provided. For $n = 18, 19, 20$, we could compute only the partial FAA-profiles and these are provided. For $n = 4$ and $n = 6, \ldots, 17$, we find that $\mathsf{FAI}(\mathsf{Bal}_n) = 1 + \lfloor n/2 \rfloor$, whereas for $n = 5$, we have $\mathsf{FAI}(\mathsf{Bal}_5) = 2 + \lfloor n/2 \rfloor$. From the complete FAA-profiles for $n = 4, \ldots, 17$, we see that $\mathsf{FAI}(\mathsf{Bal}_n)$ is equal to $2 + d$, where $(1, d)$ is in the FAA-profile of Bal_n. Based on these experimentally observed values, we put forward the following conjecture which is stronger than Remark 10.

Conjecture 1 *For $m \geq 1$, let MM_{2m} and MM_{2m+1} be defined as in (15) and (16) respectively, where $h = \mathsf{Maj}_m$ and $\pi = m$-HWBP in the definition of MM_{2m} given by (15). For $m \geq 1$, let Bal_{2m} be defined as in (19), and for $m \geq 0$, let Bal_{2m+1} be given by Definition 1 with the concrete choices of the component functions given by (18). Then for all $n \geq 6$, we have the following.*

1. $\lfloor n/3 \rfloor \leq \mathsf{AI}(\mathsf{MM}_n) \leq \mathsf{AI}(\mathsf{Bal}_n) \leq 1 + \lfloor n/3 \rfloor$.

2. $\lfloor n/2 \rfloor \leq \mathsf{FAI}(\mathsf{MM}_n) \leq \mathsf{FAI}(\mathsf{Bal}_n) = 1 + \lfloor n/2 \rfloor$.

From Table 7, note that other than $n = 16$, the upper bound on $\mathsf{AI}(\mathsf{Bal}_n)$ stated in Conjecture 1 is achieved for all n in $\{4, \ldots, 20\}$. For $n = 16$, we have $\mathsf{AI}(\mathsf{Bal}_n) = \lfloor n/3 \rfloor = 5$.

6.4 Efficiency of Computing Bal_n

Bal_{2m+1} is built from MM_{2m-2}, MM_{2m-3} and MM_{2m-4}. On the other hand, Bal_{2m} is built from MM_{2m} and Bal_m. So in both cases, the efficiency of computing Bal_n boils down to computing MM_n. Since MM_{2m+1} is defined from MM_{2m}, it is sufficient to consider the efficiency of computing MM_{2m}.

The computation of MM_{2m} requires computing h and π, and an inner product of two m-bit strings. The computations of both h and π require the computation of the weight of an m-bit string. Since $h = \mathsf{Maj}_m$, apart from the weight of its input, the computation of h additionally requires the computation of a threshold function. For the computation of $\pi = \mathsf{HWBP}$, other than the weight of its input, a cyclic shift is required. The number of bit operations (or gates) required to compute MM_{2m} is $\mathtt{W}_m + \mathtt{T}_m + \mathtt{C}_m + \mathtt{I}_m$, where

- $\mathtt{W}_m$ is the number of bit operations (or gates) required to compute the weight of an m-bit string.
- $\mathtt{T}_m$ is the number of bit operations (or gates) required to compute the threshold operation for an m-bit input.
- $\mathtt{C}_m$ is the number of bit operations (or gates) required to compute a cyclic shift of an m-bit string.
- $\mathtt{I}_m$ is the number of bit operations (or gates) required to compute an inner product of two m-bit strings.

From [2], we have $\mathtt{W}_m = O(m)$. Also, it is easy to see that $\mathtt{T}_m$, $\mathtt{C}_m$, and $\mathtt{I}_m$ are all $O(m)$. Computing MM_{2m+1} requires computing MM_{2m} and a XOR operation. So we have the following result.

Proposition 21 *The number of bit operations (or gates) required to compute either* MM_{2m} *or* MM_{2m+1} *is* $O(m)$.

The following result states the efficiency of computing Bal_n.

