The Grain Family of Stream Ciphers: an Abstraction, Strengthening of Components and New Concrete Instantiations

Palash Sarkar

Indian Statistical Institute, 203, B.T. Road, Kolkata, India 700108
Email: palash@isical.ac.in

November 17, 2025

Abstract

The first contribution of the paper is to put forward an abstract definition of the Grain family of stream ciphers which formalises the different components that are required to specify a particular member of the family. Our second contribution is to provide new and strengthened definitions of the components. These include definining new classes of nonlinear Boolean functions, improved definition of the state update function during initialisation, choice of the tap positions, and the possibility of the linear feedback shift register being smaller than the nonlinear feedback shift register. The third contribution of the paper is to put forward seven concrete proposals of stream ciphers by suitably instantiating the abstract family, one at the 80-bit security level, and two each at the 128-bit, 192-bit, and the 256-bit security levels. At the 80-bit security level, compared to the well known Grain v1, the new proposal uses Boolean functions with improved cryptographic properties *and* an overall lower gate count. At the 128-bit level, compared to ISO/IEC standard Grain-128a, the new proposals use Boolean functions with improved cryptographic properties; one of the proposals require a few extra gates, while the other has an overall lower gate count. At the 192-bit, and the 256-bit security levels, there are no proposals in the literature with smaller gate counts.
Keywords: stream cipher, Grain, state initialisation, tap positions, Boolean function, resiliency, algebraic degree, nonlinearity, algebraic immunity, gate count.

1 Introduction

Stream ciphers and block ciphers are the two basic methods of performing symmetric key encryption. Constructions of stream ciphers are typically geared either towards small hardware footprint, or towards very fast software implementation. The focus of the present work is on the former type of stream ciphers. Over the last two decades several hardware oriented stream ciphers have been proposed in the literature. One of the most successful approaches to such stream ciphers is the Grain family of stream ciphers [41, 42, 38, 1, 40, 39]. Other well known constructions include MICKEY 2.0 [4], Trivium [18], Fruit-80 [33], Sprout [2] and its successor Plantlet [63], and Lizard [36]. The regular appearance of new designs is testimony to the enduring interest of the stream cipher community in new proposals of hardware oriented stream ciphers.

The approach taken by the Grain family of stream ciphers is to combine a linear feedback shift register (LFSR) and a nonlinear feedback shift register (NFSR). The stream cipher takes as input a secret key and a publicly known initialisation vector (IV) and produces as output a sequence of bits called the

keystream. Typically the keystream bit sequence is XORed with the message sequence to perform encryption. Some versions of the Grain family also provide an option for performing authenticated encryption with associated data (AEAD), while some other versions perform only AEAD and do not have the option of performing only encryption.

Several variants of Grain have been proposed. The first version [41] was targeted towards the 80-bit security level, but was cryptanalysed [13] soon after being proposed, and led to a modified design named Grain v1 [42]. At the 128-bit security level, a proposal named Grain-128 was put forward [38]. Grain-128 was cryptanalysed using dynamic cube attacks [32, 30], and subsequently a modified version called Grain-128a was proposed [1]. Both Grain v1 and Grain-128a perform encryption, while Grain-128a optionally performs authenticated encryption. In response to a call for proposals for light weight designs by the NIST of USA, a modification of Grain-128a, named Grain-128AEAD, was put forward, and was later modified to the construction Grain-128AEADv2 [40, 39]. Grain-128AEADv2 performs authenticated encryption with associated data and does not have an encryption only mode of operation. Grain v1 (along with MICKEY 2.0 [4] and Trivium [18]) was included in the eSTREAM final portfolio of stream cipher algorithms in the hardware category. Grain-128a is an ISO/IEC standard (ISO/IEC 29167-13:2015).

A number of papers have studied various aspects of the security of the Grain family of stream ciphers. These attacks include linear approximations and fast correlation attacks [13, 59, 76], slide attacks [50, 17], cube attacks [32, 30, 75, 77, 37], near collision attacks [78, 79, 7], conditional differential cryptanalysis [48, 53, 5, 6, 57, 56, 55], time/memory trade-off attacks [27, 51], related key attacks [52, 11, 29], and fault attacks [22, 47, 9, 28, 10, 71, 60, 73, 54, 23, 8, 65].

Many such works have studied reduced versions of Grain, where the number of initialisation rounds is less than the specified number of rounds. The only known attack on the "full" versions of Grain v1 and Grain-128a is a fast correlation attack [76]. However, the attack requires a huge number of keystream bits to be generated from a single pair of key and initialisation vector (IV). In particular, for Grain v1 and Grain-128a, the fast correlation attack in [76] requires $2^{75.1}$ and $2^{113.8}$ bits of keystream respectively to be generated from a single key and IV pair.

The two versions of Grain, namely Grain v1 targeted at the 80-bit security level, and Grain-128a (and its variant Grain-128AEADv2) targeted at the 128-bit security level, have held up remarkably well against various attack ideas that have been proposed over the last two decades. As mentioned above, the only known attack against full Grain v1 and Grain-128a is a fast correlation attack [76], which requires a very large number of keystream bits to be generated from a single key and IV pair. The success of certain members of the Grain family provides the motivation for performing a more comprehensive study of the family with the goal of uncovering other members of the family which are also secure. In this paper we perform such a comprehensive study. An overview of our contributions is given below.

Definition of an abstract family. There are several aspects that distinguish the Grain family of stream ciphers. These are listed below.

1. A combination of an NFSR and an LFSR is used, where the least significant bit of the LFSR is fed into the most significant bit of the NFSR.

2. The LFSR connection polynomial is a low weight binary primitive polynomial.

3. The feedback function of the NFSR consists of a nonlinear function g to which certain other bits of the NFSR are added.

4. The function providing the output keystream bit consists of a nonlinear function h which is applied to certain bits of the NFSR and the LFSR, and to which some other bits of the NFSR and the LFSR are added.

5. There is a mechanism to load the secret key and the IV into the NFSR and the LFSR.

6. There is an initialisation procedure during which no keystream bit is produced, the goal being to obtain a good mix of the key and the IV.

7. There is a next state function during initialisation which is different from the next state function during keystream generation.

Our first contribution is to capture the above aspects in an abstract setting. We define appropriate parameters and mappings which allow us to describe the operation of the Grain family of stream ciphers at an abstract level. Concrete instantiations can then be obtained by suitably defining the values of the parameters.

New designs of constituent Boolean functions. One of the main features of the Grain family consists of the two nonlinear Boolean functions g and h. To a large extent, the security of any particular member of the Grain family depends on the cryptographic properties of the two specific functions defined for that particular member. At the same time, for the stream cipher to be suitable for efficient hardware implementation, the gate counts of g and h should be small. This tension between good cryptographic properties and low gate counts is typically not addressed in the literature on construction of Boolean functions with good cryptographic properties. In most works in the latter direction, the typical question that is studied is that of obtaining Boolean functions which achieve optimal (or, close-to-optimal, or at least good in some sense) trade-offs between various cryptographic properties. The issue of whether the constructed functions have sufficiently low gate counts for use in stream ciphers is mostly not considered. The gold standard of reference on cryptographically useful Boolean functions [19] does not highlight low gate count Boolean functions with good cryptographic properties. There are a few exceptions where gate counts of Boolean functions have received serious attention. Examples are the work [61] on design of stream cipher for use in fully homomorphic encryption, and the work [21] which explicitly considers gate counts of Boolean functions for use in the classical nonlinear filter model of stream cipher.

The nonlinear functions g and h proposed for various members of the Grain family are standalone functions. The papers which introduced Grain v1 and Grain-128a mentioned the nonlinearity, resiliency, and algebraic degree of these functions. It was, however, not clear whether these functions are obtained from a family of functions possessing similar properties. Perhaps more importantly there is no discussion on whether there are other functions with similar (or better) cryptographic properties and similar (or lower) gate counts.

An important contribution of the present work is to systematically identify several classes of functions with good cryptographic properties and low gate counts. While the classes of functions are of some theoretical interest in their own rights, here we focus on their application to the Grain family of stream ciphers. Grain v1 uses a 5-variable function as h and a 10-variable function as g. We identify a 5-variable function h and a 10-variable function g different from those used in Grain v1 having better cryptographic properties *and* lower gate counts than the corresponding functions used in Grain v1. We also identify a 7-variable function with better cryptographic properties than the function h of Grain v1 *and* a lower gate count. Grain-128a uses a 9-variable function as h and a 24-variable function as g. We identify a 10-variable function h and a 24-variable function g different from the functions used in Grain-128a which provide better cryptographic properties compared to the functions h and g of Grain-128a, and

require only a few extra gates for implementation. From our family of Boolean functions we are able to select appropriate functions to instantiate new proposals which are targeted at the 192-bit and the 256-bit security levels.

New state update function during initialisation. An important aspect of the design of the Grain family is the next state function used during initialisation. All proposed members of the Grain family use the same next state function during initialistion. The principle is that along with the next state function for keystream generation, the keystream bit that is generated from the state is not provided as output, and instead is fed back to both the NFSR and the LFSR. The idea behind this strategy is to obtain a good mix of the key and the IV in both the NFSR and the LFSR portions of the state. We examine this principle carefully. There are two nonlinear functions involved in the Grain family, namely g and h. Since the function g is the core nonlinear feedback function of the NFSR, its effect is fed back to the NFSR both during state initialisation as well as during keystream generation. The function h is the core nonlinear function of the keystream generation, and its effect is fed back to the NFSR and the LFSR during state initialisation. This strategy causes an asymmetry during state initialisation. While the effect of both g and h are fed back to the NFSR, only the effect of h is fed back to the LFSR. The asymmetry is particularly glaring, since it is the function g which is bigger and more "complex" than the function h. Having identified this undesirable imbalance in state initialisation, we put forward a new proposal for state updation during initialisation. Our proposal is to feed back the XOR of g and h to both the LFSR and the NFSR. This, however, cannot be directly done since otherwise the state updation operation may not be invertible and hence lose entropy. We provide the exact definition of the new state updation function during initialisation and prove that it is indeed invertible. The net effect of the new state updation function during initialisation is to obtain a more uniform mix of key and IV in both the NFSR and the LFSR which is more "complex" than the state updation function during initialisation used in the existing members of the Grain family.

A systematic method of choosing tap positions. The choice of tap positions for obtaining the inputs to the various functions is an important design decision. The tap positions for the feedback function of the LFSR are determined by the choice of the connection polynomial. On the other hand, for the nonlinear feedback function of the NFSR and also the output function, presently there is no guideline on how to choose the tap positions. We made a deep and general analysis of the strategy for obtaining linear approximations that was used in [76]. Based on this analysis, we identified a combinatorial condition on some of the tap positions which allows certain correlations appearing in the overall correlation of the linear approximation to be provably low. Based on this combinatorial condition, we put forward a general systematic method of choosing all the tap positions. Such a general method is advantageous since it provides a guideline for choosing tap positions for new instantiations of the Grain family of stream ciphers.

Allowing different sizes for the LFSR and the NFSR. In all previous proposals of the Grain family the NFSR and the LFSR are of the same size. In our abstract formulation of the Grain family, we allow the NFSR and the LFSR sizes to be different. We build upon this idea and put forward new concrete proposals at the 128-bit, 192-bit, and the 256-bit security levels where the LFSR is smaller than the NFSR. This leads to the situation where the size of the state of the stream cipher is less than twice the size of the secret key. It is known that this condition opens up the construction to a time/memory trade-off attack. We argue that the memory requirement of such attacks on 128-bit and higher security levels make them physically infeasible. In particular, we note that for the 192-bit and the 256-bit levels,

the attacks would require memory which is more than the number of atoms in the entire world (see Appendix B). From a practical engineering point of view, one may perhaps ignore attacks which are physically impossible to mount.

New concrete proposals. We put forward seven new concrete proposals, one targeted at the 80-bit security level, and two each targeted at the 128-bit, 192-bit and the 256-bit security levels. For the proposals at the 128-bit and higher security levels, one proposal has the NFSR and the LFSR to be of the same length, while the other proposal has the LFSR to be of smaller length than the NFSR. We provide detailed discussions on the security of the new proposals against the various classes of attacks on the Grain family that have been proposed till date. Compared to Grain v1, our proposal at the 80-bit security level offers better cryptographic properties for the core nonlinear functions g and h *and* lower gate counts. Compared to Grain-128a, our first proposal (one with equal length NFSR and LFSR) provides better cryptographic properties for g and h and requires a few extra gates, while our second proposal (one with the LFSR smaller than the NFSR) provides better cryptographic properties for g and h and has a smaller gate count (due to the smaller size of the state). For the 192-bit and the 256-bit security levels, we know of no other stream cipher which provides smaller gate counts.

Scalability. An important advantage of our approach is scalability. To obtain a stream cipher targeted at any desired security level, one may start with our abstract description and then appropriately obtain the nonlinear functions g and h from the family of functions that we have introduced, and also obtain the tap positions using the general method that we describe in this paper. This is to be contrasted with Grain v1 and Grain-128a appearing as kind of ad-hoc constructions at the 80-bit and the 128-bit security levels respectively. An imporant consequence of scalability is that it is possible to quickly scale up the parameters to improve resistance against any future attacks. Such scaling up may consist of either increasing the size of the NFSR, the size of the LFSR, or using larger functions for g or h.

Structure of the paper. The background and preliminaries are given in Section 2. In Section 3 we provide the abstract definition of the Grain family of stream ciphers. New classes of Boolean functions required to obtain new instantiations of the Grain family are presented in Section 4. New designs for the other components of the Grain family are given in Section 5. In Section 6 we present the seven new concrete proposals. Analysis of known attacks and their applicability to the new proposals are described in Section 7, while a possible new attack is described in Section 8. Finally, we conclude the paper in Section 9.

2 Background and Preliminary Results

The cardinality of a finite set S will be denoted by $\#S$. Given a subset S of the set of integers and an integer t, we define $t + S = \{t + s : s \in S\}$. For two subsets S and T of the set of integers, we define $S + T = \{s + t : s \in S, t \in T\} = \cup_{t \in T}(t + S) = \cup_{s \in S}(s + T)$. For a real number x, we define $\mathsf{sgn}(x) = 1$, if $x \geq 0$ and $\mathsf{sgn}(x) = 1$, if $x < 0$.

By $\mathbb{F}_2$ we denote the finite field of two elements; by $\mathbb{F}_2^n$, where n is a positive integer, we denote the vector space of dimension n over $\mathbb{F}_2$. The addition operator over both $\mathbb{F}_2$ and $\mathbb{F}_2^n$ will be denoted by $\oplus$. The product (which is also the logical AND) of $x, y \in \mathbb{F}_2$ will simply be written as xy. A bit vector of dimension n, i.e. an element of $\mathbb{F}_2^n$ will also be considered to be an n-bit binary string.

Let n be a positive integer. The support of $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$ is $\mathsf{supp}(\mathbf{x}) = \{1 \leq i \leq n : x_i = 1\}$, and the weight of $\mathbf{x}$ is $\mathsf{wt}(\mathbf{x}) = \#\mathsf{supp}(\mathbf{x})$. By $\mathbf{0}_n$ and $\mathbf{1}_n$ we will denote the all-zero and all-one strings

of length n respectively. For $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, with $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ the distance between $\mathbf{x}$ and $\mathbf{y}$ is $d(\mathbf{x}, \mathbf{y}) = \#\{i : x_i \neq y_i\}$; the inner product of $\mathbf{x}$ and $\mathbf{y}$ is $\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 \oplus \cdots \oplus x_n y_n$.

Boolean function. An n-variable Boolean function f is a map $f : \mathbb{F}_2^n \to \mathbb{F}_2$. The weight of f is $\mathsf{wt}(f) = \#\{\mathbf{x} \in \mathbb{F}_2^n : f(\mathbf{x}) = 1\}$; f is said to be *balanced* if $\mathsf{wt}(f) = 2^{n-1}$. The function f is said to be non-degenerate on the i-th variable if there are $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{F}_2^n$ such that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ differ only in the i-th position, and $f(\boldsymbol{\alpha}) \neq f(\boldsymbol{\beta})$; if there is no such pair $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, then f is said to be degenerate on the i-th variable.

The *algebraic normal form (ANF) representation* of an n-variable Boolean function f is the representation of f as an element of the polynomial ring $\mathbb{F}_2[X_1, \ldots, X_n]/(X_1^2 \oplus X_1, \ldots, X_n^2 \oplus X_n)$ in the following manner: $f(X_1, \ldots, X_n) = \bigoplus_{\boldsymbol{\alpha} \in \mathbb{F}_2^n} a_{\boldsymbol{\alpha}} \mathbf{X}^{\boldsymbol{\alpha}}$, where $\mathbf{X} = (X_1, \ldots, X_n)$; for $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_2^n$, $\mathbf{X}^{\boldsymbol{\alpha}}$ denotes the monomial $X_1^{\alpha_1} \cdots X_n^{\alpha_n}$; and $a_{\boldsymbol{\alpha}} \in \mathbb{F}_2$. The (algebraic) degree of f is $\mathsf{deg}(f) = \max\{\mathsf{wt}(\boldsymbol{\alpha}) : a_{\boldsymbol{\alpha}} = 1\}$; we adopt the convention that the zero function has degree 0. The degree (or sometimes also called the length) of the monomial $\mathbf{X}^{\boldsymbol{\alpha}}$ is $\mathsf{wt}(\boldsymbol{\alpha})$.

Functions of degree at most 1 are said to be affine functions. Affine functions with $a_{\mathbf{0}_n} = 0$ are said to be linear functions. Each $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_2^n$, defines the linear function $\langle \boldsymbol{\alpha}, \mathbf{X} \rangle = \langle \boldsymbol{\alpha}, (X_1, \ldots, X_n) \rangle = \alpha_1 X_1 \oplus \cdots \oplus \alpha_n X_n$. If $\mathsf{wt}(\boldsymbol{\alpha}) = w$, then the function $\langle \boldsymbol{\alpha}, \mathbf{X} \rangle$ is non-degenerate on exactly w of the n variables.

Nonlinearity and Walsh transform. The distance between two n-variable functions f and g is $d(f, g) = \#\{\mathbf{x} \in \mathbb{F}_2^n : f(\mathbf{x}) \neq g(\mathbf{x})\} = \mathsf{wt}(f \oplus g)$. The *nonlinearity* of an n-variable function f is defined to be $\mathsf{nl}(f) = \min_{\boldsymbol{\alpha} \in \mathbb{F}_2^n} \{d(f, \langle \boldsymbol{\alpha}, \mathbf{X} \rangle), d(f, 1 \oplus \langle \boldsymbol{\alpha}, \mathbf{X} \rangle)\}$, i.e. the nonlinearity of f is the minimum of the distances of f to all the affine functions. We define the *linear bias* of an n-variable function f to be $\mathsf{LB}(f) = 2(1/2 - \mathsf{nl}(f)/2^n)$. From a cryptographic point of view, the linear bias, rather than the nonlinearity, is of importance, since it is the linear bias which is used to quantify the resistance to (fast) correlation attacks.

The Walsh transform of an n-variable function f is the map $W_f : \mathbb{F}_2^n \to \mathbb{Z}$, where for $\boldsymbol{\alpha} \in \mathbb{F}_2^n$, $W_f(\boldsymbol{\alpha}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) \oplus \langle \boldsymbol{\alpha}, \mathbf{x} \rangle}$. From the definition it follows that $W_f(\boldsymbol{\alpha}) = 2^n - 2d(f, \langle \boldsymbol{\alpha}, \mathbf{X} \rangle)$ which shows $\mathsf{nl}(f) = 2^{n-1} - \frac{1}{2} \max_{\boldsymbol{\alpha} \in \mathbb{F}_2^n} |W_f(\boldsymbol{\alpha})|$.

Bias and correlation. The bias of a random bit b, denoted as $\mathsf{bias}(b)$, is defined to be $\mathsf{bias}(b) = 1 - 2\Pr[b = 1] \in [-1, 1]$. The correlation of an n-variable Boolean function f at a point $\boldsymbol{\alpha} \in \mathbb{F}_2^n$ is defined to be the correlation between $f(\mathbf{x})$ and $\langle \boldsymbol{\alpha}, \mathbf{X} \rangle$ and is given as $\mathsf{corr}_f(\boldsymbol{\alpha}) = W_f(\boldsymbol{\alpha})/2^n = 1 - 2\Pr[f(X) \neq \langle \boldsymbol{\alpha}, \mathbf{X} \rangle]$, where the probability is over the uniform random choice of X in $\mathbb{F}_2^n$; in particular, $\mathsf{corr}_f(\boldsymbol{\alpha})$ is equal to the bias of the random bit $f(\mathbf{X}) \oplus \langle \boldsymbol{\alpha}, \mathbf{X} \rangle$. Note that for all $\boldsymbol{\alpha} \in \mathbb{F}_2^n$, $|\mathsf{corr}_f(\boldsymbol{\alpha})| \leq \mathsf{LB}(f)$.

Bent functions. An n-variable function f is said to be bent [64] if $W_f(\boldsymbol{\alpha}) = \pm 2^{n/2}$ for all $\boldsymbol{\alpha} \in \mathbb{F}_2^n$. From the definition it follows that bent functions can exist only if n is even. An n-variable bent function has nonlinearity $2^{n-1} - 2^{n/2-1}$ (resp. linear bias $2^{-n/2}$), and this is the maximum possible nonlinearity (resp. least possible linear bias) that can be achieved by any n-variable function.