Proposition 22 *The number of bit operations (or gates) required to compute either* Bal_{2m} *or* Bal_{2m+1} *is* $O(m)$.

Proof: Let $T(n)$ be the number of bit operations required to compute Bal_n.

First we consider Bal_{2m+1}. This requires the computation of MM_{2m-2}, MM_{2m-3} and MM_{2m-4}. The computation of MM_{2m-3} it turn also requires the computation of MM_{2m-4}. So from Proposition 21, we have $T(2m+1)$ is $O(m)$.

To compute Bal_{2m}, it is required to compute MM_{2m}, the function $\mathbf{1}_{\mathbf{0}_m}$ and Bal_m. Clearly, the number of bit operations required to compute $\mathbf{1}_{\mathbf{0}_m}$ is $O(m)$, and from Proposition 21, the number of bit operations required to compute MM_{2m} is $O(m)$. So we have the relation

$$T(2m) \quad = \quad O(m) + T(m).$$

Expanding the recursion and using the above proved fact that for odd n, $T(n)$ is $O(n/2)$, we have the required result. □

n	$\mathsf{AI}(\mathsf{Maj}_{\lceil n/2 \rceil})$	$\mathsf{AI}(\mathsf{HWB}_{\lceil n/2 \rceil})$	$\mathsf{AI}(\mathsf{MM}_n)$	$\mathsf{AI}(\mathsf{Bal}_n)$
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 7: Algebraic immunities of MM and Bal.

n	FAA-profile of MM_n	$\mathsf{FAI}(\mathsf{MM}_n)$	FAA-profile of Bal_n	$\mathsf{FAI}(\mathsf{Bal}_n)$
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)	–	(1,9), (2,8)	–

Table 8: (Partial) FAA-profiles and FAI of MM and Bal.

Suppose $n = 2^{n_1} n_2$, with $n_1 \geq 1$ and n_2 odd. Then the actual number of bit operations required to compute Bal_n is

$$\mathtt{X}_{n_2} + \sum_{i=1}^{n_1} (\mathtt{W}_{n/2^i} + \mathtt{T}_{n/2^i} + \mathtt{C}_{n/2^i} + \mathtt{I}_{n/2^i}),$$

where $\mathtt{X}_{n_2}$ is the number of bit operations required to compute Bal_{n_2}. If the value of n_2 is not too large (say $n \leq 5$), then the computation of this Bal_{n_2} will be done using either a table look-up or a simple combinational circuit.

6.5 Cryptographic Considerations

From a cryptographic point of view, the number of variables in the function is not an overriding concern. Rather, it is the efficiency of implementation that is the major issue. For example, an 80-variable function which can be implemented very efficiently would be much more preferable to a 20-variable function which cannot be implemented more efficiently than essentially implementing the truth table. Comparison between constructions have typically fixed the number of variables and then compared the cryptographic properties of functions obtained from different constructions. Such a comparison implicitly assumes that functions with the same number of variables obtained from different constructions have roughly the same level of efficiency in implementation. While this is true for small n, the assumption is invalid when n is even of moderate size. For example, the implementation of Bal_n is much more efficient than the implementation of an n-variable CF function.

In view of the above discussion, we adopt the following approach. For security, we consider two parameters, namely LLB and FAI. The parameter LLB is the major component in quantifying the resistance to (fast) correlation attacks, while the parameter FAI is the major component in determining resistance to algebraic attacks.

Given positive integers ℓ and δ, we say that a Boolean function f provides (ℓ, δ)-security if $\mathsf{LLB}(f) \leq -\ell$ and $\mathsf{FAI}(f) \geq \delta$. A stream cipher designer would set forth the pair (ℓ, δ) based on the desired level of resistance to known attacks and then be interested in obtaining a function f which is (ℓ, δ)-secure. We tie up efficiency of implementation with security by requiring that f is implementable in time and space which is polynomial in ℓ and δ.

Next, we argue that the construction Bal_{2m} satisfies the above stated goal of a stream cipher designer.