Resilient functions. Let n be a positive integer and m be an integer such that $0 \leq m < n$. An n-variable function f is said to be m-resilient if $W_f(\boldsymbol{\alpha}) = 0$ for all $\boldsymbol{\alpha}$ satisfying $\mathsf{wt}(\boldsymbol{\alpha}) \leq m$. Equivalently, f is m-resilient if and only if $d(f, \langle \boldsymbol{\alpha}, \mathbf{X} \rangle) = 2^{n-1}$ for $\boldsymbol{\alpha}$ satisfying $\mathsf{wt}(\boldsymbol{\alpha}) \leq m$, i.e. if the distance between f and any linear function which is non-degenerate on at most m variables is equal to 2^{n-1}. For notational convenience, by resiliency equal to -1 we mean a function that is not balanced.

Algebraic immunity. The algebraic immunity of an n-variable function f is defined [26, 62] as follows: $\mathsf{AI}(f) = \min_{g \neq 0}\{\mathsf{deg}(g) : \text{ either } gf = 0, \text{ or } g(f \oplus 1) = 0\}$. It is known [26] that $\mathsf{AI}(f) \leq \lceil n/2 \rceil$.

Direct sum. A simple way to construct a function is to add together two functions on disjoint sets of variables. The constructed function is called the direct sum of the two smaller functions. Let n_1 and n_2 be positive integers and g and h be functions on n_1 and n_2 variables respectively. Define

$$f(X_1, \ldots, X_{n_1}, Y_1, \ldots, Y_{n_2}) \quad = \quad g(X_1, \ldots, X_{n_1}) \oplus h(Y_1, \ldots, Y_{n_2}). \tag{1}$$

Basic properties of direct sum construction are given by the following result.

Proposition 1 (Proposition 1 of [70]) *Let f be constructed as in* (1)*. Then*

1. $\mathsf{deg}(f) = \max\{\mathsf{deg}(g), \mathsf{deg}(h)\}$.
2. $\mathsf{nl}(f) = 2^{n_1}\mathsf{nl}(h) + 2^{n_2}\mathsf{nl}(g) - 2\mathsf{nl}(g)\mathsf{nl}(h)$. *Equivalently,* $\mathsf{LB}(f) = \mathsf{LB}(g) \cdot \mathsf{LB}(h)$.
3. *f is balanced if and only if at least one of g or h is balanced.*
4. *If g is m_1-resilient and h is m_2-resilient, then f is m-resilient, where $m = \max\{m_1, m_2\}$.*

Bounds on the algebraic immunity of a function constructed as a direct sum is given by the following result.

Proposition 2 (Lemma 3 of [61]) *For f constructed as in* (1)*,* $\max\{\mathsf{AI}(g), \mathsf{AI}(h)\} \leq \mathsf{AI}(f) \leq \mathsf{AI}(g) + \mathsf{AI}(h)$.

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

$$(\pi, h)\text{-}\mathsf{MM}_{2k}(\mathbf{X}, \mathbf{Y}) \quad = \quad \langle \pi(\mathbf{X}), \mathbf{Y} \rangle \oplus h(\mathbf{X}). \tag{2}$$

Note that the degree of (π, h)-MM_{2k} is $\max(1 + \max_{1 \leq i \leq k} \mathsf{deg}(\pi_i), \mathsf{deg}(h))$, where $\pi_1, \ldots, \pi_k$ are the coordinate functions of π. The following was proved in [21].

Theorem 1 (Theorem 2 of [21]) *Let $k \geq 2$, $n = 2k$, and $\pi : \{0,1\}^k \to \{0,1\}^k$ be an affine map, i.e. each of the coordinate functions of π is an affine function of the input variables. Then* $\mathsf{AI}((\pi, h)\text{-}\mathsf{MM}_{2k}) \geq \mathsf{AI}(h)$.

Gate count. From an implementation point of view, it is of interest to obtain functions which are efficient to implement. A measure of implementation efficiency is the number of gates required to implement an n-variable function f. For the functions that we define, we will provide counts of NOT, XOR and AND gates. By i[N]+j[X]+k[A] gates we will mean i NOT, j XOR, and k AND gates.

3 Grain Family of Stream Ciphers: an Abstraction

An elegant approach to the construction of hardware friendly stream ciphers was adopted in the design of the Grain family of stream ciphers. In this section, we distill the essential ideas of these constructions to provide an abstract description of the Grain family of stream ciphers. The parameters are the following.

- $\kappa, v, \kappa_1 \geq \kappa, \kappa_2, a, n_0, n_1, p_0, p_1, q_0, q_1$ are non-negative integers, $n = n_0 + n_1$, $p = p_0 + p_1$, and $q = q_0 + q_1$;

- $\tau(x) = x^{\kappa_2} \oplus c_{\kappa_2-1}x^{\kappa_2-1} \oplus \cdots \oplus c_1 x \oplus 1$ is a primitive polynomial of degree κ_2 over $\mathbb{F}_2$, $A = (0) \cup (\kappa_2 - i : c_i = 1)$, and $a = \#A$;
- S_0, S_1, P_0, P_1 are lists with entries from $\{0, \ldots, \kappa_1 - 1\}$, and Q_0, Q_1 are lists with entries from $\{0, \ldots, \kappa_2 - 1\}$ such that no entry is repeated in any of the lists, and $\#S_0 = n_0$, $\#S_1 = n_1$, $\#P_0 = p_0$, $\#P_1 = p_1$, $\#Q_0 = q_0$ and $\#Q_1 = q_1$.
- g and h are n_0-variable and $(p_0 + q_0)$-variable Boolean functions respectively;
- ψ is a bit permutation of $p_0 + q_0$ bits;
- δ is a positive integer.

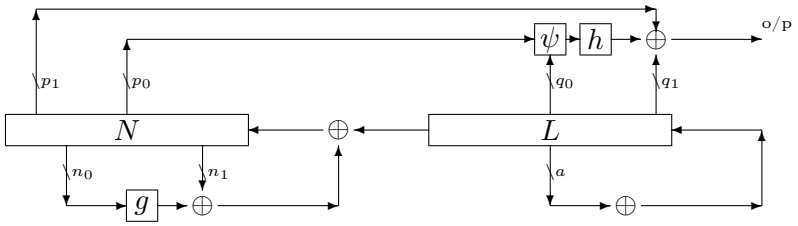

Figure 1: Schematic diagram of keystream generation by the Grain family of stream cipher.

Key, IV and state. A κ-bit key $K = (K_0, \ldots, K_{\kappa-1})$ and a v-bit initialisation vector $\mathsf{IV} = (V_0, \ldots, V_{v-1})$ are used. The state of the stream cipher consists of a κ_1-bit register $N = (\eta_0, \ldots, \eta_{\kappa_1-1})$, and a κ_2-bit register $L = (\lambda_0, \ldots, \lambda_{\kappa_2-1})$. The register N implements an NFSR, while the register L implements an LFSR. During initialisation, the κ-bit key will be loaded into N. The condition $\kappa_1 \geq \kappa$ ensures that this can be done.

Notation. We will use the following notation: given a list $T = (i_1, \ldots, i_t)$ of elements from $\{0, \ldots, \ell-1\}$ and $\mathbf{b} = (b_0, \ldots, b_{\ell-1}) \in \mathbb{F}_2^{\ell}$, we define $\mathsf{proj}(T, \mathbf{b}) = (b_{i_1}, \ldots, b_{i_t})$. Note that $i_1, \ldots, i_t$ are not necessarily in increasing (or decreasing) order. Given a list of bits $(b_1, \ldots, b_l)$, by $\mathsf{xor}(b_1, \ldots, b_l)$ we will denote the sum $b_1 \oplus \cdots \oplus b_l$.

Maps. The maps required to define the abstract Grain family of stream ciphers are the following.

$\mathsf{load}(K, \mathsf{IV}) : (K, \mathsf{IV}) \mapsto (N, L)$;
$\mathsf{NLB}(L) : L = (\lambda_0, \ldots, \lambda_{\kappa_2-1}) \mapsto \mathsf{xor}(\mathsf{proj}(A, L)) = \lambda_0 \oplus c_{\kappa_2-1}\lambda_1 \oplus \cdots \oplus c_1\lambda_{\kappa_2-1}$;
$\mathsf{NLS}(L) : L = (\lambda_0, \ldots, \lambda_{\kappa_2-1}) \mapsto (\lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2-1}, \mathsf{NLB}(L))$;
$\mathsf{NNB}(N) : N \mapsto \mathsf{xor}(\mathsf{proj}(S_1, N)) \oplus g(\mathsf{proj}(S_0, N))$;
$\mathsf{NNS}(N, L) : (N, L) = (\eta_0, \ldots, \eta_{\kappa_1-1}, \lambda_0, \ldots, \lambda_{\kappa_2-1}) \mapsto (\eta_1, \eta_2, \ldots, \eta_{\kappa_1-1}, \mathsf{NNB}(N) \oplus \lambda_0)$;
$\mathsf{NS}(N, L) : (N, L) \mapsto (\mathsf{NNS}(N, L), \mathsf{NLS}(L))$;
$\mathsf{OB}(N, L) : (N, L) \mapsto \mathsf{xor}(\mathsf{proj}(P_1, N), \mathsf{proj}(Q_1, L)) \oplus h(\psi(\mathsf{proj}(P_0, N), \mathsf{proj}(Q_0, L)))$;
$\mathsf{init}(N, L) : \{0,1\}^{\kappa_1+\kappa_2} \mapsto \{0,1\}^{\kappa_1+\kappa_2}$.

Explanations. NLB denotes 'next linear bit', NLS denotes 'next linear state', NNB denotes 'next nonlinear bit', NNS denotes 'next nonlinear state', NS denotes 'next state', OB denotes 'output bit', and init denotes 'initialise'.

Functions G and H. Let G and H be n-variable and $(p+q)$-variable functions respectively such that

$$\begin{aligned}\mathsf{NNB}(N) &= G(\mathsf{proj}(S_1, N), \mathsf{proj}(S_0, N)) \\ &= \mathsf{xor}(\mathsf{proj}(S_1, N)) \oplus g(\mathsf{proj}(S_0, N)) \quad (3)\\ \mathsf{OB}(N, L) &= H(\mathsf{proj}(Q_1, L), \mathsf{proj}(P_1, N), \mathsf{proj}(Q_0, L), \mathsf{proj}(P_0, N)) \\ &= \mathsf{xor}(\mathsf{proj}(Q_1, L), \mathsf{proj}(P_1, N)) + h(\psi(\mathsf{proj}(Q_0, L), \mathsf{proj}(P_0, N)))). \quad (4)\end{aligned}$$

Operation. Based on the above maps, the operation of the Grain family of stream ciphers is shown in Algorithm 1. A schematic diagram of the method of output generation (Steps 7 and 8 of Algorithm 1) is shown in Figure 1.

Algorithm 1 Operation of the Grain family of stream ciphers.

1: **function** $\mathsf{Grain}(K, \mathsf{IV})$
2: **initialisation phase:**
3: $(N, L) \leftarrow \mathsf{load}(K, \mathsf{IV})$
4: $(N, L) \leftarrow \mathsf{init}(N, L)$
5: **keystream generation phase:**
6: **for** $t \geq 0$ **do**
7: output $\mathsf{OB}(N, L)$
8: $(N, L) \leftarrow \mathsf{NS}(N, L)$
9: **end for**
10: **end function**.

At time point $t \geq 0$, let $(N^{(t)}, L^{(t)})$ be the state, where $(N^{(0)}, L^{(0)})$ is the state produced after the initialisation phase (Steps 3 and 4 of Algorithm 1). We write $N^{(t)} = (\eta_t, \eta_{t+1}, \ldots, \eta_{t+\kappa_1-1})$ and $L^{(t)} = (\lambda_t, \lambda_{t+1}, \ldots, \lambda_{t+\kappa_2-1})$. For $t \geq 0$, let the keystream bits that are produced from Step 7 of Algorithm 1 be denoted as $z_0, z_1, z_2, \ldots$. Then the loop in Steps 6 to 9 can be written as follows:

for $t \geq 0$ do
$z_t \leftarrow \mathsf{OB}(N^{(t)}, L^{(t)})$;
$(N^{(t+1)}, L^{(t+1)}) \leftarrow \mathsf{NS}(N^{(t)}, L^{(t)})$;
end for.

From the description of Algorithm 1 and the definitions of the various maps, we have

$$\begin{aligned}z_t = \mathsf{OB}(N^{(t)}, L^{(t)}) &= H(\mathsf{proj}(Q_1, L^{(t)}), \mathsf{proj}(P_1, N^{(t)}), \mathsf{proj}(Q_0, L^{(t)}), \mathsf{proj}(P_0, N^{(t)})) \\ &= \mathsf{xor}(\mathsf{proj}(Q_1, L^{(t)})) \oplus \mathsf{xor}(\mathsf{proj}(P_1, N^{(t)})) \oplus h(\psi(\mathsf{proj}(Q_0, L^{(t)}), \mathsf{proj}(P_0, N^{(t)}))), \quad (5)\\ \lambda_{t+\kappa_2} &= \mathsf{NLB}(L^{(t)}) = \mathsf{xor}(\mathsf{proj}(A, L^{(t)})) = c_1\lambda_{t+\kappa_2-1} \oplus c_2\lambda_{t+\kappa_2-2} \oplus \cdots \oplus c_{t+\kappa_2-1}\lambda_{t+1} \oplus \lambda_t, \quad (6)\\ \eta_{t+\kappa_1} &= \lambda_t \oplus \mathsf{NNB}(N^{(t)}) = G(\mathsf{proj}(S_1, N^{(t)}), \mathsf{proj}(S_0, N^{(t)})) \\ &= \lambda_t \oplus \mathsf{xor}(\mathsf{proj}(S_1, N^{(t)})) \oplus g(\mathsf{proj}(S_0, N^{(t)})). \quad (7)\end{aligned}$$

Tap positions for the feedback function of the LFSR. These are given by the set A which is determined by the polynomial $\tau(x)$. The number of tap positions is a.

Tap positions for the feedback function of the NFSR. The function G is the nonlinear feedback function of the nonlinear feedback shift register; n denotes the number of tap positions from N; S_0 and S_1 provide the indexes of the tap positions; the positions of N indexed by S_0 and S_1 (of sizes n_0 and n_1 respectively) are extracted (using $\mathsf{proj}(S_0, N)$ and $\mathsf{proj}(S_1, N)$ respectively) and fed into G to obtain the next nonlinear bit. The definition of G uses the "core" nonlinear function g whose input bits come from the bits of N which are indexed by S_0, and extends g by adding the bits of N which are indexed by S_1.

Proposition 3 *If $0 \in S_1$ and $0 \notin S_0$, then the map NS is invertible.*

Proof: The function NS maps the state (N, L) with $N = (\eta_0, \ldots, \eta_{\kappa_1-1})$ and $L = (\lambda_0, \ldots, \lambda_{\kappa_2-1})$ to the state $(N', L') = (\eta_1, \ldots, \eta_{\kappa_1-1}, b, \lambda_1, \ldots, \lambda_{\kappa_2-1}, b')$, where $b = \mathsf{NNB}(N) \oplus \lambda_0$ and $b' = \mathsf{NLB}(L)$. To show that NS is invertible, we argue that it is possible to recover (N, L) from (N', L'). Other than λ_0 and η_0, the other bits of (N, L) are also part of (N', L'). We show that it is possible to recover λ_0 and η_0 from (N', L').

Using $b' = \mathsf{NLB}(L) = \lambda_0 \oplus c_{\kappa_2-1}\lambda_1 \oplus \cdots \oplus c_1\lambda_{\kappa_2-1}$, we have $\lambda_0 = b' \oplus c_{\kappa_2-1}\lambda_1 \oplus \cdots \oplus c_1\lambda_{\kappa_2-1}$. So $\lambda_1, \ldots, \lambda_{\kappa_2-1}$ and b', determine the bit λ_0, i.e. λ_0 can be obtained from L'.

We have $\mathsf{NNB}(N) = \mathsf{xor}(\mathsf{proj}(S_1, N)) \oplus g(\mathsf{proj}(S_0, N))$. Since $0 \in S_1$ and $0 \notin S_0$ we may write $\mathsf{NNB}(N) = \eta_0 \oplus \eta'$, where $\eta' = \mathsf{xor}(\mathsf{proj}(S_1 \setminus \{0\}, N)) \oplus g(\mathsf{proj}(S_0, N \setminus \{0\}))$, i.e. η' is determined by the bits $\eta_1, \ldots, \eta_{\kappa_1-1}$, i.e. by N'. So $\eta_0 = \mathsf{NNB}(N) \oplus \eta' = b \oplus \lambda_0 \oplus \eta'$, from which it follows that η_0 is determined by b, λ_0 and η'. So the bits $b, b', \eta_1, \ldots, \eta_{\kappa_1-1}$ and $\lambda_1, \ldots, \lambda_{\kappa_2-1}$ determine the bit η_0. □

Remark 1 *The condition $0 \in S_1$ and $0 \notin S_0$ in Proposition 3 is fulfilled by both Grain v1 and Grain-128a.*

Tap positions for the output bit. The function H is the output function of the stream cipher. (In [1], H is called the pre-output function.) The input bits to H are obtained from both N and L. The lists P_0 and P_1 (of sizes p_0 and p_1 respectively) are the indexes of the tap positions from N, while the lists Q_0 and Q_1 (of sizes q_0 and q_1 respectively) are the indexes of the tap positions from L. The definition of H uses the "core" nonlinear function h whose inputs come from the bits of N and L indexed by P_0 and Q_0 respectively, and is extended by adding the bits of N and L indexed by P_1 and Q_1 respectively. The permutation ψ applies a bit permutation to the bits extracted from N and L based on P_0 and Q_0.

Initialisation. The function init performs the state initialisation. Two versions of init, namely init_1 and init_2 are shown in Algorithm 2. Both of these functions call the function NSI (which stands for 'next state during initialisation') which is defined below. Additionally, init_2 also calls the function NS. For all previously proposed variants of Grain, $\kappa_1 = \kappa_2 = \kappa$ and the initialisation functions were defined with this setting of parameters. The function init_1 was defined for Grain v1 and Grain-128a (and also for earlier variants), while the function init_2 was defined for Grain-128AEADv2 (and is a modification of another initialsation function that was proposed for an earlier version of Grain-128AEADv2, namely Grain-128AEAD). The function init_2 has three loops. The first loop of init_2 updates the state using the function NSI, while the third loop updates the state using NS (i.e. the next state function used for key generation). The second loop of init_2 runs for $\kappa/2$ iterations. In each iteration of the second loop, the state is updated using NSI and two bits of the key K are added to the most significant bits of N and L. This procedure is called key hardening and the idea of such key hardening was adopted from the design of the stream cipher Lizard [36]. Grain-128AEADv2 performs both encryption and authentication, where after the second loop of init_2 is completed, the registers used for authentication are initialised. The execution of the third loop of init_2 is done simultaneously with the initialisation of the authentication registers.

Algorithm 2 Descriptions of init_1 and init_2.

function $\mathsf{init}_1((N, L))$
  for $t \leftarrow 0$ to $2\max(\kappa_1, \kappa_2) - 1$ **do**
    $(N, L) \leftarrow \mathsf{NSI}(N, L)$
  end for
  return (N, L)
end function.
function $\mathsf{init}_2((N, L))$
  for $t \leftarrow 0$ to $5\max(\kappa_1, \kappa_2)/2 - 1$ **do**
    $(N, L) \leftarrow \mathsf{NSI}(N, L)$
  end for
  for $t \leftarrow 0$ to $\kappa/2 - 1$ **do**
    $(N, L) \leftarrow \mathsf{NSI}(N, L)$
    $(N, L) \leftarrow (N, L) \oplus (0^{\kappa_1 - 1}||K_t||0^{\kappa_2 - 1}||K_{t+\kappa/2})$
  end for
  for $t \leftarrow 0$ to $\max(\kappa_1, \kappa_2) - 1$ **do**
    $(N, L) \leftarrow \mathsf{NS}(N, L)$
  end for
  return (N, L)
end function.

The definition of NSI is as follows.

$$\begin{aligned}\mathsf{NSI}(N, L) : (N, L) = (\eta_0, \eta_1, \ldots, \eta_{\kappa_1 - 1}, \lambda_0, \lambda_1, \ldots, \lambda_{\kappa_2 - 1}) \\ \mapsto (\eta_1, \eta_2, \ldots, \eta_{\kappa_1 - 1}, b, \lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2 - 1}, b'),\end{aligned} \tag{8}$$

where $b = \mathsf{NNB}(N) \oplus \lambda_0 \oplus \mathsf{OB}(N, L)$ and $b' = \mathsf{NLB}(L) \oplus \mathsf{OB}(N, L)$.

The difference between NSI and the state update function NS (used during keystream generation) is that for NSI the output bit $\mathsf{OB}(N, L)$ is added to the most significant bits of both N and L.

Proposition 4 *If $0 \in S_1$, $0 \notin S_0$, $0 \notin P_0 \cup P_1$ and $0 \notin Q_0 \cup Q_1$, then the map NSI defined by (8) is invertible.*

Proof: The function NSI maps (N, L), with $N = (\eta_0, \ldots, \eta_{\kappa_1 - 1})$ and $L = (\lambda_0, \ldots, \lambda_{\kappa_2 - 1})$ to $(N', L') = (\eta_1, \eta_2, \ldots, \eta_{\kappa_1 - 1}, b, \lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2 - 1}, b')$, where $b = \mathsf{NNB}(N) \oplus \lambda_0 \oplus \mathsf{OB}(N, L)$ and $b' = \mathsf{NLB}(L) \oplus \mathsf{OB}(N, L)$.

Since $0 \notin P_0 \cup P_1$ and $0 \notin Q_0 \cup Q_1$, it follows that the bit $\mathsf{OB}(N, L)$ is uniquely determined by $\eta_1, \eta_2, \ldots, \eta_{\kappa_1 - 1}$ and $\lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2 - 1}$. So (N', L') uniquely defines $(N'', L'') = (\eta_1, \eta_2, \ldots, \eta_{\kappa_1 - 1}, \mathsf{NNB}(N) \oplus \lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2 - 1}, \mathsf{NLB}(L))$ which is the state $\mathsf{NS}(N, L)$. Now the argument in the proof of Proposition 3 shows that λ_0 and η_0 are uniquely defined from (N'', L''). □

Remark 2 *The conditions $0 \in S_1$, $0 \notin S_0$, $0 \notin P_0 \cup P_1$ and $0 \notin Q_0 \cup Q_1$ in Proposition 4 are fulfilled by both Grain v1 and Grain-128a.*

Rationale for init_2. Since both the functions NS and NSI are efficiently invertible, if the state of the stream cipher is obtained at some point of time, then the state can be retraced backwards to obtain the key. In other words, state recovery implies very efficient key recovery. Note that state recovery

itself implies that the stream cipher is broken. Nonetheless, a design goal for the Grain-128AEAD and Grain-128AEADv2 was that key recovery from state recovery should be difficult. This led to the design of the initialisation function init_2 where the secret key is re-introduced into the state after a certain number of iterations, and is based on an idea introduced in [36]. The point where the key may be safely reintroduced was analysed in details in [40] and was motivated by an earlier key-from-state recovery attack [25].

The parameter δ. The throughput can be increased by a factor of i for $i \leq \delta$ by replicating the state update maps and the output bit generation map i times. As a result, the number of clock cycles required for initialisation reduces by a factor of i, and during the output generation phase, i bits of the output are produced in each clock cycle. For simplicity of such an implementation, there should be no tap positions in the rightmost $\delta - 1$ bits of both N and L. Equivalently, all the entries of S_0, S_1, P_0 and P_1, Q_0 and Q_1 should be at most $\kappa_1 - \delta$, *and* all the entries of A, Q_0 and Q_1 should be at most $\kappa_2 - \delta$.

3.1 Grain v1 and Grain-128a

We show how the abstract Grain family can be instantiated to obtain Grain v1 and Grain-128a.

Grain v1. The values of the parameters are as follows.

1. $\kappa = \kappa_1 = \kappa_2 = 80$, $v = 64$, $a = 6$, $n_0 = 10$, $n_1 = 3$, $p_0 = 1$, $p_1 = 7$, $q_0 = 4$, $q_1 = 0$;
2. $\tau(x) = x^{80} \oplus x^{67} \oplus x^{57} \oplus x^{42} \oplus x^{29} \oplus x^{18} \oplus 1$, $A = (0, 13, 23, 38, 51, 62)$;
3. $S_0 = (9, 15, 21, 28, 33, 37, 45, 52, 60, 63)$, $S_1 = (0, 14, 62)$;
4. $P_0 = (63)$, $P_1 = (1, 2, 4, 10, 31, 43, 56)$, $Q_0 = (3, 25, 46, 64)$, $Q_1 = \emptyset$;
5. The functions G and H are defined as follows:

$$
\begin{aligned}
&G(Y_1, Y_2, Y_3, X_1, \ldots, X_{10}) = Y_1 \oplus Y_2 \oplus Y_3 \oplus g(X_1, \ldots, X_{10}), \text{ where} \\
&g(X_1, \ldots, X_{10}) \\
&\quad = \; X_1 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7 \oplus X_8 \oplus X_9 \\
&\qquad \oplus X_9X_{10} \oplus X_5X_6 \oplus X_1X_2 \oplus X_7X_8X_9 \oplus X_3X_4X_5 \\
&\qquad \oplus X_1X_4X_7X_{10} \oplus X_5X_6X_8X_9 \oplus X_2X_3X_9X_{10} \\
&\qquad \oplus X_6X_7X_8X_9X_{10} \oplus X_1X_2X_3X_4X_5 \oplus X_3X_4X_5X_6X_7X_8, \qquad (9) \\
&H(X_1, \ldots, X_7, Y_1, Y_2, Y_3, Y_4, Y_5) = X_1 \oplus \cdots \oplus X_7 \oplus h(Y_1, Y_2, Y_3, Y_4, Y_5), \text{ where} \\
&h(Y_1, Y_2, Y_3, Y_4, Y_5) \\
&\quad = \; Y_2 \oplus Y_5 \oplus Y_1Y_4 \oplus Y_3Y_4 \oplus Y_4Y_5 \oplus Y_1Y_2Y_3 \oplus Y_1Y_3Y_4 \oplus Y_1Y_3Y_5 \oplus Y_2Y_3Y_5 \oplus Y_3Y_4Y_5; \qquad (10)
\end{aligned}
$$

6. ψ is the identity permutation;
7. $\delta = 16$;
8. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||1^{16}$;
9. The function init_1 (see Algorithm 2) is used for state initialisation.

We rewrite (9) and (10) as follows.

$$
\begin{aligned}
&g(X_1, \ldots, X_{10}) \\
&\quad = \; X_3(1 \oplus X_4X_5(1 \oplus X_1X_2 \oplus X_6X_7X_8) \oplus X_2X_9X_{10}) \\
&\qquad \oplus X_6(1 \oplus X_5 \oplus X_8X_9(X_5 \oplus X_7X_{10})) \oplus X_7(1 \oplus X_8X_9 \oplus X_1X_4X_{10}) \\
&\qquad \oplus X_1(1 \oplus X_2) \oplus X_9(1 \oplus X_{10}) \oplus X_4 \oplus X_5 \oplus X_8, \qquad (11)
\end{aligned}
$$

$$
\begin{aligned}
&h(Y_1, Y_2, Y_3, Y_4, Y_5) \\
&\quad = \quad Y_2 \oplus Y_3 \oplus Y_4(Y_3(Y_1 \oplus Y_5 \oplus 1) \oplus Y_1 \oplus Y_5) + Y_5 Y_3(Y_1 \oplus Y_2) \oplus Y_1 Y_2 Y_3. \qquad (12)
\end{aligned}
$$

Using (11) it is possible to implement $g(X_1, \ldots, X_{10})$ using 6[N]+12[X]+17[A] gates. Using (12) and noting that $Y_1 \oplus Y_5$ occurs twice, it is possible to implement $h(Y_1, Y_2, Y_3, Y_4, Y_5)$ using 1[N]+7[X]+6[A] gates.

Grain-128a. The values of the parameters are as follows.

1. $\kappa = \kappa_1 = \kappa_2 = 128$, $v = 96$, $a = 6$, $n_0 = 24$, $n_1 = 5$, $p_0 = 2$, $p_1 = 7$, $q_0 = 7$, $q_1 = 1$;
2. $\tau(x) = x^{128} \oplus x^{121} \oplus x^{90} \oplus x^{58} \oplus x^{47} \oplus x^{32} \oplus 1$, $A = (0, 7, 38, 70, 81, 96)$;
3. $S_0 = (3, 67, 11, 13, 17, 18, 27, 59, 40, 48, 61, 65, 68, 84, 88, 92, 93, 95, 22, 24, 25, 70, 78, 82)$, $S_1 = (0, 26, 56, 91, 96)$;
4. $P_0 = (12, 95)$, $P_1 = (2, 15, 36, 45, 64, 73, 89)$, $Q_0 = (8, 13, 20, 42, 60, 79, 94)$, $Q_1 = (93)$.
5. The functions G and H are as follows:

$$
\begin{aligned}
&G(X_1, \ldots, X_5, Y_1, \ldots, Y_{14}, Z_1, \ldots, Z_{10}) \\
&\quad = \quad X_1 \oplus \cdots \oplus X_5 \oplus g(Y_1, \ldots, Y_{14}, Z_1, \ldots, Z_{10}), \text{ where} \qquad (13) \\
&g(Y_1, \ldots, Y_{14}, Z_1, \ldots, Z_{10}) \\
&\quad = \quad Y_1 Y_2 \oplus Y_3 Y_4 \oplus Y_5 Y_6 \oplus Y_7 Y_8 \oplus Y_9 Y_{10} \oplus Y_{11} Y_{12} \oplus Y_{13} Y_{14} \\
&\qquad \oplus Z_1 Z_2 Z_3 Z_4 \oplus Z_5 Z_6 Z_7 \oplus Z_8 Z_9 Z_{10}, \qquad (14) \\
&H(X_1, \ldots, X_8, Y_1, \ldots, Y_9) = X_1 \oplus \cdots \oplus X_8 \oplus h(Y_1, \ldots, Y_9), \text{ where} \\
&h(Y_1, \ldots, Y_9) = Y_1 Y_2 \oplus Y_3 Y_4 \oplus Y_5 Y_6 \oplus Y_7 Y_8 \oplus Y_1 Y_5 Y_9; \qquad (15)
\end{aligned}
$$

6. ψ is the following bit permutation of $p_0 + q_0 = 9$ bits:

$$
\psi(b_1, \ldots, b_9) \quad = \quad (b_8, b_1, b_2, b_3, b_9, b_4, b_5, b_6, b_7);
$$

7. $\delta = 32$;
8. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||1^{31}0$;
9. The function init_1 (see Algorithm 2) is used for state initialisation.

The function g given by (14) can be implemented using 9[X]+14[A] gates, while the function h given by (15) can be implemented using 4[X]+6[A] gates.

Remark 3 *The AEAD algorithm Grain-128AEADv2 is a variant of Grain-128a which uses the functions* init_2 *for state initialisation. The other parameters of Grain-128AEADv2 remain the same as that of Grain-128a.*

Cryptographic properties and gate counts. Table 2 (in Section 6) provides a summary of the cryptographic properties of the functions g, G, h and H for Grain v1 and Grain-128a. The gate counts for these functions are shown in Table 3 (also in Section 6). We note that the total gate count for Grain-128a is less than the total gate count for Grain v1.

4 Constructions of Boolean Functions

In this section, we provide constructions of certain Boolean functions which we will later use in obtaining new instantiations of the Grain family of stream ciphers. We start by noting the following function from [69].

Proposition 5 *[69] Let*

$$h_5(X_1, X_2, Z_1, Z_2, Z_3) = Z_1 \oplus Z_2 \oplus X_1(Z_1 \oplus Z_3) \oplus X_2(Z_2 \oplus Z_3) \oplus X_1X_2(Z_1 \oplus Z_2 \oplus Z_3). \quad (16)$$

The function h_5 is a 5-variable, 1-resilient function having degree 3, algebraic immunity 2, nonlinearity 12 (and hence linear bias equal to 2^{-2}), and can be implemented using 7[X]+4[A] gates.

Remark 4 *The cryptographic properties of h_5 given by* (16) *are the same as those of h in Grain v1 given by* (10) *(see Table 2). On the other hand, the function h_5 can be implemented using 7[X]+4[A], while the function h of Grain v1 can be implemented using 1[N]+7[X]+6[A] gates. So h_5 has a smaller gate count compared to h of Grain v1 while offering the same cryptographic properties.*

Proposition 6 *Let*

$$\begin{aligned} & h_7(X_1, X_2, X_3, Z_1, Z_2, Z_3, Z_4) \\ &= Z_1X_1X_2X_3 \oplus Z_1X_1X_2 \oplus Z_1X_2X_3 \oplus Z_1X_3 \oplus Z_1 \\ &\quad \oplus Z_2X_1X_2X_3 \oplus Z_2X_1 \oplus Z_2X_2X_3 \oplus Z_2X_2 \oplus Z_2 \\ &\quad \oplus Z_3X_1 \oplus Z_3X_2X_3 \oplus Z_4X_1X_2 \oplus Z_4X_2 \oplus Z_4X_3. \end{aligned} \quad (17)$$

The function h_7 is a 7-variable, 1-resilient function having degree 4, algebraic immunity 3, nonlinearity 56 (and hence linear bias equal to 2^{-3}), and can be implemented using 2[N]+9[X]+8[A] gates.

Proof: The function h_7 given by (17) can be viewed as a concatenation of 8 linear functions of Z_1, Z_2, Z_3 and Z_4 which are non-degenerate on 2 or 3 variables. From this it follows that h_7 is 1-resilient and has nonlinearity 56 (equivalently linear bias of 2^{-3}). It is easy to see that the degree of h_7 is 4. The algebraic immunity is obtained through direct computation.

For the gate count, we rewrite h_7 as follows.

$$\begin{aligned} & h_7(X_1, X_2, X_3, Z_1, Z_2, Z_3, Z_4) \\ &= Z_4(X_3 \oplus X_2(1 \oplus X_1)) \oplus Z_3(X_1 \oplus X_2X_3) \\ &\quad \oplus Z_2(1 \oplus X_1 \oplus X_2 \oplus X_2X_3(1 \oplus X_1)) \oplus Z_1(1 \oplus X_3 \oplus X_2X_3(1 \oplus X_1) \oplus X_1X_2) \end{aligned}$$

The following sequence of operations shows the number of bit operations (which provides the gate count) required for computing h_7.

$T_1 \leftarrow 1 \oplus X_1$; (1[N])
$T_2 \leftarrow X_1X_2$; $T_3 \leftarrow X_2X_3$; (2[A])
$T_4 \leftarrow X_2T_1$; $T_5 \leftarrow T_3T_1$; (2[A])
$Z_4(X_3 \oplus T_4) \oplus Z_3(X_1 \oplus T_3) \oplus Z_2(T_1 \oplus X_2 \oplus T_5) \oplus Z_1(1 \oplus X_3 \oplus T_5 \oplus T_2)$. (1[N]+9[X]+4[A])

□

The strategy used to construct h_7 (and also h_5) is to concatenate linear functions of at least 2 variables to construct 1-resilient functions. This is a well known strategy (introduced in [16]) and the obtained functions are called Maiorana-McFarland (resilient) functions. The same strategy can be extended to h_{2m+1}, for $m \geq 1$ to obtain 1-resilient functions with nonlinearity $2^{2m} - 2^m$ and degree $m + 1$. This method was proposed earlier in [20]. The maximum possible degree of an n-variable, 1-resilient function is $n - 2$ [74]. A previous work [70] proposed the construction of $(2m + 1)$-variable, 1-resilient functions, having degree $2m - 1$ and nonlinearity $2^{2m} - 2^m$. The important aspect of low gate count was not

considered in either of the previous works [70, 20]. The descriptions of the functions in [70, 20] are in general terms. We wrote out these descriptions for $n = 9$ and $n = 11$. It turns out that the gate counts are quite high making such functions unsuitable for implementation in stream ciphers targeted for low hardware footprint. To keep the gate count low, we adopt a different and simpler approach to the construction of 1-reslient functions on an odd number of variables. Compared to the previous constructions [70, 20], the simpler approach provides 1-resilient functions with the same linear bias, but lower degrees, and substantially smaller gate counts. Keeping practical implementation in mind, we find the trade-off between degree and low gate count to be acceptable. As the first step towards the construction of our desired family of 1-resilient functions, we note the following simple result.

Proposition 7 *For $k \geq 1$, we define $h_{2k}(U_1, \ldots, U_k, V_1, \ldots, V_k)$ as follows.*

$$h_{2k}(U_1, \ldots, U_k, V_1, \ldots, V_k) \;=\; U_1V_1 \oplus \cdots \oplus U_kV_k \oplus U_1U_2 \cdots U_k. \tag{18}$$

The function h_{2k} defined in (18) is an MM bent function with nonlinearity $2^{2k-1} - 2^{k-1}$ (equivalently linear bias 2^{-k}), and degree k, which is the highest possible degree for a $2k$-variable bent function. The function h_{2k} can be implemented using k XOR and $2k - 2$ AND gates.

Proof: The statement on the nonlinearity and degree is easy to see. By writing $U_kV_k \oplus U_1U_2 \cdots U_{k-1}U_k$ as $U_k(V_k \oplus U_1U_2 \cdots U_{k-1})$, we note that h_{2k} can be implemented using k XOR and $2k - 2$ AND gates. □

Proposition 8 *For $k \geq 1$, let h_{5+2k} be defined as follows.*

$$\begin{aligned} &h_{5+2k}(X_1, X_2, Z_1, Z_2, Z_3, U_1, \ldots, U_k, V_1, \ldots, V_k) \\ &\quad = \; h_5(X_1, X_2, Z_1, Z_2, Z_3) \oplus h_{2k}(U_1, \ldots, U_k, V_1, \ldots, V_k) \end{aligned} \tag{19}$$

The function h_{5+2k} is an n-variable, 1-resilient function, having degree $\max(3, k)$, and nonlinearity $2^{n-1} - 2^{(n-1)/2}$ (equivalently, linear bias $2^{-(n-1)/2}$), where $n = 5+2k$. Further, h_{5+2k} can be implemented using $7 + k$ XOR gates and $2k + 2$ AND gates.

Proof: The function h_{5+2k} defined in (19) is the direct sum of h_5 and h_{2k}. The statement about the resiliency and nonlinearity follows from the properties of h_5 and h_{2k} and Proposition 1. The gate count for h_{5+2k} follows from the gate counts of h_5 and h_{2k}. □

Remark 5 *For the MM functions in Proposition 7, from Theorem 1 we obtain the lower bound of 1 on the algebraic immunity. It is difficult to prove improved nontrivial lower bounds on the algebraic immunities of the functions in Proposition 7 and the same holds for the functions in Proposition 8. Instead, we computed the algebraic immunities of some of these functions. The algebraic immunities of h_8, h_{10}, h_{12} and h_{14} all turned out to be 3, and the algebraic immunities of h_{13}, h_{15}, h_{17} and h_{19} all turned out to be 4. In our new instantiations of the Grain family, we will need only h_{10}, h_{15} and h_{19}.*

The functions h constructed above will be used to instantiate the output functions of the new instances of the Grain family. Next we provide new constructions of functions g which will be used to instantiate the feedback function of the NFSR of the new instances of the Grain family.

The function g of Grain v1 given by (9), is a 10-variable, unbalanced function having degree 6, nonlinearity 430 (equivalently, linear bias $2^{-2.642}$) and algebraic immunity 4. A 10-variable bent function has nonlinearity 496 (equivalently linear bias 2^{-5}), however, its degree can be at most 5. By simply adding the monomial of degree 10 to a 10-variable MM bent function, one obtains a function with

nonlinearity 495 and degree 10; however, the function has algebraic immunity only 2. Our goal is to increase both the degree and the nonlinearity while retaining algebraic immunity to be 4. By starting with a 10-variable MM bent function, and quite a bit of experimentation, we identified a very interesting 10-variable function which is given in the following result.

Proposition 9 *Let*

$$\begin{aligned} & g_{10}(U_1, \ldots, U_5, V_1, \ldots, V_5) \\ &= U_1V_1 \oplus U_2V_2 \oplus U_3V_3 \oplus U_4V_4 \oplus U_5V_5 \oplus U_1U_2U_3U_4V_1V_2V_3 \oplus U_1U_2V_4V_5 \oplus U_3U_4V_5. \end{aligned} \tag{20}$$

Then g_{10} has degree 7, nonlinearity 492 (equivalently linear bias $2^{-4.678}$), and algebraic immunity 4. Further, g_{10} can be implemented using 1[N]+6[X]+10[A] gates.

Proof: The nonlinearity and algebraic immunity were obtained using direct computation. The degree is clearly 7. To see the gate count, we write

$$\begin{aligned} & g_{10}(U_1, \ldots, U_5, V_1, \ldots, V_5) \\ &= U_1V_1(1 \oplus (U_2V_2)(U_3V_3)U_4) \oplus U_2V_2 \oplus U_3V_3 \oplus V_4(U_4 \oplus U_1U_2V_5) \oplus V_5(U_5 \oplus U_3U_4). \end{aligned} \tag{21}$$

From (21) we observe that g_{10} can be implemented using 1[N]+6[X]+10[A] gates. □

The maximum possible algebraic immunity of any 10-variable function is 5 [26]. To obtain a 10-variable, degree 7 function, the nonlinearity of any 10-variable has to be lower than 496. Considering 10-variable, degree 7 functions we know of no other functions in the literature which has nonlinearity 492 and algebraic immunity at least 4. Further, the number of gates required to implement the function in (20) is quite small. The combination of low gate count and the very attractive cryptographic properties make the function g_{10} in (20) quite unique in the literature.

We also found the following interesting 10-variable function.

$$\begin{aligned} & f((U_1, \ldots, U_5, V_1, \ldots, V_5) \\ &= U_1V_1 \oplus U_2V_2 \oplus U_3V_3 \oplus U_4V_4 \oplus U_5V_5 \oplus U_1U_2U_3U_4V_1V_2V_3V_4 \oplus U_1U_2V_4V_5 \end{aligned} \tag{22}$$

which has nonlinearity 494 (equivalently linear bias $2^{-4.83}$), degree 8, and algebraic immunity 3. Compared to g_{10} given by (20), the function f given by (22) has higher degree and lower linear bias; however, it has algebraic immunity 3 which is one less than g_{10} (and the function g of Grain v1). We did not wish to reduce any of the security properties of the function g used in Grain v1, and that is why we opted to highlight g_{10} instead of f given by (22).

Remark 6 *Comparing g_{10} given by (20) to the function g of Grain v1 given by (9), we see that while both are unbalanced, 10-variable functions having algebraic immunity 4, g_{10} has higher degree and significantly higher nonlinearity (equivalently significantly lower linear bias). Further, g_{10} can be implemented using a smaller number of gates compared to g of Grain v1.*

In view of the above comment and Remark 4, if we replace the function h of Grain v1 by h_5 given by (16), and the function g of Grain v1 by g_{10} given by (20), then we obtain quantifiably better cryptographic properties and at the same time a lower gate count.