Proposition 23 *Let ℓ be a positive integer. Then for $\lfloor n/2 \rfloor \geq \ell$, $\mathsf{LLB}(\mathsf{Bal}_n) \leq -\ell$.*

Proof: First suppose $n = 2m + 1$, with $m \geq \ell$. We have $\mathsf{LLB}(\mathsf{Bal}_{2m+1}) = -m \leq -\ell$.

Now suppose $n = 2m$, with $m \geq \ell$. Writing $n = 2^{n_1} n_2$, we have

$$\begin{aligned} \mathsf{LLB}(\mathsf{Bal}_n) &= \log_2 \left(\frac{2^{m-1} + 2^{m/2-1} + \cdots + 2^{n_2-1} + 2^{\lfloor n_2/2 \rfloor}}{2^{2m}} \right) \\ &< \log_2 \left(\frac{2^{m-1} + 2^{m-1}}{2^{2m}} \right) \\ &= -m \leq -\ell. \end{aligned}$$

□

Conjecture 1 states that for $n \geq 6$, $\mathsf{FAI}(\mathsf{Bal}_n) = 1 + \lfloor n/2 \rfloor$. Given a value of δ, we propose choosing $n = 2\delta$ and then by Conjecture 1, we have $\mathsf{FAI}(\mathsf{Bal}_n) = 1 + \lfloor n/2 \rfloor = 1 + \delta$.

Suppose now that we require an (ℓ, δ)-secure function. Choose $n = 2 \cdot \max\{\ell, \delta\}$. Then Bal_n is an (ℓ, δ)-secure function, where $\mathsf{LLB}(\mathsf{Bal}_n) \leq -\ell$ is guaranteed by Proposition 23, while $\mathsf{FAI}(\mathsf{Bal}_n) = 1 + \delta$ is based upon Conjecture 1. Proposition 22 shows that Bal_n can be computed in time and space linear in ℓ and δ. So the construction Bal_n provides a good theoretical as well as a practical solution to the stream cipher designer's problem.

Remark 11 *We note that in practical applications where a Boolean function is used as the filtering function in the nonlinear filter model of stream ciphers, it may not be possible to arbitrarily increase the value of n. This is due to the constraint that the number of variables n of the Boolean function is at most the number of bits required to store the state of the underlying state machine. Further, it may not be desirable to extract too many bits from the state.*

Transciphering. One of the applications of stream ciphers is to transciphering in the context of homomorphic encryption [28]. See for example the stream cipher FiLIP [19, 24]. This application requires the filtering function of the stream cipher to be homomorphic friendly. Both the majority and the HWB functions are homomorphic friendly [19, 24, 26]. Since Bal_n is built based on these two functions and other very simple bit operations, the function Bal_n is also homomorphic friendly.

Concrete examples. We consider three concrete examples.

Suppose it is desired to construct a $(20, 20)$-secure function f, i.e. $\mathsf{LLB}(f) \leq -20$ and $\mathsf{FAI}(f) \geq 20$. We choose $n = 2 \cdot \max\{20, 20\} = 40$. Then Bal_{40} is $(20, 20)$-secure, where $\mathsf{LLB}(\mathsf{Bal}_{40}) \leq -20$ is guaranteed by Proposition 23, while $\mathsf{FAI}(\mathsf{Bal}_{40}) = 21$ is based upon Conjecture 1. The number of bit operations required for computing Bal_{40} is

$$\mathtt{X}_5 + \sum_{i=0}^{2} (\mathtt{W}_{5 \cdot 2^i} + \mathtt{T}_{5 \cdot 2^i} + \mathtt{C}_{5 \cdot 2^i} + \mathtt{I}_{5 \cdot 2^i}),$$

where $\mathtt{X}_5$ is the number of bit operations required to compute Bal_5.

Suppose it is desired to construct a $(40, 40)$-secure function f, i.e. $\mathsf{LLB}(f) \leq -40$ and $\mathsf{FAI}(f) \geq 40$. We choose $n = 2 \cdot \max\{40, 40\} = 80$. Then Bal_{80} is $(40, 40)$-secure, where $\mathsf{LLB}(\mathsf{Bal}_{80}) \leq -40$ is guaranteed by Proposition 23, while $\mathsf{FAI}(\mathsf{Bal}_{80}) = 41$ is based upon Conjecture 1. The number of bit operations required for computing Bal_{80} is

$$\mathtt{X}_5 + \sum_{i=0}^{3} (\mathtt{W}_{5 \cdot 2^i} + \mathtt{T}_{5 \cdot 2^i} + \mathtt{C}_{5 \cdot 2^i} + \mathtt{I}_{5 \cdot 2^i}).$$

Suppose it is desired to construct a $(64, 64)$-secure function f, i.e. $\mathsf{LLB}(f) \leq -64$ and $\mathsf{FAI}(f) \geq 64$. Choose $n = 128$. Then Bal_{128} is $(64, 64)$-secure, where $\mathsf{LLB}(\mathsf{Bal}_{128}) \leq -128$ is guaranteed by Proposition 23, while $\mathsf{FAI}(\mathsf{Bal}_{128}) = 65$ is based upon Conjecture 1. The number of bit operations required to compute Bal_{128} is

$$\sum_{i=1}^{6} (\mathtt{W}_{2^i} + \mathtt{T}_{2^i} + \mathtt{C}_{2^i} + \mathtt{I}_{2^i}).$$

To utilise an n-bit Boolean function in the nonlinear combiner model, the number of bits used to represent the state has to be at least n. (See Remark 11.) For the examples above, the values of n are 40, 80 and 128. In practice, we expect state machines to be have states with substantially larger number of bits.

6.6 Comparison to IntHWB Functions

Comparing the nonlinearities of Bal given in Table 3 with the nonlinearities of IntHWB given in Table 6, we see that the nonlinearities of Bal are substantially higher. On the other hand, comparing the algebraic immunities of Bal given in Table 7 with the algebraic immunities of IntHWB given in Table 4, we find that the algebraic immunities of IntHWB are higher. So while both Bal and IntHWB are efficiently implementable functions, for a fixed value of n, the choice between the two types of functions, i.e. Bal and IntHWB represents a trade-off between nonlinearity and algebraic resistance. If a function on 20 or less variables is required with excellent algebraic resistance and good nonlinearity, then IntHWB will be preferable. If nonlinearity is to be given priority, then Bal will be preferable. If the number of variables is not a constraint, then as discussed in Section 6.5, by appropriately choosing n, Bal_n can be used to achieve any given target resistance against (fast) correlation attacks as well as (fast) algebraic attacks.

7 Conclusion

We have described two families of functions which are efficient to implement and achieve a good combination of nonlinearity and algebraic resistance making them excellent choices for use as the filtering function in the filter model of stream ciphers. The nonlinearity and algebraic resistance achieved by functions in the first family are both substantially better than what is achieved by all previously known families which are efficient to implement. Given a pair of positive integers (ℓ, δ) we show that it is possible to select a function from the second family such that linear bias is provably at most $2^{-\ell}$ and the fast algebraic immunity is at least δ (based on a conjecture). Further, the function can be implemented in time and space linear in ℓ and δ. This provides a good theoretical as well as a very efficient practical solution to the design problem for Boolean functions to be used in the filter model of stream ciphers. There are, however, several questions that remain.

For the first family we provide experimental results on nonlinearities for n up to 30 and on algebraic resistance for n up to 20. It would be good to obtain proofs which apply for general values of n. To the best of our understanding, the presently known proof techniques are difficult to apply to the functions in the first family. So obtaining proofs may require developing new proof techniques.

For the second family, the main open problem is to settle Conjecture 1. Again, due to the combination of the majority and the HWB functions, the known proof techniques seem to be difficult to apply.

From an implementation point of view, it would be of interest to actually propose stream cipher designs based on the functions that have been introduced in this paper. Given the recent interest in transciphering, functions from the second family may provide a good solution to design of stream ciphers for transciphering.