Generalised triangular function. Let $k > 1$ be a positive integer and k_1 be the largest positive integer such that $k_1(k_1+1)/2 \leq k$. Let $k_2 = k_1(k_1+1)/2$. Define the function $E_k : \mathbb{F}_2^k \to \mathbb{F}_2$ as follows.

$$E_k(X_1, \ldots, X_k) = X_1 \oplus X_2X_3 \oplus X_4X_5X_6 \oplus \cdots \oplus X_{k_2-2k_1+2} \cdots X_{k_2-k_1} \oplus X_{k_2-k_1+1} \cdots X_k. \tag{23}$$

Note that E_k is the direct sum of k_1 monomials, where the i-th monomial with $1 \leq i \leq k_1 - 1$, is of degree i, and the last monomial is of degree $k + k_1 - k_2$. For the case $k = k_2$, the function E_k was called a triangular function in [61]. Clearly, $E_k(X_1, \ldots, X_k)$ has degree $k + k_1 - k_2 \geq k_1$. Further, $E_k(X_1, \ldots, X_k)$ can be implemented using $k_1 - 1$ XOR gates, and $k - k_2 + k_1(k_1 - 1)/2$ AND gates. The following result follows as a consequence of Lemma 18 of [61].

Proposition 10 *Let $k > 1$ be a positive integer and k_1 be the largest positive integer such that $k_1(k_1 + 1)/2 \leq k$. For $E_k(X_1, \ldots, X_k)$ defined using* (23), $\mathsf{AI}(E_k) \geq k_1$.

Proposition 11 *For $k \geq 6$, define $g_{2k}(U_1, \ldots, U_k, V_1, \ldots, V_k)$ as follows.*

$$g_{2k}(U_1, \ldots, U_k, V_1, \ldots, V_k) \;=\; U_1V_1 \oplus \cdots \oplus U_kV_k \oplus E_k(U_1, \ldots, U_k). \tag{24}$$

The function g_{2k} given by (24) *is an MM bent function having nonlinearity $2^{2k-1} - 2^{k-1}$ (equivalently linear bias equal to 2^{-k}), and degree $k + k_1 - k_2$, where k_1 is the largest positive integer such that $k_1(k_1 + 1)/2 \leq k$, and $k_2 = k_1(k_1 + 1)/2$. Further, the algebraic immunity of g_{2k} is at least k_1.*

Proof: The statement on nonlinearity and degree is easy to see. From Theorem 1, the algebraic immunity of g_{2k} is at least that of E_k, and hence using Proposition 10, the algebraic immunity of g_{2k} is at least k_1. □

We highlight g_{24}, g_{30} and g_{36} since we will require these functions later.

$$\begin{aligned}
g_{24}(U_1, \ldots, U_{12}, V_1, \ldots, V_{12}) &= U_1V_1 \oplus \cdots \oplus U_{12}V_{12} \\
&\quad \oplus U_1 \oplus U_2U_3 \oplus U_4U_5U_6 \oplus U_7U_8U_9U_{10}U_{11}U_{12}, \qquad (25) \\
g_{30}(U_1, \ldots, U_{15}, V_1, \ldots, V_{15}) &= U_1V_1 \oplus \cdots \oplus U_{15}V_{15} \\
&\quad \oplus U_1 \oplus U_2U_3 \oplus U_4U_5U_6 \oplus U_7U_8U_9U_{10} \oplus U_{11}U_{12}U_{13}U_{14}U_{15} \qquad (26) \\
g_{36}(U_1, \ldots, U_{18}, V_1, \ldots, V_{18}) &= U_1V_1 \oplus \cdots \oplus U_{18}V_{18} \\
&\quad \oplus U_1 \oplus U_2U_3 \oplus U_4U_5U_6 \oplus U_7U_8U_9U_{10} \\
&\quad \oplus U_{11}U_{12}U_{13}U_{14}U_{15}U_{16}U_{17}U_{18}. \qquad (27)
\end{aligned}$$

From the above, the degrees of g_{24}, g_{30} and g_{36} are 6, 5, and 8 respectively. We write $g_{24}(U_1, \ldots, U_{12}, V_1, \ldots, V_{12})$ as follows.

$$\begin{aligned}
&g_{24}(U_1, \ldots, U_{12}, V_1, \ldots, V_{12}) \\
&= U_1(1 \oplus V_1) \oplus U_2(V_2 \oplus U_3) \oplus U_4(V_4 \oplus U_5U_6) \oplus U_7(V_7 \oplus U_8U_9U_{10}U_{11}U_{12}) \\
&\quad \oplus U_3V_3 \oplus U_5V_5 \oplus U_6V_6 \oplus U_8V_8 \oplus U_9V_9 \oplus U_{10}V_{10} \oplus U_{11}V_{11} \oplus U_{12}V_{12}.
\end{aligned}$$

The above rewriting of g_{24} shows that the function can be implemented using 1[N]+14[X]+17[A] gates. A similar rewriting of g_{30} and g_{36} show that these functions can be implemented using 1[N]+19[X]+21[A] gates and 1[N]+22[X]+27[A] gates respectively.

5 Design of New Components

Section 3 provided an abstract description of the Grain family of stream ciphers which identified the different components of the family. Section 4 presented new constructions of Boolean functions which can be used to instantiate the functions g and h of the Grain family. In this section, we provide new designs of some of the other components of the Grain family.

5.1 Choice of Tap Positions

The tap positions are determined by the lists $S_0, S_1,\ P_0, P_1,\ Q_0, Q_1$ and A. We impose the following conditions on the tap positions.

1. The lists S_0, S_1, P_0, P_1 are pairwise disjoint, which ensures that no position of the NFSR N is tapped twice. The lists A, Q_0, Q_1 are pairwise disjoint which ensures that no position of the LFSR L is tapped twice.
2. n_0 is even.
3. $0 \in S_1$.
4. $0 \notin P_0 \cup P_1$ and $0 \notin Q_0 \cup Q_1$ which ensures that the output bit $\mathsf{OB}(N, L)$ does not depend on either η_0 or λ_0.
5. All entries of S_0, S_1, P_0, P_1 (i.e. the tap positions of N) are at most $\kappa_1 - \delta$, and all entries of A, Q_0, Q_1 (i.e. the tap positions of L) are at most $\kappa_2 - \delta$. This allows speeding up initialisation and keystream generation by a factor of i for any $i \leq \delta$.
6. $\#(P_1 + S_0) = \#P_1 \cdot \#S_0$. This condition allows obtaining the upper bound $\mathsf{LB}(g)^{p_1}$ on the correlation of a direct sum of p_1 copies of g. See Remark 13 (given in Section 7.2) for an explanation of the usefulness of this upper bound in the context of the bias of a linear approximation of a sum of keystream bits.

The restrictions on the tap positions given above fulfill the conditions for the function NS to be invertible (see Proposition 3).

Remark 7 *We note that the first condition does not hold for either Grain v1, or Grain-128a. For Grain v1, 63 belongs to both S_0 and P_3, while for Grain-128a, 95 belongs to both S_0 and P_0.*

We next describe our method of choosing S_0, S_1, P_0 and P_1. First we define P_1 and S_0 as follows.

$$\left.\begin{array}{rcl} P_1 & = & (1, 2, 3, \ldots, p_1), \\ S_0 & = & (1+p_1, 1+2p_1, 1+3p_1, \ldots, 1+(n_0/2)p_1, \\ & & \quad 1+n_0p_1, 1+(n_0-1)p_1, 1+(n_0-2)p_1, \ldots, 1+(n_0/2+1)p_1). \end{array}\right\} \quad (28)$$

Since n_0 is even, the definition of S_0 is meaningful. The rationale for the ordering of S_0 is the following. In all our instantiations, g is either an MM bent function, or constructed from an MM bent function on n_0 variables. So there are $n_0/2$ quadratic terms. These $n_0/2$ quadratic terms are formed by multiplying the bit of N indexed by the i-th entry of S_0 with the bit of N indexed by the $(i + n/2)$-th entry of S_0, $i = 1, \ldots, n_0/2$. Reversing the second $n_0/2$ entries of S_0 ensures that shifts of N do not result in the cancellation of any quadratic term.

Proposition 12 *Suppose P_1 and S_0 are as defined in* (28)*. Then $\#(P_1 + S_0) = \#P_1 \cdot \#S_0$.*

Proof: Note that if we ignore ordering, then the elements of S_0 form the set $\{1 + ip_1 : 1 \leq i \leq n_0\}$. For any $j \in P_1 + S_0$, we argue that it can be written uniquely as $j = k + l$ with $k \in P_1$ and $l \in S_0$. Suppose there are $k_1, k_2 \in P_1$ and $l_1, l_2 \in S_0$ such that $k_1 + l_1 = k_2 + l_2$. Then $k_1 - k_2 = l_2 - l_1$. Let $l_1 = (p_1 + 1) + i_1p_1$ and $l_2 = (p_1 + 1) + i_2p_1$ for some $0 \leq i_1, i_2 \leq n_0 - 1$. So $k_1 - k_2 = l_2 - l_1 = p_1(i_2 - i_1)$ which implies that p_1 divides $k_1 - k_2$. If $k_1 \neq k_2$, then $|k_1 - k_2| < p_1$ which contradicts that $p_1 \mid k_1 - k_2$. $\square$

Remark 8 *The maximum element in S_0 is $1 + n_0p_1$. We require the condition $1 + n_0p_1 \leq \kappa_1 - \delta$ to ensure that no tap position of N is at a position greater than $\kappa_1 - \delta$.*

Our procedure for choosing the tap positions consists of the following steps.

1. Choose the primitive polynomial $\tau(x)$ such that all entries of A are at most $\kappa_2 - \delta$.
2. Choose the entries of Q_0 and Q_1 randomly from the set $\{0, \ldots, \kappa_2 - \delta\} \setminus A$.
3. Choose P_1 and S_0 as in (28).
4. Put 0 in S_1. Choose the entries of $S_1 \setminus \{0\}$, and the entries of P_0 randomly from the set $\{0, \ldots, \kappa_1 - 1\} \setminus (P_1 \cup S_0 \cup \{0\})$.

The choices in Steps 2 and 4 are of the above type. From the set $\{0, \ldots, \ell - 1\}$ choose a pair of disjoint sets of r_1 and r_2 elements avoiding a set of r elements. The strategy that we used is the following. In an array of length $\ell - r$, set the first r_1 positions to 1, the next r_2 positions to 2, and the rest of the elements of the array to 0. Next we perform the Fisher-Yates (see Algorithm P in Section 3.4.2 of [49]) random shuffle on the array a total of four times. (Fisher-Yates shuffle needs to be performed once to obtain a uniform random permutation; however, since our choice of the array index at each step is pseudo-random and not uniform random we perform the shuffle four times). After the shuffling is over, we insert the value 3 into the array at the positions of the r elements that are to be avoided. This step increases the length of the array from $\ell - r$ to ℓ. In this array of length ℓ, the positions marked by 1 give the subset of size r_1 and the positions marked by 2 give the subset of size r_2.

5.2 A New Initialisation Procedure

We define a new version of the next state function to be used during initialisation which we call NSIG. The definition of NSIG is as follows.

$$\begin{aligned} \mathsf{NSIG}(N, L) : (N, L) = (\eta_0, \eta_1, \ldots, \eta_{\kappa_1 - 1}, \lambda_0, \lambda_1, \ldots, \lambda_{\kappa_2 - 1}) \\ \mapsto (\eta_1, \eta_2, \ldots, \eta_{\kappa_1 - 1}, b, \lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2 - 1}, b'), \end{aligned} \tag{29}$$

where $b = \lambda_0 \oplus \mathsf{NNB}(N) \oplus \mathsf{OB}(N, L)$ and $b' = \mathsf{NLB}(L) \oplus b = \mathsf{NLB}(L) \oplus \lambda_0 \oplus \mathsf{NNB}(N) \oplus \mathsf{OB}(N, L)$.

Proposition 13 *If $0 \in S_1$, $0 \notin S_0$, $0 \notin P_0 \cup P_1$ and $0 \notin Q_0 \cup Q_1$, then the map NSIG defined by (29) is invertible.*

Proof: The function NSIG maps (N, L), with $N = (\eta_0, \ldots, \eta_{\kappa_1 - 1})$ and $L = (\lambda_0, \ldots, \lambda_{\kappa_2 - 1})$ to $(N', L') = (\eta_1, \eta_2, \ldots, \eta_{\kappa - 1}, b, \lambda_1, \lambda_2, \ldots, \lambda_{\kappa - 1}, b')$, where $b = \lambda_0 \oplus \mathsf{NNB}(N) \oplus \mathsf{OB}(N, L)$ and $b' = \mathsf{NLB}(L) \oplus b$.

Since $0 \notin P_0 \cup P_1$ and $0 \notin Q_0 \cup Q_1$, it follows that the bit $\mathsf{OB}(N, L)$ is determined by $\eta_1, \eta_2, \ldots, \eta_{\kappa_1 - 1}$ and $\lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2 - 1}$. Further, as in Proposition 3 since $0 \in S_1$ and $0 \notin S_0$, we may write $\mathsf{NNB}(N, L) = \eta_0 \oplus \eta'$, where η' is determined by $\eta_1, \eta_2, \ldots, \eta_{\kappa_1 - 1}$. Also, $\mathsf{NLB}(L) = \lambda_0 \oplus \lambda'$, where $\lambda' = c_{\kappa_2 - 1} \lambda_1 \oplus \cdots \oplus c_1 \lambda_{\kappa_2 - 1}$ is determined by $\lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2 - 1}$.

We have

$$\begin{aligned} b' &= b \oplus \mathsf{NLB}(L) \\ &= b \oplus \lambda_0 \oplus \lambda' \\ &= \mathsf{NNB}(N) \oplus \lambda_0 \oplus \mathsf{OB}(N, L) \oplus \lambda_0 \oplus \lambda' \\ &= \eta_0 \oplus \eta' \oplus \mathsf{OB}(N, L) \oplus \lambda', \end{aligned}$$

which shows that $\eta_0 = b' \oplus \eta' \oplus \mathsf{OB}(N, L) \oplus \lambda'$. So the bits b', $\lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2 - 1}$ and $\eta_1, \eta_2, \ldots, \eta_{\kappa_1 - 1}$ determine the bit η_0.

Next we write $b = \lambda_0 \oplus \mathsf{NNB}(N) \oplus \mathsf{OB}(N, L) = \eta_0 \oplus \eta' \oplus \lambda_0 \oplus \mathsf{OB}(N, L)$ which implies $\lambda_0 = b \oplus \eta_0 \oplus \eta' \oplus \mathsf{OB}(N, L)$. So the bits b, η_0, η', and $\mathsf{OB}(N, L)$ determine the bit λ_0. From the discussion in the previous paragraphy, it follows that λ_0 is determined by b, b', $\lambda_1, \lambda_2, \ldots, \lambda_{\kappa_2-1}$, and $\eta_1, \eta_2, \ldots, \eta_{\kappa_1-1}$. □

Remark 9 *Our choice of tap positions described in Section 5.1 fulfills the conditions of Proposition 13.*

Difference between NSI given by (8) **and NSIG given by** (29)**.** Both NSI and NSIG provide nonlinear feedbacks to the most significant bits of both N and L. The feedbacks provided by both NSI and NSIG to the most significant bit of N are the same. The difference between NSI and NSIG is in the type of feedback provided to the most significant bit of L; NSI provides the feedback $\mathsf{NLB}(L) \oplus \mathsf{OB}(N, L)$, while NSIG provides the feedback $\lambda_0 \oplus \mathsf{NLB}(L) \oplus \mathsf{OB}(N, L) \oplus \mathsf{NNB}(N)$. Since $\mathsf{NLB}(L)$ is a linear function of L, the nonlinear part of the feedback to L by NSI is $\mathsf{OB}(N, L)$, whereas the nonlinear part of the feedback to L by NSIG is $\mathsf{OB}(N, L) \oplus \mathsf{NNB}(N)$.

The rationale for introducing NSIG is the following. During initialisation, the key K is loaded to the register N while the padded IV is loaded to the register L. Since the IV is known, the only unknown is the secret key K. A goal of the initialisation phase is that at the end of this phase all the bits of both N and L depend in a complex nonlinear manner on the unknown bits of the secret key K. In NSI, the output of the function OB is added to the most significant bit of L, while in NSIG, along with the output of OB, the output of the function NNB is also added to the most significant bit of L. The function OB depends nonlinearly on p_0 bits of K while the function NNB depends nonlinearly on n_0 bits of K. Further, the nonlinear function h is used to determine OB while the nonlinear function g is used to determine $\mathsf{NNB}(N)$. In the proposals Grain v1, Grain-128a, and also our new proposals given below, the condition $p_0 < n_0$ holds (see Table 1). Further, in all cases, h is a much smaller function than g both in terms of the number of variables and algebraic degree (see Table 2). Due to the introduction of $\mathsf{NNB}(N)$ in the feedback to L, compared to the function NSI, the function NSIG achieves a more complex nonlinear injection of the bits of K to the register L.

Remark 10 *In* (29)*, we have* $b' = \mathsf{NLB}(L) \oplus \lambda_0 \oplus \mathsf{NNB}(N) \oplus \mathsf{OB}(N, L)$*. The addition of* λ_0 *is important. If instead we had defined* $b' = \mathsf{NLB}(L) \oplus \mathsf{NNB}(N) \oplus \mathsf{OB}(N, L)$*, then the corresponding function* NSIG *would not have been invertible.*

The function initG. We define the function $\mathsf{initG} : \{0,1\}^{\kappa_1} \times \{0,1\}^{\kappa_2} \to \{0,1\}^{\kappa_1} \times \{0,1\}^{\kappa_2}$ to be the function init_1 given by Algorithm 2 where the call to the function NSI is replaced by a call to the function NSIG.

Remark 11 *Since* NS *and* NSIG *are both efficiently invertible functions, recovery of the state at any point of time leads to recovering the key efficiently. This is similar to the situation for* init_1 *used with the function* NSI *and has been discussed earlier in Section 3. As has been mentioned earlier, state recovery using less than* 2^κ *operations implies that the stream cipher is broken. If it is desired that the key recovery should be difficult from state recovery, then one may use the initialsation function* init_2 *with the call to* NSI *replaced by the call to* NSIG*.*

The function load(K, IV). The κ-bit key K and the v-bit IV are used to load the state (N, L) which is of length $\kappa_1 + \kappa_2$. In all our instantiations, we have $\kappa_1 + \kappa_2 - (\kappa + v)$ to be even. We define the function $\mathsf{load}(K, \mathsf{IV})$ to output $K||\mathsf{IV}||(01)^c$, where $c = (\kappa_1 + \kappa_2 - (\kappa + v))/2$. Since $\kappa_1 \geq \kappa$, the key K is loaded completely into N, the remaining bits of N and the most significant bits of L are loaded with IV,

and finally the left over bits of L are padded with the balanced string $(10)^c$. In all our instantiations, we have $\kappa_1 = \kappa$, so N is completely filled with K.

An early analysis [50] had pointed out problems if the padding of L is done by either the all-zero or the all-one string. In response, Grain-128a [1] had defined the padding to be the all-zero string followed by a single 1, and had mentioned that this padding rule rendered the attacks reported in [50, 17, 52] inapplicable. Our proposal of using a balanced string also has the effect of rule out these attacks. A recent work [58] performed a more systematic study of the padding rule in the load function. This study does not show any attack against using the balanced string for padding.

5.3 Choice of the Permutation ψ

We instantiate the function h (which is of $p_0 + q_0$ variables) with either of h_7 (given by (17)), h_{10} (given by (18)), h_{15} or h_{19} (given by (19)). The inputs to these functions come from L and N with tap positions determined by Q_0 and P_0 respectively. The number of tap positions from L is p_0 and the number of tap positions from N is q_0. Recall that in the definition of the output bit function, the permutation ψ is first applied to $(\mathsf{proj}(P_0, N), \mathsf{proj}(Q_0, L))$ and then the function h is applied, i.e. the computation required is $h(\psi(\mathsf{proj}(P_0, N), \mathsf{proj}(Q_0, L)))$. This requires specifying the permutation ψ. For $q_0 = p_0 + 1$, we define the following permutation.

$$\psi_{p_0+q_0}(a_1, a_2, \ldots, a_{p_0}, b_1, b_2, \ldots, b_{q_0}) = (\underbrace{b_1, a_1, b_2, a_2, b_3}, \underbrace{a_3,\ a_4, \ldots, a_{p_0}}, \underbrace{b_4, b_5, \ldots, b_{q_0}}). \quad (30)$$

The rationale for defining the above permutation is the following. Consider h_{15}. From (19),

$$h_{15}(X_1, X_2, Z_1, Z_2, Z_3, U_1, \ldots, U_5, V_1, \ldots, V_5) = h_5(X_1, X_2, Z_1, Z_2, Z_3) \oplus h_{10}(U_1, \ldots, U_5, V_1, \ldots, V_5).$$

For $p_0 = 7$ and $q_0 = 8$, suppose $P_0 = (i_1, \ldots, i_7)$ and $Q_0 = (j_1, \ldots, j_8)$. Then $\mathsf{proj}(P_0, N) = (\eta_{i_1}, \ldots, \eta_{i_7})$ and $\mathsf{proj}(Q_0, L) = (\lambda_{j_1}, \ldots, \lambda_{j_8})$ and

$$\begin{aligned} h_{15}(\psi_{15}(\eta_{i_1}, \ldots, \eta_{i_7}, \lambda_{j_1}, \ldots, \lambda_{j_8})) &= h_{15}(\lambda_{j_1}, \eta_{i_1}, \lambda_{j_2}, \eta_{i_2}, \lambda_{j_3}, \eta_{i_3}, \eta_{i_4}, \ldots, \eta_{i_7}, \lambda_{j_4}, \lambda_{j_5}, \ldots, \lambda_{j_8}) \\ &= h_5(\lambda_{j_1}, \eta_{i_1}, \lambda_{j_2}, \eta_{i_2}, \lambda_{j_3}) \oplus h_{10}(\eta_{i_3}, \eta_{i_4}, \ldots, \eta_{i_7}, \lambda_{j_4}, \lambda_{j_5}, \ldots, \lambda_{j_8}). \end{aligned}$$

The above ensures that the quadratic terms in the definition of h_{10} given by (18) consist of one element from N and one element from L. Further, there is a good mix of the λ and η terms in the definition of h_5 given by (16). For any $q_0 = 3 + k$ and $p_0 = 2 + k$, we obtain a mix of the λ and η terms in the composition $h_{p_0+q_0} \circ \psi_{p_0+q_0}$ which is similar to the mix illustrated above for $q_0 = 8$ and $p_0 = 7$.

5.4 Sizes of the NFSR and the LFSR

The NFSR N is of length κ_1 bits and the LFSR L is of length κ_2 bits. The initialisation procedure loads the κ-bit secret key K into N, which necessitates $\kappa_1 \geq \kappa$. The total size of the state is $\kappa_1 + \kappa_2$. Since the feedback polynomial of the LFSR is primitive, if the LFSR is loaded with a nonzero value, then the combined state of NFSR and LFSR takes distinct values for $2^{\kappa_2} - 1$ time steps. If at most 2^B bits of keystream are to be generated from a single pair of secret key and IV, then setting $2^{\kappa_2} > 2^B$ (equivalently, $\kappa_2 > B$) ensures that the combined state is not repeated while generating the 2^B bits of the keystream.

There are several possibilities for choosing $\kappa_1 \geq \kappa$ and $\kappa_2 > B$ which can be divided into two broad categories depending on whether $\kappa_1 + \kappa_2 \leq 2\kappa$ or $\kappa_1 + \kappa_2 > 2\kappa$. The various options are given below

Category 1. $\kappa_1 + \kappa_2 \leq 2\kappa$.

	κ	v	κ_1	κ_2	$\kappa_1+\kappa_2$	a	n_0	n_1	p_0	p_1	q_0	q_1	p_0+q_0	δ
Grain v1	80	64	80	80	160	6	10	3	1	7	4	0	5	16
Grain-128a	128	96	128	128	256	6	24	5	2	7	7	1	9	32
R-80	80	64	80	80	160	6	10	3	3	6	4	1	7	16
R-128	128	96	128	128	256	6	24	5	5	4	5	4	10	31
W-128	128	96	128	112	240	6	24	5	5	4	5	4	10	31
R-192	192	128	192	192	384	6	30	7	7	5	8	5	15	32
W-192	192	128	192	160	352	6	30	7	7	5	8	5	15	32
R-256	256	128	256	256	512	6	36	9	9	6	10	6	19	32
W-256	256	128	256	208	464	6	36	9	9	6	10	6	19	32

Table 1: Values of the parameters for Grain v1, Grain-128a, and the new instantiations.

Option 1. $\kappa_1 = \kappa_2 = \kappa$, $\kappa_1 + \kappa_2 = 2\kappa$.
Option 2. $\kappa_1 = \kappa$, $\kappa_2 < \kappa$, $\kappa_1 + \kappa_2 < 2\kappa$.
Option 3. $\kappa_1 > \kappa$, $\kappa_2 < \kappa$, $\kappa_1 + \kappa_2 \leq 2\kappa$.

Category 2. $\kappa_1 + \kappa_2 > 2\kappa$.

Option 1 was followed by all previous proposals of the Grain family. Properly chosen proposals based on Option 2 will be smaller than proposals based on Option 1. Option 3 provides for the NFSR to be of length greater than κ. This allows using bigger nonlinear feedback functions. So proposals for Option 3 can potentially provide higher security than proposals for Option 1 and Option 2 without varying the size of the hardware too much. Appropriate proposals for Category 2 can potentially provide higher security though the increase in the size of the state will increase the size of the stream cipher.

In the next section, we put forward four new proposals at the 80-bit, 128-bit, 192-bit and 256-bit security levels based on Option 1. We also put forward three new proposals based on Option 2 at the 128-bit, 192-bit and the 256-bit security levels. The methodology that we have developed in putting forward the new proposals can also be applied to obtain new proposals following Option 3 and Category 2. We leave these as future tasks.

6 New Concrete Proposals of the Grain Family

We put forward seven new proposals of the Grain family targeted at four different security levels, namely 80-bit, 128-bit, 192-bit and 256-bit security levels. The proposals are named as R or W with the appropriate security level attached. As members of the Grain family, 'R' may be taken to stand for 'Rice' and 'W' may be taken to stand for 'Wheat'. The parameters of the new proposals are given in Table 1. The tap positions, the feedback functions, the permutation ψ, and the function load are given below.

IV size. For the 80-bit and the 128-bit security levels following Grain v1 and Grain-128a, we propose the IV sizes to be 64 bits and 96 bits respectively. For the 192-bit and the 256-bit security levels, we propose the IV sizes to be 128-bit and 192-bit respectively.

R-80.

1. $\tau(x) = x^{80} \oplus x^{77} \oplus x^{65} \oplus x^{29} \oplus x^{19} \oplus x^{16} \oplus 1$, $A = (0, 3, 15, 51, 61, 64)$;
2. $S_0 = (7, 13, 19, 25, 31,\ 61, 55, 49, 43, 37)$, $S_1 = (0, 54, 57)$;

3. $P_0 = (15, 16, 39)$, $P_1 = (1, 2, 3, 4, 5, 6)$, $Q_0 = (5, 12, 16, 19)$, $Q_1 = (11)$.
4. The functions g and h are as follows.

$$\begin{aligned} g(X_1, \ldots, X_{10}) &= g_{10}(X_1, \ldots, X_{10}), \\ h(X_1, X_2, X_3, Z_1, Z_2, Z_3, Z_4) &= h_7(X_1, X_2, X_3, Z_1, Z_2, Z_3, Z_4), \end{aligned}$$

   where g_{10} is the function given in (20), and h_7 is the function given in (17).
5. ψ is the permutation ψ_7 given by (30), with 7 written as $7 = p_0 + q_0 = 3 + 4$
6. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||(10)^8$.
7. The function initG is used for state initialisation.

R-128.

1. $\tau(x) = x^{128} \oplus x^{108} \oplus x^{97} \oplus x^{54} \oplus x^{46} \oplus x^{32} \oplus 1$, $A = (0, 20, 31, 74, 82, 96)$;
2. $S_0 = (5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49,\ 97, 93, 89, 85, 81, 77, 73, 69, 65, 61, 57, 53)$;
3. $S_1 = (0, 36, 55, 71, 91)$;
4. $P_0 = (6, 31, 39, 50, 67)$, $P_1 = (1, 2, 3, 4)$, $Q_0 = (1, 12, 38, 87, 97)$, $Q_1 = (5, 10, 30, 85)$;
5. The functions g and h are as follows.

$$\begin{aligned} g(U_1, \ldots, U_{12}, V_1, \ldots, V_{12}) &= g_{24}(U_1, \ldots, U_{12}, V_1, \ldots, V_{12}) \\ h(U_1, \ldots, U_5, V_1, \ldots, V_5) &= h_{10}(U_1, \ldots, U_5, V_1, \ldots, V_5), \end{aligned}$$

   where g_{24} is given by (25) and h_{10} is given by (18).
6. ψ is the identity permutation.
7. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||(10)^{32}$.
8. The function initG is used for state initialisation.

W-128.

1. $\tau(x) = x^{112} \oplus x^{93} \oplus x^{84} \oplus x^{74} \oplus x^{43} \oplus x^{32} \oplus 1$, $A = (0, 19, 28, 38, 69, 80)$;
2. $S_0 = (5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49,\ 97, 93, 89, 85, 81, 77, 73, 69, 65, 61, 57, 53)$;
3. $S_1 = (0, 28, 54, 67, 68)$;
4. $P_0 = (11, 26, 30, 44, 76)$, $P_1 = (1, 2, 3, 4)$, $Q_0 = (11, 36, 56, 73, 76)$, $Q_1 = (13, 31, 39, 77)$;
5. The functions g and h are as follows.

$$\begin{aligned} g(U_1, \ldots, U_{12}, V_1, \ldots, V_{12}) &= g_{24}(U_1, \ldots, U_{12}, V_1, \ldots, V_{12}) \\ h(U_1, \ldots, U_5, V_1, \ldots, V_3) &= h_{10}(U_1, \ldots, U_5, V_1, \ldots, V_3), \end{aligned}$$

   where g_{24} is given by (25) and h_{10} is given by (18).
6. ψ is the identity permutation.
7. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||(10)^8$.
8. The function initG is used for state initialisation.

R-192.

1. $\tau(x) = x^{192} \oplus x^{131} \oplus x^{123} \oplus x^{118} \oplus x^{79} \oplus x^{32} \oplus 1$, $A = (0, 61, 69, 74, 113, 160)$;
2. $S_0 = (6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56, 61, 66, 71, 76,$
   $151, 146, 141, 136, 131, 126, 121, 116, 111, 106, 101, 96, 91, 86, 81)$;
   $S_1 = (0, 22, 68, 75, 82, 89, 129)$;
3. $P_0 = (35, 69, 83, 88, 98, 104, 150)$, $P_1 = (1, 2, 3, 4, 5)$,
   $Q_0 = (1, 26, 57, 77, 83, 103, 116, 127)$, $Q_1 = (60, 75, 101, 122, 123)$;

4. The functions g and h are as follows.

$$\begin{aligned} g(U_1,\ldots,U_{15},V_1,\ldots,V_{15}) &= g_{30}(U_1,\ldots,U_{15},V_1,\ldots,V_{15}) \\ h(X_1,X_2,Z_1,Z_2,Z_3,U_1,\ldots,U_5,V_1,\ldots,V_5) &= h_{15}(X_1,X_2,Z_1,Z_2,Z_3,U_1,\ldots,U_5,V_1,\ldots,V_5), \end{aligned}$$

   where g_{30} is given by (26) and h_{15} is given by (19).
5. ψ is the permutation ψ_{15} given by (30), with 15 written as $15 = p_0 + q_0 = 7 + 8$.
6. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||(10)^{32}$.
7. The function initG is used for state initialisation.

W-192.

1. $\tau(x) = x^{160} \oplus x^{142} \oplus x^{76} \oplus x^{57} \oplus x^{44} \oplus x^{32} \oplus 1$, $A = (0, 18, 84, 103, 116, 128)$;
2. $S_0 = (6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56, 61, 66, 71, 76,$
   $151, 146, 141, 136, 131, 126, 121, 116, 111, 106, 101, 96, 91, 86, 81)$;
   $S_1 = (0, 43, 53, 72, 75, 123, 140)$;
3. $P_0 = (30, 54, 58, 80, 112, 156, 160)$, $P_1 = (1, 2, 3, 4, 5)$,
   $Q_0 = (10, 43, 51, 91, 96, 110, 111, 127)$, $Q_1 = (8, 26, 108, 113, 115)$;
4. The functions g and h are as follows.

$$\begin{aligned} g(U_1,\ldots,U_{15},V_1,\ldots,V_{15}) &= g_{30}(U_1,\ldots,U_{15},V_1,\ldots,V_{15}) \\ h(X_1,X_2,Z_1,Z_2,Z_3,U_1,\ldots,U_5,V_1,\ldots,V_5) &= h_{15}(X_1,X_2,Z_1,Z_2,Z_3,U_1,\ldots,U_5,V_1,\ldots,V_5), \end{aligned}$$

   where g_{30} is given by (26) and h_{15} is given by (19).
5. ψ is the permutation ψ_{15} given by (30), with 15 written as $15 = p_0 + q_0 = 7 + 8$.
6. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||(10)^{16}$.
7. The function initG is used for state initialisation.

R-256. The values of the parameters are as follows.

1. $\tau(x) = x^{256} \oplus x^{203} \oplus x^{138} \oplus x^{76} \oplus x^{46} \oplus x^{32} \oplus 1$, $A = (0, 53, 118, 180, 210, 224)$;
2. $S_0 = (7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 91, 97, 103, 109,$
   $217, 211, 205, 199, 193, 187, 181, 175, 169, 163, 157, 151, 145, 139, 133, 127, 121, 115)$;
   $S_1 = (0, 16, 26, 83, 84, 92, 134, 160, 192)$;
3. $P_0 = (8, 74, 99, 131, 135, 136, 144, 189, 218)$, $P_1 = (1, 2, 3, 4, 5, 6)$;
4. $Q_0 = (1, 11, 61, 110, 131, 133, 170, 198, 208, 218)$, $Q_1 = (66, 74, 90, 97, 124, 193)$;
5. The functions g and h are as follows.

$$\begin{aligned} g(U_1,\ldots,U_{18},V_1,\ldots,V_{18}) &= g_{36}(U_1,\ldots,U_{18},V_1,\ldots,V_{18}) \\ h(X_1,X_2,Z_1,Z_2,Z_3,U_1,\ldots,U_7,V_1,\ldots,V_7) &= h_{19}(X_1,X_2,Z_1,Z_2,Z_3,U_1,\ldots,U_7,V_1,\ldots,V_7), \end{aligned}$$

   where g_{36} is given by (27) and h_{19} is given by (19).
6. ψ is the permutation ψ_{19} given by (30) with 19 written as $19 = p_0 + q_0 = 9 + 10$.
7. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||(10)^{32}$.
8. The function initG is used for state initialisation.

W-256.

1. $\tau(x) = x^{208} \oplus x^{169} \oplus x^{164} \oplus x^{114} \oplus x^{35} \oplus x^{32} \oplus 1$, $A = (0, 39, 44, 94, 173, 176)$;

		var	res	deg	AI	nl	LB
Grain v1	g	10	-1	6	4	430	$2^{-2.642}$
	G	13	2	6	4	$2^3 \cdot 430$	$2^{-2.642}$
	h	5	1	3	2	12	2^{-2}
	H	12	8	3	3	$2^7 \cdot 12$	2^{-2}
Grain-128a	g	24	-1	4	≤ 4	8356352	$2^{-8.023}$
	G	29	4	4	≤ 4	$2^5 \cdot 8356352$	$2^{-8.023}$
	h	9	-1	3	3	240	2^{-4}
	H	17	7	3	3	$2^8 \cdot 240$	2^{-4}
R-80	g	10	-1	7	4	492	$2^{-4.678}$
	G	13	2	7	4	$2^3 \cdot 492$	$2^{-4.678}$
	h	7	1	4	3	56	2^{-3}
	H	14	8	3	3	$2^7 \cdot 56$	2^{-3}
R-128 W-128	g	24	-1	6	≥ 4	$2^{23} - 2^{11}$	2^{-12}
	G	29	4	6	≥ 4	$2^{28} - 2^{16}$	2^{-12}
	h	10	-1	5	3	496	2^{-5}
	H	18	7	5	3	$2^8 \cdot 496$	2^{-5}
R-192 W-192	g	30	-1	5	≥ 5	$2^{29} - 2^{14}$	2^{-15}
	G	37	6	5	≥ 5	$2^{35} - 2^{23}$	2^{-15}
	h	15	1	5	4	$2^{14} - 2^7$	2^{-7}
	H	25	11	5	≥ 4	$2^{24} - 2^{17}$	2^{-7}
R-256 W-256	g	36	-1	8	≥ 6	$2^{35} - 2^{17}$	2^{-18}
	G	45	8	8	≥ 6	$2^{44} - 2^{26}$	2^{-18}
	h	19	1	7	4	$2^{18} - 2^9$	2^{-9}
	H	31	13	7	≥ 4	$2^{30} - 2^{21}$	2^{-9}

Table 2: Comparison of cryptographic properties of the nonlinear functions used in the different proposals.

2. $S_0 = (7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 91, 97, 103, 109,$
   $217, 211, 205, 199, 193, 187, 181, 175, 169, 163, 157, 151, 145, 139, 133, 127, 121, 115)$;
   $S_1 = (0, 17, 38, 41, 89, 132, 146, 186, 190)$;
3. $P_0 = (8, 72, 75, 99, 128, 176, 188, 212, 215)$, $P_1 = (1, 2, 3, 4, 5, 6)$;
4. $Q_0 = (22, 53, 54, 73, 82, 86, 99, 143, 148, 167)$, $Q_1 = (8, 70, 118, 151, 157, 171)$;
5. The functions g and h are as follows.

$$\begin{aligned} g(U_1, \ldots, U_{18}, V_1, \ldots, V_{18}) &= g_{36}(U_1, \ldots, U_{18}, V_1, \ldots, V_{18}) \\ h(X_1, X_2, Z_1, Z_2, Z_3, U_1, \ldots, U_7, V_1, \ldots, V_7) &= h_{19}(X_1, X_2, Z_1, Z_2, Z_3, U_1, \ldots, U_7, V_1, \ldots, V_7), \end{aligned}$$

   where g_{36} is given by (27) and h_{19} is given by (19).
6. ψ is the permutation ψ_{19} given by (30) with 19 written as $19 = p_0 + q_0 = 9 + 10$.
7. The function $\mathsf{load}(K, \mathsf{IV})$ outputs $K||\mathsf{IV}||(10)^8$.
8. The function initG is used for state initialisation.

Comparison and discussion. For the various proposals, the cryptographic properties of the constituent Boolean functions are given in Table 2 and the corresponding gate counts are given in Table 3. Along with the details of the new proposals, we also provide the details of Grain v1 and Grain-128a.

At the 80-bit security level, from Table 2 we note that compared to Grain v1, R-80 improves the algebraic degree (from 6 to 7) and linear bias (from $2^{-2.642}$ to $2^{-4.678}$) of g while maintaining the same level of algebraic immunity, and the algebraic degree (from 3 to 4), algebraic immunity (from 2 to 3),

	#FFs	g	G	h	H	$(G+H)$
Grain v1	160	6[N]+12[X]+17[A]	6[N]+15[X]+17[A]	1[N]+7[X]+6[A]	1[N]+14[X]+6[A]	7[N]+29[X]+23[A]
Grain-128a	256	9[X]+14[A]	14[X]+14[A]	4[X]+6[A]	12[X]+6[A]	26[X]+20[A]
R-80	160	1[N]+6[X]+10[A]	1[N]+9[X]+10[A]	2[N]+9[X]+8[A]	2[N]+16[X]+8[A]	3[N]+25[X]+18[A]
R-128	256	1[N]+14[X]+17[A]	1[N]+19[X]+17[A]	5[X]+8[A]	13[X]+8[A]	1[N]+32[X]+25[A]
W-128	240	1[N]+14[X]+17[A]	1[N]+19[X]+17[A]	5[X]+8[A]	13[X]+8[A]	1[N]+32[X]+25[A]
R-192	384	1[N]+19[X]+21[A]	1[N]+26[X]+21[A]	12[X]+13[A]	22[X]+13[A]	1[N]+48[X]+34[A]
W-192	352	1[N]+19[X]+21[A]	1[N]+26[X]+21[A]	12[X]+13[A]	22[X]+13[A]	1[N]+48[X]+34[A]
R-256	512	1[N]+22[X]+27[A]	1[N]+31[X]+27[A]	14[X]+17[A]	26[X]+17[A]	1[N]+55[X]+44[A]
W-256	464	1[N]+22[X]+27[A]	1[N]+31[X]+27[A]	14[X]+17[A]	26[X]+17[A]	1[N]+55[X]+44[A]

Table 3: Comparison of the number of flip-flops and gate counts for the nonlinear functions used in the different proposals.

and linear bias (from 2^{-2} to 2^{-3}) of h. From Table 3, we note that compared to Grain v1, R-80 reduces the total gate count for G and H from 7[N]+29[X]+23[A] gates to 3[N]+25[X]+18[A] gates. So overall, R-80 improves the cryptographic properties of the constituent functions *and* at the same time reduces the gate counts.

At the 128-bit security level, from Table 2, we note that compared to Grain-128a, R-128 and W-128 improve the algebraic degree (from 4 to 6) and the linear bias (from $2^{-8.023}$ to 2^{-12}) of g without reducing the algebraic immunity, and improve the algebraic degree (from 3 to 5) and the linear bias (from 2^{-4} to 2^{-5}) of h while maintaining the same level of algebraic immunity. From Table 3, we note that compared to Grain-128a, R-128 and W-128 increase the total gate count for G and H from 26[X]+20[A] gates to 1[N]+32[X]+25[A] gates. So the improvement of the cryptographic properties of g and h offered by R-128 and W-128 come at a small increase in gate count by 1[N]+6[X]+5[A] gates. While both Grain-128a and R-128 use 256 flip-flops, W-128 uses 240 flip-flops, i.e. a reduction of 16 flip-flops. Comparing Grain-128a and W-128, the reduction in the number of flip-flops by W-128 more than balances out the small increase in the number of gates required by G and H. So overall W-128 has a smaller gate count than Grain-128a.

For the 192-bit and the 256-bit security levels, to the best of our knowledge, the gate counts of the proposals R-192, W-192, R-256 and W-256 are smaller than all previous proposals for stream ciphers at these security levels.

Length of keystream. We propose that at most 2^{64} keystream bits should be generated from a single pair of secret key and IV. See Appendix B for a justification of this bound.

We note that Grain v1 and Grain-128a do not put any limit on the number of keystream bits that can be generated from a single key and IV pair. The fast correlation attack [76] on Grain v1 and Grain-128a requires $2^{75.1}$ and $2^{113.8}$ bits of keystream respectively to be generated from a single key and IV pair. As a response to this attack, Grain-128AEADv2 puts the restriction that at most 2^{81} bits should be generated from a single key-IV pair.