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] Beate Bollig, Martin Löbbing, Martin Sauerhoff, and Ingo Wegener. On the complexity of the hidden weighted bit function for various BDD models. *RAIRO Theor. Informatics Appl.*, 33(2):103–116, 1999. 7, 27

[2] 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`. 28

[3] Nina Brandstätter, Tanja Lange, and Arne Winterhof. On the non-linearity and sparsity of Boolean functions related to the discrete logarithm in finite fields of characteristic two. In Øyvind Ytrehus, editor, *Coding and Cryptography, International Workshop, WCC 2005, Bergen, Norway, March 14-18, 2005. Revised Selected Papers*, volume 3969 of *Lecture Notes in Computer Science*, pages 135–143. Springer, 2005. 6

[4] 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. 2, 7, 27

[5] Claude Carlet. Comments on "Constructions of cryptographically significant Boolean functions using primitive polynomials". *IEEE Trans. Inf. Theory*, 57(7):4852–4853, 2011. 7

[6] Claude Carlet. *Boolean Functions for Cryptography and Coding Theory.* Cambridge University Press, 2021. 1, 3, 4, 6

[7] Claude Carlet. A wide class of Boolean functions generalizing the hidden weight bit function. *IEEE Trans. Inf. Theory*, 68(2):1355–1368, 2022. 2, 7, 13, 19

[8] 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. 27

[9] 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, 6, 19, 21, 25

[10] L. Carlitz and S. Uchiyama. Bounds for exponential sums. *Duke Math. J.*, 24(1):37–41, 1957. 8

[11] 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. 5

[12] 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. 5

[13] 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, 26

[14] F. Didier. A new upper bound on the block error probability after decoding over the erasure channel. *IEEE Trans. Inf. Theory*, 52(10):4496–4503, 2006. 5

[15] 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, 24

[16] Keqin Feng, Qunying Liao, and Jing Yang. Maximal values of generalized algebraic immunity. *Des. Codes Cryptogr.*, 50(2):243–252, 2009. 6

[17] Xiutao Feng and Guang Gong. On algebraic immunity of trace inverse functions on finite fields of characteristic two. *J. Syst. Sci. Complex.*, 29(1):272–288, 2016. 8

[18] Kishan Chand Gupta and Palash Sarkar. Toward a general correlation theorem. *IEEE Trans. Inf. Theory*, 51(9):3297–3302, 2005. 6

[19] Clément Hoffmann, Pierrick Méaux, and Thomas Ricosset. Transciphering, using FiLIP and TFHE for an efficient delegation of computation. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *Progress in Cryptology - INDOCRYPT 2020 - 21st International Conference on Cryptology in India, Bangalore, India, December 13-16, 2020, Proceedings*, volume 12578 of *Lecture Notes in Computer Science*, pages 39–61. Springer, 2020. 31

[20] Donald E. Knuth. *The Art of Computer Programming Volume 4, Fascicle 1: Bitwise Tricks and Techniques: Binary Decision Diagrams*. Addison-Wesley Professional, 2009. 7, 27

[21] Matthias Krause. BDD-based cryptanalysis of keystream generators. 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 222–237. Springer, 2002. 7, 27

[22] Matthias Krause and Dirk Stegemann. Reducing the space complexity of BDD-based attacks on keystream generators. In Matthew J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2006. 7

[23] Meicheng Liu, Yin Zhang, and Dongdai Lin. Perfect algebraic immune functions. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 2012. 5, 6, 21

[24] Pierrick Méaux, Claude Carlet, Anthony Journault, and François-Xavier Standaert. Improved filter permutators for efficient FHE: better instances and implementations. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors, *Progress in Cryptology - INDOCRYPT 2019 - 20th International*

Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings, volume 11898 of *Lecture Notes in Computer Science*, pages 68–91. Springer, 2019. 31

[25] Pierrick Méaux and Yassine Ozaim. On the cryptographic properties of weightwise affine and weightwise quadratic functions. *Discrete Applied Mathematics*, 355:13–29, 2024. 2, 7, 8, 9, 10, 12, 13, 19, 20

[26] Pierrick Méaux, Jeongeun Park, and Hilder V. L. Pereira. Towards practical transciphering for FHE with setup independent of the plaintext space. *IACR Communications in Cryptology*, 1(1), 2024. `https://cic.iacr.org/p/1/1/20`. 31

[27] 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. 5

[28] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011*, pages 113–124. ACM, 2011. 3, 31

[29] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 55–64. Springer, 1993. 8

[30] Oscar S. Rothaus. On "bent" functions. *J. Comb. Theory, Ser. A*, 20(3):300–305, 1976. 4

[31] 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. 3, 23, 24, 25

[32] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949. 1

[33] 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, 7, 13, 27

[34] 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, 7

A Primitive Polynomials Used to Construct CF Functions

For $n = 13$ to 30, the following primitive polynomials were used in the construction of the CF functions.

$x^{13} \oplus x^4 \oplus x^3 \oplus x \oplus 1$
$x^{14} \oplus x^{12}x^{11} \oplus x \oplus 1$

$x^{15} \oplus x \oplus 1$
$x^{16} \oplus x^5 \oplus x^3 \oplus x^2 \oplus 1$
$x^{17} \oplus x^3 \oplus 1$
$x^{18} \oplus x^7 \oplus 1$
$x^{19} \oplus x^6 \oplus x^5 \oplus x \oplus 1$
$x^{20} \oplus x^3 \oplus 1$
$x^{21} \oplus x^2 \oplus 1$
$x^{22} \oplus x \oplus 1$
$x^{23} \oplus x^5 \oplus 1$
$x^{24} \oplus x^4 \oplus x^3 \oplus x \oplus 1$
$x^{25} \oplus x^3 \oplus 1$
$x^{26} \oplus x^6 \oplus x^2 \oplus x^1 \oplus 1$
$x^{27} \oplus x^5 \oplus x^2 \oplus x^1 \oplus 1$
$x^{28} \oplus x^3 \oplus 1$
$x^{29} \oplus x^2 \oplus 1$
$x^{30} \oplus x^{23} \oplus x^2 \oplus x^1 \oplus 1$