6.1 Security Conjecture

From a theoretical point of view, a stream cipher supporting an IV is modelled as pseudo-random function (PRF) that maps the IV to the keystream using the unknown secret key [12]. Assuming a stream cipher to be a PRF permits using the stream cipher for constructing other primitives and higher level functionalities. We mention some examples. Certain kinds of almost XOR universal (AXU) hash functions require a key which is as long as the message (and even longer than the message for the

Toeplitz construction). A stream cipher can be used to generate such a key. Bit oriented construction of such hash functions was put forward in [67] and their implementation was reported in [24]. Generic methods to construct AEAD and deterministic AEAD (DAEAD) schemes based on stream ciphers and appropriate AXU hash functions were described in [68] and these schemes were later analysed in [46]. Construction of a tweakable enciphering scheme from a stream cipher was described in [66], and hardware implementations of the scheme based on AXU functions from [67] and the stream ciphers in the hardware profile of the eStream recommendations were reported in [24].

The adversarial model for the PRF security of stream ciphers is the following. Fix a stream cipher. Suppose that at most 2^B keystream bits are to be generated from a single key and IV pair. (Recall that in our case $B = 64$.) Let $\mathcal{O}_0$ be an oracle which behaves as follows: a uniform random κ-bit secret key K is chosen and the stream cipher is instantiated with K; on provided with a v-bit IV as input and a positive integer $\ell \leq 2^B$, $\mathcal{O}_0$ returns the first ℓ-bit segment of the keystream generated using K and IV. Let $\mathcal{O}_1$ be an oracle which on provided with a v-bit IV and a positive integer $\ell \leq 2^B$ as input returns an independent and uniform random string of length ℓ. An adversary is an algorithm $\mathcal{A}^{\mathcal{O}}$ which has access to an oracle $\mathcal{O}$; $\mathcal{A}$ adaptively queries $\mathcal{O}$ with pairs (IV, ℓ), where IV is a v-bit string and $\ell \leq 2^B$ is a positive integer, and receives in return a bit string of length ℓ. After querying the oracle $\mathfrak{q}$ times, $\mathcal{A}$ outputs a bit. The advantage of $\mathcal{A}$ is defined to be $|\Pr[\mathcal{A}^{\mathcal{O}_0} \text{ outputs } 1] - \Pr[\mathcal{A}^{\mathcal{O}_1} \text{ outputs } 1]|$. An alternative way to view the advantage of an adversary $\mathcal{A}$ is the following. A uniform random bit b is chosen and $\mathcal{A}$ is given access to $\mathcal{O}_b$. At the end of the interaction with $\mathcal{O}_b$, $\mathcal{A}$ outputs a bit b'. It is not difficult to show that the absolute value of the bias of the bit $b \oplus b'$ is the advantage of $\mathcal{A}$.

The resources of $\mathcal{A}$ consist of the following: the total number $\mathfrak{q}$ of queries that it makes, the total number of bits $\mathfrak{B}$ that it receives in response to all its queries, the total time $\mathfrak{T}$ that it takes, and the maximum number $\mathfrak{M}$ of bits of memory that it requires. Since there are 2^v possible IV's, the maximum value of $\mathfrak{q}$ is 2^v. We make the reasonable assumptions that $\mathfrak{M} \leq \mathfrak{T}$ (i.e. each memory position is accessed at least once), and $\mathfrak{B} \leq \mathfrak{T}$ (i.e. each bit received from the oracle is read at least once).

Attacks which require physically impossible amounts of memory are perhaps not of any practical interest. In Appendix B we provide justifications for considering attacks requiring about 2^{100} or more bits of memory to be physically impossible to mount.

Suppose the advantage of an adversary is ε. By repeating the attack ε^{-2} times, the advantage can be made close to 1. This increases the required time by also a factor of ε^{-2}. Also, in each repetition the adversary is attacking a uniform and independently chosen random key, so repetitions correspond to multi-target attacks. Repetitions, on the other hand, do not increase the memory requirement. Our security conjecture is the following.

Conjecture 1 *For a stream cipher proposed in this work which is targeted at the κ-bit security level, and for which at most 2^{64} keystream bits are to be generated from a single pair of key and IV, for any adversary having advantage ε, requiring time $\mathfrak{T}$ and memory $\mathfrak{M} < 2^{\min(\kappa,100)}$ bits, we conjecture that $\mathfrak{T} \cdot \varepsilon^{-2} < 2^{\kappa}$.*

7 Analysis of Known Attacks

Several classes of attacks have been proposed on the previously proposed members of the Grain family of stream ciphers. In this section, we provide descriptions of these attacks and discuss their applicability to the new proposals.

7.1 Exhaustive Search

The basic attack is to search exhaustively for the secret key. Given a segment of the keystream generated with an unknown key and a known IV, the attacker tries out all possible choices of the secret key. For each choice of the secret key, the attacker generates a keystream segment of length equal to the given segment. If the generated keystream segment is equal to the given keystream segment, then the corresponding choice of the secret key is a candidate key. If the keystream segment is long enough (a little longer than κ), then it is likely that there will be unique candidate key. Clearly the procedure requires trying out 2^κ candidate keys.

7.2 Linear Approximations

Linear approximations of a sum of keystream bits have been used earlier to attack members of the Grain family [59, 13]. A comprehensive fast correlation attack on Grain v1, Grain-128 and Grain-128a was proposed in [76].

The crux of all correlation attacks lies in obtaining "good" linear approximations of a sum of keystream bits in terms of the bits of the LFSR sequence. This allows a divide-and-conquer attack, where the LFSR state is first recovered and then the NFSR state is recovered [76]. The task of obtaining linear approximations was done separately for Grain v1, Grain-128, and Grain-128a in [76]. In this section, we perform a very general and theoretical analysis of bias of linear approximations for the entire abstract Grain family that we have defined. For this purpose, we will use some results on correlations which are given in Appendix A.

As in Section 3, let $(N^{(0)}, L^{(0)})$ be the state produced after the initialisation phase (Steps 3 and 4) of Algorithm 1. For $t \geq 0$, let $N^{(t)} = (\eta_t, \eta_{t+1}, \ldots, \eta_{t+\kappa-1})$, $L^{(t)} = (\lambda_t, \lambda_{t+1}, \ldots, \lambda_{t+\kappa-1})$, and the t-th keystream bit be z_t.

We introduce a new notation. For $t \geq 0$ and $w \geq 1$, let

$$\boldsymbol{\lambda}_{t,w} = (\lambda_t, \lambda_{t+1}, \ldots, \lambda_{t+w-1}) \text{ and } \boldsymbol{\eta}_{t,w} = (\eta_t, \eta_{t+1}, \ldots, \eta_{t+w-1}). \tag{31}$$

Let T be a finite non-empty set of non-negative integers, and

$$S_1' = T \triangle (S_1 \cup \{\kappa_1\}), \tag{32}$$

where $\triangle$ denotes symmetric difference of sets. Using (7) we can write

$$\begin{aligned}
\lambda_t \oplus g(\mathsf{proj}(S_0, N^{(t)})) &= \eta_{t+\kappa_1} \oplus \mathsf{xor}(\mathsf{proj}(S_1, N^{(t)})) \\
&= \left(\bigoplus_{j \in S_1 \cup \{\kappa_1\}} \eta_{t+j}\right) \oplus \left(\bigoplus_{i \in T} \eta_{t+i}\right) \oplus \left(\bigoplus_{i \in T} \eta_{t+i}\right) \\
&= \left(\bigoplus_{i \in S_1'} \eta_{t+i}\right) \oplus \left(\bigoplus_{i \in T} \eta_{t+i}\right).
\end{aligned}$$

So

$$\lambda_t \oplus g(\mathsf{proj}(S_0, N^{(t)})) \oplus \bigoplus_{i \in S_1'} \eta_{t+i} = \bigoplus_{i \in T} \eta_{t+i} \tag{33}$$

From (5), we have

$$\bigoplus_{i \in T} z_{t+i} = \bigoplus_{i \in T} \mathsf{xor}(\mathsf{proj}(Q_1, L^{(t+i)})) \oplus \bigoplus_{i \in T} \mathsf{xor}(\mathsf{proj}(P_1, N^{(t+i)}))$$

$$
\begin{aligned}
&\oplus \bigoplus_{i\in T} h(\psi(\mathsf{proj}(Q_0, L^{(t+i)}), \mathsf{proj}(P_0, N^{(t+i)})))\\
&= \bigoplus_{i\in T}\bigoplus_{j\in Q_1} \lambda_{t+i+j} \oplus \bigoplus_{i\in T}\bigoplus_{j\in P_1} \eta_{t+i+j} \oplus \bigoplus_{i\in T} h(\psi(\mathsf{proj}(Q_0, L^{(t+i)}), \mathsf{proj}(P_0, N^{(t+i)})))\\
&= \bigoplus_{i\in T}\bigoplus_{j\in Q_1} \lambda_{t+i+j} \oplus \bigoplus_{j\in P_1}\bigoplus_{i\in T} \eta_{t+i+j} \oplus \bigoplus_{i\in T} h(\psi(\mathsf{proj}(Q_0, L^{(t+i)}), \mathsf{proj}(P_0, N^{(t+i)})))\\
&\qquad\qquad \text{(by interchanging the sums over } P_1 \text{ and } T)\\
&= \bigoplus_{i\in T}\bigoplus_{j\in Q_1} \lambda_{t+i+j} \oplus \bigoplus_{j\in P_1} \left(\lambda_{t+j} \oplus g(\mathsf{proj}(S_0, N^{(t+j)})) \oplus \bigoplus_{i\in S_1'} \eta_{t+i+j} \right) \quad \text{(from (33))}\\
&\oplus \bigoplus_{i\in T} h(\psi(\mathsf{proj}(Q_0, L^{(t+i)}), \mathsf{proj}(P_0, N^{(t+i)}))))\\
&= \bigoplus_{i\in T}\bigoplus_{j\in Q_1} \lambda_{t+i+j} \oplus \bigoplus_{j\in P_1} \lambda_{t+j} \oplus \bigoplus_{j\in P_1} g(\mathsf{proj}(S_0, N^{(t+j)})) \oplus \bigoplus_{j\in P_1}\bigoplus_{i\in S_1'} \eta_{t+i+j}\\
&\oplus \bigoplus_{i\in T} h(\psi(\mathsf{proj}(Q_0, L^{(t+i)}), \mathsf{proj}(P_0, N^{(t+i)}))). \qquad (34)
\end{aligned}
$$

Let

$$
\left.
\begin{aligned}
r &= 1 + \max(\max(Q_1 + T), \max(P_1), \max(Q_0 + T)),\\
s &= 1 + \max(\max(P_1 + S_0), \max(P_1 + S_1'), \max(P_0 + T)).
\end{aligned}
\right\} \qquad (35)
$$

Consider the sequences $\boldsymbol{\lambda}_{t,r}$ (of length r) and $\boldsymbol{\eta}_{t,s}$ (of length s). In (34) no bit outside the sequences $\boldsymbol{\lambda}_{t,r}$ and $\boldsymbol{\eta}_{t,s}$ are involved. So we may focus our attention only on the bits of $\boldsymbol{\lambda}_{t,r}$ and $\boldsymbol{\eta}_{t,s}$. Note that not all of the bits of these two sequences appear in (34).

Let $\boldsymbol{\gamma} \in \mathbb{F}_2^r$ and define the bit

$$
b_{t,T,\boldsymbol{\gamma}} = \langle \boldsymbol{\gamma}, \boldsymbol{\lambda}_{t,r} \rangle \oplus \bigoplus_{i\in T} z_{t+i}. \qquad (36)
$$

For a fixed T, we are interested in all those $\boldsymbol{\gamma}$'s such that $|\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}})|$ is sufficiently high for all $t \geq 0$. The fast correlation attack in [76] requires about $|\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}})|^{-2}$ keystream bits.

Using (34), we have

$$
\begin{aligned}
b_{t,T,\boldsymbol{\gamma}} &= \langle \boldsymbol{\gamma}, \boldsymbol{\lambda}_{t,r} \rangle \oplus \bigoplus_{i\in T}\bigoplus_{j\in Q_1} \lambda_{t+i+j} \oplus \bigoplus_{j\in P_1} \lambda_{t+j} \oplus \bigoplus_{i\in T} h(\psi(\mathsf{proj}(Q_0, L^{(t+i)}), \mathsf{proj}(P_0, N^{(t+i)})))\\
&\oplus \bigoplus_{j\in P_1}\bigoplus_{i\in S_1'} \eta_{t+i+j} \oplus \bigoplus_{j\in P_1} g(\mathsf{proj}(S_0, N^{(t+j)})). \qquad (37)
\end{aligned}
$$

Given a non-empty subset T of non-negative integers and $\boldsymbol{\gamma} \in \mathbb{F}_2^r$, define

$$
\left.
\begin{aligned}
&B = (T + Q_1) \triangle P_1 \triangle \mathsf{supp}(\boldsymbol{\gamma}).\\
&C = P_1 + S_1', \text{ where } S_1' = T \triangle (S_1 \cup \{\kappa_1\}).\\
&\text{For } i \in T, D_i = Q_0 + i, \text{ and } D = \cup_{i\in T} D_i = Q_0 + T.\\
&\text{For } i \in T, E_i = P_0 + i, \text{ and } E = \cup_{i\in T} E_i = P_0 + T.\\
&\text{For } j \in P_1, F_j = S_0 + j, \text{ and } F = \cup_{j\in P_1} F_j = S_0 + P_1.
\end{aligned}
\right\} \qquad (38)
$$

With the above notation and the definitions of $\boldsymbol{\lambda}_{t,r}$ and $\boldsymbol{\eta}_{t,s}$, for each $i \in T$ and $j \in P_1$, we have

$$\left.\begin{array}{rcl} \mathsf{proj}(D_i, \boldsymbol{\lambda}_{t,r}) & = & \mathsf{proj}(Q_0, L^{(t+i)}), \\ \mathsf{proj}(E_i, \boldsymbol{\eta}_{t,s}) & = & \mathsf{proj}(P_0, N^{(t+i)}), \\ \mathsf{proj}(F_j, \boldsymbol{\eta}_{t,s}) & = & \mathsf{proj}(S_0, N^{(t+j)}). \end{array}\right\} \tag{39}$$

Using (38) and (39), we can rewrite $b_{t,T,\boldsymbol{\gamma}}$ given by (37) as follows.

$$\begin{aligned} b_{t,T,\boldsymbol{\gamma}} &= \mathsf{xor}(\mathsf{proj}(B, \boldsymbol{\lambda}_{t,r})) \oplus \bigoplus_{i \in T} h(\psi(\mathsf{proj}(D_i, \boldsymbol{\lambda}_{t,r}), \mathsf{proj}(E_i, \boldsymbol{\eta}_{t,s}))) \\ &\quad \oplus \mathsf{xor}(\mathsf{proj}(C, \boldsymbol{\eta}_{t,s})) \oplus \bigoplus_{j \in P_1} g(\mathsf{proj}(F_j, \boldsymbol{\eta}_{t,s})). \end{aligned} \tag{40}$$

Proposition 14 *If $B \setminus D \neq \emptyset$, or $C \setminus (E \cup F) \neq \emptyset$, then $\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}}) = 0$. Consequently, if $b_{t,T,\boldsymbol{\gamma}}$ has a non-zero bias, then*

$$(T + Q_1) \triangle P_1 \triangle \mathsf{supp}(\boldsymbol{\gamma}) \subseteq Q_0 + T \quad \textit{and} \quad P_1 + (T \triangle (S_1 \cup \{\kappa_1\})) \subseteq (P_0 + T) \cup (S_0 + P_1). \tag{41}$$

Proof: The positions of B determine the bits of $\boldsymbol{\lambda}_{t,r}$ which appear as linear terms in (40), while the positions of D determine the bits of $\boldsymbol{\lambda}_{t,r}$ which appear in the inputs to the different invocations of h in (40). If there is a position in B which does not appear in D, then the entire expression given by (37) is uniformly distributed (due to the assumption on the independent and uniform distribution of the bits of $\boldsymbol{\lambda}_{t,r}$) and hence $\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}}) = 0$. The statement that if $C \setminus (E \cup F) \neq \emptyset$, then $\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}}) = 0$ is obtained by a similar argument over $\boldsymbol{\eta}_{t,s}$.

So if $\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}}) \neq 0$, then $B \subseteq D$ and $C \subseteq E \cup F$. Then (41) is obtained using the definitions of B, C, D, E and F given in (38). □

The condition on T given by (41) is a necessary condition for $b_{t,T,\boldsymbol{\gamma}}$ to have a nonzero bias. While choosing T, the cryptanalyst must ensure that (41) holds, as otherwise the bias of $b_{t,T,\boldsymbol{\gamma}}$ will be zero. So Proposition 14 provides a basic guidance on how to choose T. Note that since (41) is a necessary condition, it is not guaranteed that choosing T satisfying this condition ensures that $b_{t,T,\boldsymbol{\gamma}}$ has a non-zero bias.

Let $\boldsymbol{\gamma}' \in \mathbb{F}_2^r$ and $\boldsymbol{\delta} \in \mathbb{F}_2^s$ be such that

$$\langle \boldsymbol{\gamma}', \boldsymbol{\lambda}_{t,r} \rangle = \mathsf{xor}(\mathsf{proj}(B, \boldsymbol{\lambda}_{t,r})) \quad \text{and} \quad \langle \boldsymbol{\delta}, \boldsymbol{\eta}_{t,s} \rangle = \mathsf{xor}(\mathsf{proj}(C, \boldsymbol{\eta}_{t,s})). \tag{42}$$

Further, for $i \in T$ and for $j \in P_1$ define $\mathfrak{h}_i$ and $\mathfrak{g}_j$ to be functions such that

$$\left.\begin{array}{rcl} \mathfrak{h}_i(\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s}) & = & h(\psi(\mathsf{proj}(D_i, \boldsymbol{\lambda}_{t,r}), \mathsf{proj}(E_i, \boldsymbol{\eta}_{t,s}))) \\ \mathfrak{g}_j(\boldsymbol{\eta}_{t,s}) & = & g(\mathsf{proj}(F_j, \boldsymbol{\eta}_{t,s})). \end{array}\right\} \tag{43}$$

From the definition, it follows that $\mathfrak{h}_i$ is degenerate on all bits of $\boldsymbol{\lambda}_{t,r}$ other than those indexed by D_i, $\mathfrak{h}_i$ is degenerate on all bits of $\boldsymbol{\eta}_{t,s}$ other than those indexed by E_i, and $\mathfrak{g}_j$ is degenerate on all bits of $\boldsymbol{\eta}_{t,s}$ other than those indexed by F_j.

Let $\mathfrak{h}$ and $\mathfrak{g}$ be functions such that

$$\mathfrak{h}(\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s}) = \bigoplus_{i \in T} \mathfrak{h}_i(\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s}) \quad \text{and} \quad \mathfrak{g}(\boldsymbol{\eta}_{t,s}) = \bigoplus_{j \in P_1} \mathfrak{g}_j(\boldsymbol{\eta}_{t,s}). \tag{44}$$

Using (42), (43) and (44), we rewrite (40) as follows.

$$b_{t,T,\boldsymbol{\gamma}} = \langle \boldsymbol{\gamma}', \boldsymbol{\lambda}_{t,r} \rangle \oplus \langle \boldsymbol{\delta}, \boldsymbol{\eta}_{t,s} \rangle \oplus \bigoplus_{i \in T} \mathfrak{h}_i(\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s}) \oplus \bigoplus_{j \in P_1} \mathfrak{g}_j(\boldsymbol{\eta}_{t,s}) \tag{45}$$

$$= \langle \boldsymbol{\gamma}', \boldsymbol{\lambda}_{t,r} \rangle \oplus \langle \boldsymbol{\delta}, \boldsymbol{\eta}_{t,s} \rangle \oplus \mathfrak{h}(\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s}) \oplus \mathfrak{g}(\boldsymbol{\eta}_{t,s}). \tag{46}$$

Using Propositions 18 and 19 (given in Appendix A), we relate the correlations of $\mathfrak{h}_i$ and $\mathfrak{g}_j$ to those of h and g respectively.

Proposition 15 *For $\boldsymbol{\alpha} \in \mathbb{F}_2^r$ and $\boldsymbol{\beta} \in \mathbb{F}_2^s$,*

$$\mathsf{corr}_{\mathfrak{h}_i}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{cases} 0 & \text{if } \mathsf{supp}(\boldsymbol{\alpha}) \setminus D_i \neq \emptyset \text{ or } \mathsf{supp}(\boldsymbol{\beta}) \setminus E_i \neq \emptyset, \\ \mathsf{corr}_h(\psi(\mathsf{proj}(D_i, \boldsymbol{\alpha}), \mathsf{proj}(E_i, \boldsymbol{\beta}))) & \text{if } \mathsf{supp}(\boldsymbol{\alpha}) \subseteq D_i \text{ and } \mathsf{supp}(\boldsymbol{\beta}) \subseteq E_i. \end{cases} \tag{47}$$

For $\boldsymbol{\beta} \in \mathbb{F}_2^s$,

$$\mathsf{corr}_{\mathfrak{g}_j}(\boldsymbol{\beta}) = \begin{cases} 0 & \text{if } \mathsf{supp}(\boldsymbol{\beta}) \setminus F_j \neq \emptyset, \\ \mathsf{corr}_g(\mathsf{proj}(F_j, \boldsymbol{\beta})) & \text{if } \mathsf{supp}(\boldsymbol{\beta}) \subseteq F_j. \end{cases} \tag{48}$$

Using the definitions of $\mathfrak{h}$ and $\mathfrak{g}$ given by (44) and Theorem 3 (given in Appendix A), we have the following result.