B ANFs of $\lambda_{5,i}$, $i = 1, \ldots, 12$

$$\begin{aligned}
\lambda_{5,1}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3 \oplus X_1X_2X_4 \oplus X_1X_2 \oplus X_1X_3X_4X_5 \oplus X_1X_3 \oplus X_1X_5 \oplus X_1 \oplus X_3X_5 \\
&\quad \oplus X_4X_5 \oplus X_4 \oplus X_5 \oplus 1 \\
\lambda_{5,2}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_5 \oplus X_1X_2X_3 \oplus X_1X_2X_5 \oplus X_1X_3X_5 \oplus X_1 \oplus X_2X_3X_5 \oplus X_2X_3 \oplus X_2X_4X_5 \\
&\quad \oplus X_2X_4 \oplus X_3X_4X_5 \oplus X_3X_4 \oplus X_3X_5 \oplus X_4X_5 \oplus X_4 \oplus X_5 \oplus 1 \\
\lambda_{5,3}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_5 \oplus X_1X_2X_5 \oplus X_1X_3X_5 \oplus X_1X_3 \oplus X_1X_4X_5 \oplus X_1 \oplus X_2X_3X_4X_5 \oplus X_2X_4X_5 \\
&\quad \oplus X_2X_5 \oplus X_2 \oplus X_3X_4X_5 \oplus X_3X_4 \oplus X_3 \oplus X_4X_5 \oplus X_4 \oplus 1 \\
\lambda_{5,4}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_4 \oplus X_1X_2X_5 \oplus X_1X_2 \oplus X_1X_3X_4X_5 \oplus X_1X_4 \oplus X_1 \oplus X_2X_3X_4 \oplus X_2X_3 \\
&\quad \oplus X_2X_5 \oplus X_2 \oplus X_3X_4X_5 \oplus X_3X_5 \oplus X_3 \oplus 1 \\
\lambda_{5,5}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_5 \oplus X_1X_2X_3 \oplus X_1X_2X_5 \oplus X_1X_3X_5 \oplus X_1X_5 \oplus X_1 \oplus X_2X_3X_4 \oplus X_2X_4X_5 \\
&\quad \oplus X_2X_4 \oplus X_3X_4X_5 \oplus X_3X_4 \oplus X_3X_5 \oplus X_4X_5 \oplus X_4 \oplus X_5 \oplus 1 \\
\lambda_{5,6}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3 \oplus X_1X_2X_4 \oplus X_1X_2 \oplus X_1X_3X_4X_5 \oplus X_1X_3X_4 \oplus X_1X_4 \oplus X_1 \oplus X_2X_3X_4 \\
&\quad \oplus X_2X_3 \oplus X_2X_4 \oplus X_2 \oplus X_3X_4 \oplus X_3X_5 \oplus X_3 \oplus X_4X_5 \oplus 1 \\
\lambda_{5,7}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_4 \oplus X_1X_2X_5 \oplus X_1X_2 \oplus X_1X_3X_4X_5 \oplus X_1X_3X_4 \oplus X_1X_3X_5 \oplus X_1X_4 \oplus X_3 \\
&\quad \oplus X_4X_5 \oplus X_4 \oplus X_5 \oplus 1 \\
\lambda_{5,8}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_5 \oplus X_1X_2X_3 \oplus X_1X_2X_5 \oplus X_1X_4X_5 \oplus X_1X_4 \oplus X_1X_5 \oplus X_1 \oplus X_2X_4X_5 \\
&\quad \oplus X_2X_4 \oplus X_3X_5 \oplus X_4X_5 \oplus X_4 \oplus X_5 \oplus 1 \\
\lambda_{5,9}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_4 \oplus X_1X_2X_3 \oplus X_1X_2X_4X_5 \oplus X_1X_2X_4 \oplus X_1X_2 \oplus X_1X_3X_4 \oplus X_1X_5 \oplus X_1 \\
&\quad \oplus X_2X_3X_4X_5 \oplus X_2X_4 \oplus X_2 \oplus X_3X_4 \oplus X_3X_5 \oplus X_4 \oplus X_5 \oplus 1 \\
\lambda_{5,10}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_4 \oplus X_1X_2X_3 \oplus X_1X_2X_4X_5 \oplus X_1X_2X_5 \oplus X_1X_3X_4 \oplus X_1X_4X_5 \oplus X_1 \oplus X_2X_3X_4X_5 \\
&\quad \oplus X_2X_4X_5 \oplus X_2X_4 \oplus X_3X_4 \oplus X_3X_5 \oplus X_4X_5 \oplus 1 \\
\lambda_{5,11}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_4 \oplus X_1X_2X_3X_5 \oplus X_1X_2X_3 \oplus X_1X_2X_4X_5 \oplus X_1X_2X_5 \oplus X_1X_3X_4X_5 \oplus X_1X_3X_5 \oplus X_1X_3 \\
&\quad \oplus X_1X_5 \oplus X_1 \oplus X_2X_3X_4X_5 \oplus X_2X_4 \oplus X_2X_5 \oplus X_2 \oplus X_3X_5 \oplus X_4X_5 \oplus X_4 \oplus X_5 \\
\lambda_{5,12}(X_1,X_2,X_3,X_4,X_5) &= X_1X_2X_3X_4 \oplus X_1X_2X_3X_5 \oplus X_1X_2X_4X_5 \oplus X_1X_3X_4X_5 \oplus X_1X_3X_5 \oplus X_1X_3 \oplus X_1X_4X_5 \oplus X_1X_4 \\
&\quad \oplus X_1 \oplus X_2X_3X_4X_5 \oplus X_2X_3 \oplus X_2X_5 \oplus X_2 \oplus X_3X_4X_5 \oplus X_4X_5 \oplus X_4
\end{aligned}$$