Proposition 16 *Let $\mathfrak{t} = \#T$. Then*

$$\mathsf{corr}_{\mathfrak{h}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{\{(\mathbf{u}_i, \mathbf{v}_i) \in \mathbb{F}_2^r \times \mathbb{F}_2^s : 1 \leq i \leq \mathfrak{t}-1\}} \prod_{i=1}^{\mathfrak{t}} \mathsf{corr}_{\mathfrak{h}_i}(\mathbf{u}_i \oplus \mathbf{u}_{i-1}, \mathbf{v}_i \oplus \mathbf{v}_{i-1}), \tag{49}$$

where $\mathbf{u}_0 = \mathbf{0}_r$, and $\mathbf{u}_{\mathfrak{t}} = \boldsymbol{\alpha}$, $\mathbf{v}_0 = \mathbf{0}_s$, and $\mathbf{v}_{\mathfrak{t}} = \boldsymbol{\beta}$.

Further,

$$\mathsf{corr}_{\mathfrak{g}}(\boldsymbol{\beta}) = \sum_{\{\mathbf{w}_j \in \mathbb{F}_2^s : 1 \leq j \leq p_1 - 1\}} \prod_{j=1}^{p_1} \mathsf{corr}_{\mathfrak{g}_j}(\mathbf{w}_j \oplus \mathbf{w}_{j-1}), \tag{50}$$

where $\mathbf{w}_0 = \mathbf{0}_s$, and $\mathbf{w}_{p_1} = \boldsymbol{\beta}$.

Remark 12 *Note that the expression in (49) is a sum of products, where each product term is obtained by multiplying together $\mathfrak{t}$ correlations of $\mathfrak{h}_i$, which from (47) amounts to multiplying together at most $\mathfrak{t}$ correlations of h. So the absolute value of each of these products of correlations of h is at most $\varepsilon_{h,\mathfrak{t}} = \mathsf{LB}(h)^{\mathfrak{t}}$.*

Theorem 2 *Let $t \geq 0$, T be a finite non-empty set of integers, and r be defined as in (35). For $\boldsymbol{\gamma} \in \mathbb{F}_2^r$, let $b_{t,T,\boldsymbol{\gamma}}$ be defined as in (36). Let $\boldsymbol{\gamma}'$ and $\boldsymbol{\delta}$ be defined as in (42). Then*

$$\mathsf{bias}(b_{t,T,\gamma}) = \sum_{\boldsymbol{\beta} \in \mathbb{F}_2^r} \mathsf{corr}_{\mathfrak{h}}(\boldsymbol{\gamma}', \boldsymbol{\beta}) \cdot \mathsf{corr}_{\mathfrak{g}}(\boldsymbol{\beta} \oplus \boldsymbol{\delta}). \tag{51}$$

Proof: Using (46), we have

$$b_{t,T,\boldsymbol{\gamma}} \;=\; \langle(\boldsymbol{\gamma}', \boldsymbol{\delta}), (\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s})\rangle \oplus \mathfrak{h}(\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s}) \oplus \mathfrak{g}(\boldsymbol{\eta}_{t,s}). \tag{52}$$

So $\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}})$ is equal to $\mathsf{corr}_{\mathfrak{f}}(\boldsymbol{\gamma}', \boldsymbol{\delta})$, where $\mathfrak{f}(\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s}) = \mathfrak{h}(\boldsymbol{\lambda}_{t,r}, \boldsymbol{\eta}_{t,s}) \oplus \mathfrak{g}(\boldsymbol{\eta}_{t,s})$. From Theorem 3 (given in Appendix A), we have

$$\mathsf{corr}_{\mathfrak{f}}(\boldsymbol{\gamma}', \boldsymbol{\delta}) \;=\; \sum_{(\boldsymbol{\alpha},\boldsymbol{\beta}) \in \mathbb{F}_2^s \times \mathbb{F}_2^r} \mathsf{corr}_{\mathfrak{h}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \cdot \mathsf{corr}_{\mathfrak{g}}((\boldsymbol{\alpha}, \boldsymbol{\beta}) \oplus (\boldsymbol{\gamma}', \boldsymbol{\delta})). \tag{53}$$

Since $\mathfrak{g}$ is degenerate on $\boldsymbol{\lambda}_{t,r}$, from Proposition 18 we have, $\mathsf{corr}_{\mathfrak{g}}((\boldsymbol{\alpha}, \boldsymbol{\beta}) \oplus (\boldsymbol{\gamma}', \boldsymbol{\delta})) = 0$ unless $\boldsymbol{\alpha} = \boldsymbol{\gamma}'$. So (53) simplifies to

$$\mathsf{corr}_{\mathfrak{f}}(\boldsymbol{\gamma}', \boldsymbol{\delta}) \;=\; \sum_{\boldsymbol{\beta} \in \mathbb{F}_2^r} \mathsf{corr}_{\mathfrak{h}}(\boldsymbol{\gamma}', \boldsymbol{\beta}) \cdot \mathsf{corr}_{\mathfrak{g}}(\boldsymbol{\beta} \oplus \boldsymbol{\delta}). \tag{54}$$

□

In (51), the value of $\mathsf{corr}_{\mathfrak{h}}(\boldsymbol{\gamma}', \boldsymbol{\beta})$ is obtained from (49) as a sum of products of the correlations of the $\mathfrak{h}_i$'s, where as mentioned in Remark 12 the absolute value of each of these products is at most $\mathsf{LB}(h)^{\mathfrak{t}}$.

From Table 2, $\mathsf{LB}(h) = 2^{-2}$ and $\mathsf{LB}(h) = 2^{-4}$ for Grain v1 and Grain-128a respectively. In [76], $\mathfrak{t}$ was taken to be 10 and 6 for Grain v1 and Grain-128a respectively. The corresponding values of $\varepsilon_{h,\mathfrak{t}}$ are 2^{-20} and 2^{-24} for Grain v1 and Grain-128a respectively.

Using the values of $\mathsf{LB}(h)$ (from Table 2), we have $\varepsilon_{h,\mathfrak{t}}$ equal to $2^{-3\mathfrak{t}}$, $2^{-5\mathfrak{t}}$, $2^{-7\mathfrak{t}}$ and $2^{-9\mathfrak{t}}$ for the 80-bit (proposal R-80), 128-bit (proposals R-128 and W-128), 192-bit (proposals R-192 and W-192), and 256-bit (proposals R-256 and W-256) security levels respectively. The actual value of $\varepsilon_{h,\mathfrak{t}}$ depends on the value of $\mathfrak{t}$. If, as in [76], we choose $\mathfrak{t} = 10$ and $\mathfrak{t} = 6$ for the 80-bit and the 128-bit security levels, then the corresponding values of $\varepsilon_{h,\mathfrak{t}}$ that we obtain are both equal to 2^{-30} which is substantially lower than the values of $\varepsilon_{h,\mathfrak{t}}$ obtained for Grain v1 and Grain-128a.

Next we consider $\mathsf{corr}_{\mathfrak{g}}$ which appears in (51). Combining (44) and (43) we see that the function $\mathfrak{g}$ is a sum of p_1 copies of the function g. The correlation of $\mathfrak{g}$ can be expressed in terms of the correlations of g by combining (50) and (48). In [76] however, for both Grain v1 and Grain-128a, the correlation of $\mathfrak{g}$ was estimated by directly considering its ANF and individually estimating the correlation of various groups of terms in the ANF. In general, even though $\mathfrak{g}$ is a sum of p_1 copies of the function g, the different copies of g are not applied to disjoint set of variables, i.e. $\mathfrak{g}$ is not a direct sum of p_1 copies of g. If $\mathfrak{g}$ were such a direct sum, then from Proposition 20 (in Appendix A) it would have been easy to obtain the correlations of $\mathfrak{g}$ in terms of the correlations of g. Whether or not $\mathfrak{g}$ is a direct sum of p_1 copies of g depends on the choice of the tap positions. The following result provides a sufficient condition for $\mathfrak{g}$ to be a direct sum and in this case relates the correlation of $\mathfrak{g}$ to the correlation of g.

Proposition 17 *If $\#(S_0 + P_1) = \#S_0 \cdot \#P_1$, then for $\mathfrak{g}$ defined in (44),*

$$\mathsf{corr}_{\mathfrak{g}}(\boldsymbol{\beta}) \;=\; \prod_{j \in P_1} \mathsf{corr}_g(\mathsf{proj}(F_j, \boldsymbol{\beta})). \tag{55}$$

Consequently,

$$|\mathsf{corr}_{\mathfrak{g}}(\boldsymbol{\beta})| \;\leq\; \varepsilon_g. \tag{56}$$

where $\varepsilon_g = \mathsf{LB}(g)^{p_1}$.

Proof: The condition $\#(S_0 + P_1) = \#S_0 \cdot \#P_1$ holds if and only if the F_j's are pairwise disjoint. Using the definition of $\mathfrak{g}_j$ from (43) in the defintion of $\mathfrak{g}$ in (44), we have

$$\mathfrak{g}(\boldsymbol{\eta}_{t,s}) \;=\; \bigoplus_{j \in P_1} \mathfrak{g}_j(\boldsymbol{\eta}_{t,s}) = \bigoplus_{j \in P_1} g(\mathsf{proj}(F_j, \boldsymbol{\eta}_{t,s})).$$

Since the F_j's are pairwise disjoint it follows that $\mathfrak{g}$ is a direct sum of p_1 copies of g. The result then follows from Proposition 20 (given in Appendix A).

Since $|\mathsf{corr}_g(\mathsf{proj}(F_j, \boldsymbol{\beta}))| \leq \mathsf{LB}(g)$, we obtain (56) from (55). □

Remark 13 *Using Proposition 17, if we choose S_0 and P_1 such that $\#(S_0 + P_1) = \#S_0 \cdot \#P_1$, then we obtain a good upper bound on the quantity $|\mathsf{corr}_{\mathfrak{g}}(\boldsymbol{\beta} \oplus \boldsymbol{\delta})|$ appearing in (51), namely $|\mathsf{corr}_{\mathfrak{g}}(\boldsymbol{\beta} \oplus \boldsymbol{\delta})| \leq \mathsf{LB}(g)^{p_1}$. So based on Proposition 17 we set a design condition that $\#(S_0 + P_1) = \#S_0 \cdot \#P_1$. The new proposals that we put forward satisfy this condition. Using the values of p_1 (from Table 1) and $\mathsf{LB}(g)$ (from Table 2), the upper bounds on $|\mathsf{corr}_{\mathfrak{g}}(\boldsymbol{\beta} \oplus \boldsymbol{\delta})|$ are $2^{-28.068}$, 2^{-48}, 2^{-75} and 2^{-108} for the 80-bit (proposal R-80), 128-bit (proposals R-128 and W-128), 192-bit (proposals R-192 and W-192), and 256-bit (proposals R-256 and W-256) security levels respectively.*

From a security point of view, it is desirable to be able to prove a good upper bound on $|\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}})|$ as given by (51). While, as discussed above, we can prove that $\varepsilon_{h,\mathfrak{t}}$ and ε_g are provably low (see Remarks 12 and 13), due to the complications created by the signs of the correlations it seems difficult to use these values to prove useful upper bounds on $|\mathsf{bias}(b_{t,T,\boldsymbol{\gamma}})|$. So the low values of $\varepsilon_{h,\mathfrak{t}}$ and ε_g provide good evidence of resistance to fast correlation attacks, but do not provide provable security against such attacks.

7.3 Chosen IV Attacks

In chosen IV attacks, one or more bits of the keystream are obtained from the stream cipher with the same secret key and several chosen IVs. The obtained keystream bits are analysed to gain information about the secret key. Two types of chosen IV attacks that have been applied to the Grain family are cube attacks and conditional differential attacks. Below we briefly discuss these two types of attacks.

7.3.1 Cube Attacks

Fix any bit z of the keystream (produced after initialisation phase is over). This keystream bit is a function of the secret key K and the initialisation vector IV. Let us denote this dependence as $z = f(K, \mathrm{IV})$, where f is a Boolean function of $\kappa + v$ variables $K_1, \ldots, K_\kappa$ and $V_1, \ldots, V_v$. Suppose there is a subset $\{i_1, \ldots, i_u\} \subseteq \{1, \ldots, v\}$, and corresponding list of variables $\mathbf{V} = (V_{i_1}, \ldots, V_{i_u})$ such that

$$f(K, \mathrm{IV}) \;=\; V_{i_1} \cdots V_{i_u} p(K, \mathrm{IV}) \oplus q(K, \mathrm{IV}), \tag{57}$$

for some Boolean functions $p(K, \mathrm{IV})$ and $q(K, \mathrm{IV})$ satisfying the following two conditions.

1. $p(K, \mathrm{IV})$ does not involve any variable from $\mathbf{V}$.
2. Each term in the ANF of $q(K, \mathrm{IV})$ misses out at least one of the variables from $\mathbf{V}$.

For $\mathbf{v} \in \mathbb{F}_2^u$, let $f_{\mathbf{V} \leftarrow \mathbf{v}}$ be the function obtained by setting the variables in $\mathbf{V}$ to the values in $\mathbf{v}$. We have

$$\bigoplus_{\mathbf{v} \in \mathbb{F}_2^u} f_{\mathbf{V} \leftarrow \mathbf{v}}(K, \mathrm{IV}) \;=\; \bigoplus_{\mathbf{v} \in \mathbb{F}_2^u} (V_{i_1} \cdots V_{i_u} p(K, \mathrm{IV}) \oplus q(K, \mathrm{IV})) = p(K, \mathrm{IV}). \tag{58}$$

Since each term of $q(K, \mathrm{IV})$ misses out at least one variable from $\mathbf{V}$, the sum over the "cube" $\mathbb{F}_2^u$ results in each term of $q(K, \mathrm{IV})$ appearing an even number of times in the sum, and hence the total contribution of $q(K, \mathrm{IV})$ over the entire cube is 0. On the other hand the monomial $V_{i_1} \cdots V_{i_u}$ is 1 for exactly one point in the cube $\mathbb{F}_2^u$ (i.e. when $\mathbf{v} = \mathbf{1}_u$). This shows that (58) indeed holds. The function $p(K, \mathrm{IV})$ is called a superpoly.

In an offline phase, the cryptanalyst obtains one or more superpolys. In the online phase, the cryptanalyst makes 2^u queries to the oracle for the stream cipher fixing the variables in $V_1, \ldots, V_v$ outside that of $\mathbf{V}$ to some fixed values, and trying out all values of the variables in $\mathbf{V}$ over the cube $\mathbb{F}_2^u$. From (58), the cryptanalyst obtains the value of $p(K, \mathrm{IV})$ for the unknown secret key K. If the form of the superpoly is simple, then this value provides information to the cryptanalyst to narrow down the search for the secret K.

Cube attacks were introduced in [31]. Dynamic cube attacks were applied [32, 30] to perform key recovery for Grain-128 (which is an earlier variant of Grain-128a). Cube attacks were improved in [75, 77] using the division property. Refinement of this technique led to key recovery attacks on 190-round Grain-128a [37].

Resistance to cube attacks arises from the effectiveness of the mixing of the secret key K and the IV that is performed during the initialisation phase. After the initialisation phase, we may consider each of the bits of the state, i.e. the bits of N and L, to be functions of K and IV. The output bit is generated from these state bits by applying the function H. So a design goal is to try to ensure that the functions determining the state bits after the initialisation phase are complex nonlinear functions of K and IV. In particular, such functions should not have low degrees. The functions determining the state bits after the initialisation phase are based on the nature of the nonlinear functions g and h and the method of updating the state during initialisation.

In our proposals, for both the 80-bit and the 128-bit security levels, the degrees of the functions g and h are higher than the corresponding functions for Grain v1 and Grain-128a. Further, the new method of updating the state during the initialisation (function NSIG in (29)) that we propose provides a better nonlinear mixing of the key and the IV compared to the state updation method during initialisation (function NSI in (8)) used for all prior members of the Grain family (see the discussion following Remark 9). Due to the higher algebraic degrees of the constituent Boolean functions and the better mixing of the key and the IV, we expect the new proposals to provide improved resistance to cube attacks.

7.3.2 Conditional Differential Attacks

The attacker introduces two IVs with a difference in a single bit. Consider the differences between the states generated from the two IVs. Initially there is a difference at only a single point. As the initialisation proceeds, the difference propagates to more and more bits. Analysis of the stopping conditions for the differences leads to two types of relations. The first type of relations involves the bits of the IV, while the second type of relations involves both the bits of the IV and the secret key. The first type of relations are easy to satisfy. For the second type of relations, the attacker tries to control the parameters such that the relations are expressed as a direct sum of two Boolean functions, one on the key bits and the other on the IV bits. The ability to stop the propagation of the differences provides the adversary with some information about the non-randomness of the keystream bits. This information can be exploited to launch an attack. Conditional differential attack on Grain v1 was introduced in [48] and later explored in several papers for both Grain v1 and Grain-128a [53, 5, 6, 56, 57, 55].

Resistance to conditional differential attacks is determined to a large extent by the algebraic degree of the update functions. It was noted in [6] that application of conditional differential attack to Grain v1 is more difficult that to Grain-128a due to the degree of g for Grain v1 being higher than the degree

of g for Grain-128a. For the new proposals at the 80-bit and the 128-bit security levels, the degrees of the function g are higher than the corresponding degrees of the function g for Grain v1 and Grain-128a respectively. Further, due to our new state update function during initialisation, the degrees of the functions representing the bits of the LFSR grow much faster than the degrees of the functions obtained from the state update function during initialisation used for all earlier members of the Grain family. Due to the combined effect of these two points, the new proposals offer improved resistance to the conditional differential attacks than the previous proposals.

7.4 Near Collision Attacks

Near collisions attacks on Grain v1 were reported in [78, 79, 7]. However, as explained in [7] the attacks reported in [78, 79] are incorrect. The idea behind the near collision attack on Grain v1 described in [7] is the following. If about $2^{81.5}$ keystream bits are generated from a single key and IV pair, then with high probability during the process two states will be encountered which differ only in one bit of the LFSR. This observation is used to build a state recovery algorithm which performs about $2^{74.6}$ encryptions and $2^{80.5}$ table accesses which leads to a time complexity which is a little above the exhaustive search algorithm.

While in principle the idea behind the attack can also apply to other members of the Grain family, the number of keystream bits needed to be generated from a single key and IV pair to apply the attack appears to be well beyond the bound of 2^{64} that we have imposed. Further, as in the application to Grain v1, the time complexity of the attack also appears to be above that of exhaustive search.

7.5 Time/Memory/Data Trade-Off Attacks

Time/memory trade-off (TMTO) attacks were introduced by Hellman [43] in the context of block ciphers. Application to stream ciphers was pointed out by Babbage [3] and Golic [34] and is called the BG attack. Biryukov and Shamir [14] proposed a time/memory/data trade-off (TMDTO) attack called the BS attack. The A5/1 stream cipher was cryptanalysed using a trade-off attack in [15]. Let P, T, M, and D denote the pre-computation time, the online time, the memory, and the data required for carrying out a trade-off attack on a search space of size N_0. In the context of stream ciphers N_0 is the size of the possible values of the state which is equal to $2^{\kappa_1+\kappa_2}$ for the Grain family. The BG attack has the trade-off curve $TM = N_0$, $P = M$ and $T \leq D$ with best attack complexity given by the trade-off point $T = M = D = N_0^{1/2}$; the BS attack has the trade-off curve $MT^2D^2 = N_0^2$, $P = N_0/D$ and $T \geq D^2$ with the best attack complexity given by the trade-off point $T = M = (N_0/D)^{2/3}$ and $P = N_0/D$. Application of TMDTO to stream ciphers with IV was proposed in [45]. TMDTO has been applied to Grain-128a [27, 51].

If $\kappa_1 + \kappa_2 = 2\kappa$, then the complexity of the BG attack is not better than exhaustive search. The BS attack does not offer lower attack complexity. TMDTO attacks sometime allow P to be greater than 2^κ, for example the attack on Grain-128a in [51] has $P = 2^{170}$, which is significantly greater than exhaustive search time of 2^{128}. This aspect makes such attacks somewhat theoretical in nature. There is another problematic issue regarding the feasibility of TMDTO attacks. Such attacks require very high memory, for example the attack on Grain-128a in [51] require $M = 2^{113}$.

If $\kappa_1 + \kappa_2 < 2\kappa$, then the BG attack requires $P = T = M = D = 2^{(\kappa_1+\kappa_2)/2} < 2^\kappa$. Since $T < 2^\kappa$, such an attack may be considered to break the security of the corresponding stream cipher. For example, for W-128, W-192, and W-256, the TMDTO attacks require all three of time (both pre-computation and online), memory, and data to be equal to 2^{120}, 2^{176}, and 2^{232} respectively. If we consider only time, then these are valid attacks. However, the memory requirements for these attacks are very high. In

Appendix B, we provide arguments which suggest that it is physically infeasible to apply attacks which require 2^{100} or more bits of memory. In particular, the numbers 2^{176} and 2^{232} are both greater than the number of atoms in the world. So even though the TMDTO attacks on W-128, W-192, and W-256 require time less than 2^{κ}, the attacks cannot be mounted due to the infeasible memory requirement.

More generally, theoretical time/memory/data trade-off attacks requiring physically infeasible amounts of memory are not a real concern. For assessing the security of a real world cryptosystem, it is perhaps best to ignore physically infeasible attacks.

7.6 Related Key Attacks

Related key attacks have been proposed on Grain v1 and Grain-128a in [52] and later on Grain-128a in [11, 29]. The attack in [11] finds the secret key using a small multiple of 2^{32} related keys and 2^{64} chosen IVs, whereas the attack in [29] finds the secret key using two related keys, 2^{96} chosen IVs, and 2^{104} keystream bits. The basic idea behind the attacks is that the states that are produced after the initialisation corresponding to two or more related keys are related in some detectable manner. This suggests that the state update function that is used during initialisation does not adequately mix the secret key and the IV. In our new proposals, we have introduced a new state update function during initialisation which performs a better mixing of the secret key and the IV (in a qualitative sense). As a result, the new proposals will provide better resistance to related key attacks. We also note that whether related key attacks are a concern for stream ciphers is a debatable issue.

7.7 Fault Attacks

Fault attacks on stream ciphers were proposed in [44]. A long and productive line of work [22, 47, 9, 28, 10, 71, 60, 73, 54, 23, 8, 65] has investigated various aspects of fault attacks on members of the Grain family. The design of the Grain family does not include any inherent protection against fault attacks. In scenarios where fault attacks are considered a threat, appropriate counter-mechanisms (such as shielding) need to be deployed.

8 State Guessing Attacks

For three of the proposals that we put forward, the length of the LFSR is smaller than the length of the NFSR. This opens up the possiblity of new kinds of attacks which would not be applicable to previous members of the Grain family. In this section, we consider the possibility of applying a state guessing attack to proposals which have the LFSR to be smaller than the NFSR.

At each time point, the state of the stream cipher consists of the state of the NFSR and the state of the LFSR. Suppose that the attacker knows a sufficiently long keystream segment. An attack approach is to guess the value of ℓ bits of the state and use the keystream to verify the guess. Here guessing essentially means trying out all possible 2^{ℓ} values of the part of the state which is of interest. Suppose the verification requires 2^{v} operations. For the attack to be valid (i.e. faster than exhaustive search), we must have $2^{\ell+v} < 2^{\kappa}$. Since the NFSR is of length $\kappa_1 \geq \kappa$, guessing the state of the NFSR does not provide a valid attack.

For the W-series of stream ciphers, we propose the length of the LFSR to be $\kappa_2 < \kappa$. This opens up the possibility that guessing the state of the LFSR will lead to a valid attack. Suppose at some time point, the state $L^{(t)}$ of the LFSR is known. Since the evolution of the LFSR does not depend on the NFSR, given $L^{(t)}$ we may assume that the state of the LFSR is known for all time points (greater than or less than t). The output bit OB is given by the function H which consists of $p = p_0 + p_1$ bits from the

NFSR and $q = q_0 + q_1$ bits from the LFSR. Since the state of the LFSR is known at all time points, the inputs to H coming from the LFSR are fixed at all time points. So the keystream bit z_t at time point t can be expressed as a sum of p_1 bits extracted from $N^{(t)}$, and the output of a nonlinear function h_t on p_0 other bits, where for each t, h_t is obtained from h by fixing the values of the q_0 variables obtained from the LFSR to specific values. Since the tap positions for the p_1 bits and the tap positions for the p_0 bits are disjoint, it follows that z_t is a balanced bit. So the value of z_t reduces the number of options for the $p_0 + p_1$ positions of $N^{(t)}$ to $2^{p_0+p_1-1}$ from $2^{p_0+p_1}$. So effectively the knowledge of the bit z_t reduces the number of possible choices for $N^{(t)}$ by half. Successive keystream bits $z_{t+1}, z_{t+2}, \ldots$ further reduces the choices of $N^{(t)}$ by half each. This continues for a few time points until the nonlinear feedback bit produced from $N^{(t)}$ gets shifted into one of the tap positions of the NFSR that is required to produce the output bit. From that time point, the keystream bit becomes a more complex nonlinear function of the bits of $N^{(t)}$. To summarise, once $L^{(t)}$ is known at some time point t, any keystream bit at time point t, or previous, or future time point can be written as a nonlinear function of the bits of $N^{(t)}$, where the nonlinear function varies with t. For a few time points near t, the nonlinear function has a simple structure, but becomes more and more complex at time points further away from t. The requirement is to solve such a system of nonlinear equations in κ_1 variables to obtain $N^{(t)}$. Since the size of the LFSR is κ_2, to obtain a vaild attack it is required to solve the system of nonlinear equations using less than $2^{\kappa-\kappa_2}$ operations. For the W-series proposals that we have put forward, the values of $\kappa - \kappa_2$ are 16, 32, 48 for the 128-bit, 192-bit and the 256-bit security levels, and the corresponding sizes of the NFSR are 128, 192, and 256 bits respectively. We do not see how to solve the system of nonlinear equations in κ_1 variables in time less than $2^{\kappa-\kappa_2}$. Nonetheless, this is an attack avenue which may be investigated futher.

Multi-target attacks. For the W-series of stream ciphers, since κ_2 is less than κ, a possible multi-target attack is the following. If the attacker obtains keystream segments from 2^{κ_2} combinations of secret key and IV, then almost certainly for one of the targets, the state of the LFSR after initialisation, i.e. $L^{(0)}$ will be the all-zero value. As a result, the LFSR will play no part in the generation of the keystream and the entire keystream segment will be determined entirely by the NFSR. This opens up the possibility that the attacker obtains the state of the NFSR for the corresponding target using less than 2^{κ} operations. The difficulty with this approach is that the attacker does not actually know for which target the LFSR state is the all-zero value. We do not see any efficient way of determining this condition. Also, 2^{κ_2} is quite high, and it is not practical for an attacker to obtain keystream segments from 2^{κ_2} combinations of key and IV. Nevertheless, this is also an attack avenue which may need to be further studied.

9 Conclusion

In this paper, we provided an abstract description of the Grain family of stream ciphers which formalised the different components of the family. We proposed strengthened definitions of the components which improve security. Seven new stream cipher proposals were proposed at various security levels. We have argued that the new proposals provide good and improved resistance to known attacks on previous members of the Grain family. It is our hope that the present work will spur further research into the Grain family and more specifically into the security of the concrete proposals that we have put forward in this paper.

Acknowledgement

We would like to thank Alfred Menezes for helpful comments on Appendix B and for pointing out [72].

References

[1] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *Int. J. Wirel. Mob. Comput.*, 5(1):48–59, 2011. 1, 2, 10, 21

[2] Frederik Armknecht and Vasily Mikhalev. On lightweight stream ciphers with shorter internal states. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 451–470. Springer, 2015. 1

[3] Steve Babbage. Improved 'exhaustive search' attacks on stream ciphers. In *Proceedings of the European Convention on Security and Detection*, pages 161–166, May 1995. 35

[4] Steve Babbage and Matthew Dodd. The MICKEY stream ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 191–209. Springer, 2008. 1, 2

[5] Subhadeep Banik. Some insights into differential cryptanalysis of Grain v1. In Willy Susilo and Yi Mu, editors, *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*, volume 8544 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2014. 2, 34

[6] Subhadeep Banik. Conditional differential cryptanalysis of 105 round Grain v1. *Cryptogr. Commun.*, 8(1):113–137, 2016. 2, 34

[7] Subhadeep Banik, Daniel Collins, and Willi Meier. Near collision attack against Grain v1. In Mehdi Tibouchi and Xiaofeng Wang, editors, *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part I*, volume 13905 of *Lecture Notes in Computer Science*, pages 178–207. Springer, 2023. 2, 35

[8] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A differential fault attack on grain-128a using MACs. In Andrey Bogdanov and Somitra Kumar Sanadhya, editors, *Security, Privacy, and Applied Cryptography Engineering - Second International Conference, SPACE 2012, Chennai, India, November 3-4, 2012. Proceedings*, volume 7644 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2012. 2, 36

[9] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A differential fault attack on the Grain family of stream ciphers. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 122–139. Springer, 2012. 2, 36

[10] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A differential fault attack on the Grain family under reasonable assumptions. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata,*

India, December 9-12, 2012. Proceedings, volume 7668 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2012. 2, 36

[11] Subhadeep Banik, Subhamoy Maitra, Santanu Sarkar, and Meltem Sönmez Turan. A chosen IV related key attack on Grain-128a. In Colin Boyd and Leonie Simpson, editors, *Information Security and Privacy - 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings*, volume 7959 of *Lecture Notes in Computer Science*, pages 13–26. Springer, 2013. 2, 36

[12] Côme Berbain and Henri Gilbert. On the security of IV dependent stream ciphers. In Alex Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*, pages 254–273. Springer, 2007. 26

[13] Côme Berbain, Henri Gilbert, and Alexander Maximov. Cryptanalysis of Grain. 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 15–29. Springer, 2006. 2, 28

[14] Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2000. 35

[15] Alex Biryukov, Adi Shamir, and David A. Wagner. Real time cryptanalysis of A5/1 on a PC. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2000. 35

[16] Paul Camion, Claude Carlet, Pascale Charpin, and Nicolas Sendrier. On correlation-immune functions. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 1991. 14

[17] Christophe De Cannière, Özgül Küçük, and Bart Preneel. Analysis of Grain's initialization algorithm. In Serge Vaudenay, editor, *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, volume 5023 of *Lecture Notes in Computer Science*, pages 276–289. Springer, 2008. 2, 21

[18] Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008. 1, 2

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

[20] Claude Carlet and Philippe Gaborit. On the construction of balanced boolean functions with a good algebraic immunity. In *Proceedings of the 2005 IEEE International Symposium on Information Theory, ISIT 2005, Adelaide, South Australia, Australia, 4-9 September 2005*, pages 1101–1105. IEEE, 2005. 14, 15

[21] Claude Carlet and Palash Sarkar. The nonlinear filter model of stream cipher redivivus. Cryptology ePrint Archive, Paper 2025/160, 2025. 3, 7

[22] Guilhem Castagnos, Alexandre Berzati, Cécile Canovas, Blandine Debraize, Louis Goubin, Aline Gouget, Pascal Paillier, and Stephanie Salgado. Fault analysis of Grain-128. In Mohammad Tehranipoor and Jim Plusquellic, editors, *IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, San Francisco, CA, USA, July 27, 2009. Proceedings*, pages 7–14. IEEE Computer Society, 2009. 2, 36

[23] Abhishek Chakraborty, Bodhisatwa Mazumdar, and Debdeep Mukhopadhyay. A combined power and fault analysis attack on protected Grain family of stream ciphers. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 36(12):1968–1977, 2017. 2, 36

[24] Debrup Chakraborty, Cuauhtemoc Mancillas-López, and Palash Sarkar. STES: A stream cipher based low cost scheme for securing stored data. *IEEE Trans. Computers*, 64(9):2691–2707, 2015. 27

[25] Donghoon Chang and Meltem Sönmez Turan. Recovering the key from the internal state of Grain-128AEAD. *IACR Cryptol. ePrint Arch.*, page 439, 2021. 12

[26] 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. 7, 16

[27] Deepak Kumar Dalai, Santu Pal, and Santanu Sarkar. A state bit recovery algorithm with TMDTO attack on Lizard and Grain-128a. *Des. Codes Cryptogr.*, 90(3):489–521, 2022. 2, 35

[28] Prakash Dey, Abhishek Chakraborty, Avishek Adhikari, and Debdeep Mukhopadhyay. Improved practical differential fault analysis of Grain-128. In Wolfgang Nebel and David Atienza, editors, *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*, pages 459–464. ACM, 2015. 2, 36

[29] Lin Ding and Jie Guan. Related key chosen IV attack on Grain-128a stream cipher. *IEEE Trans. Inf. Forensics Secur.*, 8(5):803–809, 2013. 2, 36

[30] Itai Dinur, Tim Güneysu, Christof Paar, Adi Shamir, and Ralf Zimmermann. An experimentally verified attack on full Grain-128 using dedicated reconfigurable hardware. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 327–343. Springer, 2011. 2, 34

[31] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009. 34

[32] Itai Dinur and Adi Shamir. Breaking Grain-128 with dynamic cube attacks. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February*

13-16, 2011, Revised Selected Papers, volume 6733 of *Lecture Notes in Computer Science*, pages 167–187. Springer, 2011. 2, 34

[33] Vahid Amin Ghafari and Honggang Hu. Fruit-80: A secure ultra-lightweight stream cipher for constrained environments. *Entropy*, 20(3):180, 2018. 1

[34] Jovan Dj. Golic. Cryptanalysis of alleged A5 stream cipher. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 1997. 35

[35] Kishan Chand Gupta and Palash Sarkar. Toward a general correlation theorem. *IEEE Trans. Inf. Theory*, 51(9):3297–3302, 2005. 45

[36] Matthias Hamann, Matthias Krause, and Willi Meier. LIZARD - A lightweight stream cipher for power-constrained devices. *IACR Trans. Symmetric Cryptol.*, 2017(1):45–79, 2017. 1, 10, 12

[37] Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128AEAD. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 466–495. Springer, 2020. 2, 34

[38] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. In *Proceedings 2006 IEEE International Symposium on Information Theory, ISIT 2006, The Westin Seattle, Seattle, Washington, USA, July 9-14, 2006*, pages 1614–1618. IEEE, 2006. 1, 2

[39] Martin Hell, Thomas Johansson, Alexander Maximov, Willi Meier, Jonathan Sönnerup, and Hirotaka Yoshida. Grain-128AEADv2 - a lightweight AEAD stream cipher. `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/grain-128aead-spec-final.pdf`, 2021. 1, 2

[40] Martin Hell, Thomas Johansson, Alexander Maximov, Willi Meier, and Hirotaka Yoshida. Grain-128AEADv2: Strengthening the initialization against key reconstruction. Cryptology ePrint Archive, Paper 2021/751, `https://eprint.iacr.org/2021/751`, 2021. 1, 2, 12

[41] Martin Hell, Thomas Johansson, and Willi Meier. Grain - a stream cipher for constrained environments. ECRYPT Stream Cipher Project Report 2005/001, `http://www.ecrypt.eu.org/stream`, 2005. 1, 2

[42] Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *Int. J. Wirel. Mob. Comput.*, 2(1):86–93, 2007. 1, 2

[43] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980. 35

[44] Jonathan J. Hoch and Adi Shamir. Fault analysis of stream ciphers. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 240–253. Springer, 2004. 36

[45] Jin Hong and Palash Sarkar. New applications of time memory data tradeoffs. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 353–372. Springer, 2005. 35

[46] Kazuya Imamura, Kazuhiko Minematsu, and Tetsu Iwata. Integrity analysis of authenticated encryption based on stream ciphers. *Int. J. Inf. Sec.*, 17(5):493–511, 2018. 27

[47] Sandip Karmakar and Dipanwita Roy Chowdhury. Fault analysis of Grain-128 by targeting NFSR. In Abderrahmane Nitaj and David Pointcheval, editors, *Progress in Cryptology - AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, volume 6737 of *Lecture Notes in Computer Science*, pages 298–315. Springer, 2011. 2, 36

[48] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of NLFSR-based cryptosystems. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2010. 2, 34

[49] Donald E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*. Addison-Wesley, 1981. 19

[50] Özgül Küçük. Slide resynchronization attack on the initialization of Grain 1.0. `http://www.ecrypt.eu.org/stream/papersdir/2006/044.ps`, 2006. 2, 21

[51] Satyam Kumar and Santanu Sarkar. Conditional TMDTO as a MILP instance. *IEEE Trans. Inf. Theory*, 69(5):3330–3346, 2023. 2, 35

[52] Yuseop Lee, Kitae Jeong, Jaechul Sung, and Seokhie Hong. Related-key chosen IV attacks on Grain-v1 and Grain-128. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008, Proceedings*, volume 5107 of *Lecture Notes in Computer Science*, pages 321–335. Springer, 2008. 2, 21, 36

[53] Michael Lehmann and Willi Meier. Conditional differential cryptanalysis of Grain-128a. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, *Cryptology and Network Security, 11th International Conference, CANS 2012, Darmstadt, Germany, December 12-14, 2012. Proceedings*, volume 7712, pages 1–11. Springer, 2012. 2, 34

[54] Haitao Li, Zhaoqi Liu, and Wenrong Li. Nonsingularity of Grain-like cascade feedback shift registers subject to fault attacks. *Sci. China Inf. Sci.*, 67(9), 2024. 2, 36

[55] Jun-Zhi Li and Jie Guan. Advanced conditional differential attack on Grain-like stream cipher and application on Grain v1. *IET Inf. Secur.*, 13(2):141–148, 2019. 2, 34

[56] Zhen Ma, Tian Tian, and Wen-Feng Qi. Conditional differential attacks on Grain-128a stream cipher. *IET Inf. Secur.*, 11(3):139–145, 2017. 2, 34

[57] Zhen Ma, Tian Tian, and Wen-Feng Qi. Improved conditional differential attacks on Grain v1. *IET Inf. Secur.*, 11(1):46–53, 2017. 2, 34

[58] Diana Maimuț and George Teșeleanu. New configurations of Grain ciphers: Security against slide attacks. In Peter Y. A. Ryan and Cristian Toma, editors, *Innovative Security Solutions for Information Technology and Communications - 14th International Conference, SecITC 2021, Virtual Event, November 25-26, 2021, Revised Selected Papers*, volume 13195 of *Lecture Notes in Computer Science*, pages 260–285. Springer, 2021. 21

[59] Alexander Maximov. Cryptanalysis of the "Grain" family of stream ciphers. In Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Paul Lin, Shiuhpyng Shieh, and Sushil Jajodia, editors, *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2006, Taipei, Taiwan, March 21-24, 2006*, pages 283–288. ACM, 2006. 2, 28

[60] Bodhisatwa Mazumdar, Debdeep Mukhopadhyay, and Indranil Sengupta. Construction of RSBFs with improved cryptographic properties to resist differential fault attack on grain family of stream ciphers. *Cryptogr. Commun.*, 7(1):35–69, 2015. 2, 36

[61] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 311–343. Springer, 2016. 3, 7, 17

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

[63] Vasily Mikhalev, Frederik Armknecht, and Christian Müller. On ciphers that continuously access the non-volatile key. *IACR Trans. Symmetric Cryptol.*, 2016(2):52–79, 2016. 1

[64] Oscar S. Rothaus. On "bent" functions. *J. Comb. Theory, Ser. A*, 20(3):300–305, 1976. 6

[65] Md. Iftekhar Salam, Thian Hooi Ooi, Luxin Xue, Wei-Chuen Yau, Josef Pieprzyk, and Raphaël C.-W. Phan. Random differential fault attacks on the lightweight authenticated encryption stream cipher Grain-128AEAD. *IEEE Access*, 9:72568–72586, 2021. 2, 36

[66] Palash Sarkar. Tweakable enciphering schemes from stream ciphers with IV. *IACR Cryptol. ePrint Arch.*, page 321, 2009. 27

[67] Palash Sarkar. A new multi-linear universal hash family. *Des. Codes Cryptogr.*, 69(3):351–367, 2013. 27

[68] Palash Sarkar. Modes of operations for encryption and authentication using stream ciphers supporting an initialisation vector. *Cryptogr. Commun.*, 6(3):189–231, 2014. 27

[69] Palash Sarkar. Constructions of efficiently implementable Boolean functions with provable nonlinearity/resiliency/algebraic immunity trade-offs. *IACR Cryptol. ePrint Arch.*, page 1800, 2025. 13, 14

[70] 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. 7, 14, 15

[71] Santanu Sarkar, Subhadeep Banik, and Subhamoy Maitra. Differential fault attack against Grain family with very few faults and minimal assumptions. *IEEE Trans. Computers*, 64(6):1647–1657, 2015. 2, 36

[72] John Schanck. Improving post-quantum cryptography through cryptanalysis (PhD thesis). `https://dspacemainprd01.lib.uwaterloo.ca/server/api/core/bitstreams/1881e34e-5255-40ac-858c-3a63bb92d8f3/content`, 2020. 38, 46

[73] Akhilesh Siddhanti, Santanu Sarkar, Subhamoy Maitra, and Anupam Chattopadhyay. Differential fault attack on Grain v1, ACORN v3 and Lizard. In Sk Subidh Ali, Jean-Luc Danger, and Thomas Eisenbarth, editors, *Security, Privacy, and Applied Cryptography Engineering - 7th International Conference, SPACE 2017, Goa, India, December 13-17, 2017, Proceedings*, volume 10662 of *Lecture Notes in Computer Science*, pages 247–263. Springer, 2017. 2, 36

[74] Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Inf. Theory*, 30(5):776–780, 1984. 14

[75] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 250–279. Springer, 2017. 2, 34

[76] Yosuke Todo, Takanori Isobe, Willi Meier, Kazumaro Aoki, and Bin Zhang. Fast correlation attack revisited - cryptanalysis on full Grain-128a, Grain-128, and Grain-v1. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 129–159. Springer, 2018. 2, 4, 26, 28, 29, 32

[77] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 275–305. Springer, 2018. 2, 34

[78] Bin Zhang, Zhenqi Li, Dengguo Feng, and Dongdai Lin. Near collision attack on the Grain v1 stream cipher. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 518–538. Springer, 2013. 2, 35

[79] Bin Zhang, Chao Xu, and Willi Meier. Fast near collision attack on the Grain v1 stream cipher. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 771–802. Springer, 2018. 2, 35

A Some Relevant Results on Correlation

The generalised convolution theorem [35] expresses the Walsh transform of a finite sum of Boolean functions in terms of the Walsh transforms of the individual Boolean functions. Below we restate the result in terms of correlations.

Theorem 3 (Theorem 2 of [35]) *Let $h_1, \ldots, h_k$ be n-variable Boolean functions and $h(\mathbf{X}) = h_1(\mathbf{X}) \oplus \cdots \oplus h_k(\mathbf{X})$. Then for all $\mathbf{u} \in \mathbb{F}_2^n$,*

$$\mathsf{corr}_h(\mathbf{u}) \quad = \quad \sum_{\{\mathbf{u}_i \in \mathbb{F}_2^n : 1 \le i \le k-1\}} \prod_{i=1}^{k} \mathsf{corr}_{h_i}(\mathbf{u}_i \oplus \mathbf{u}_{i-1}), \tag{59}$$

where $\mathbf{u}_0 = \mathbf{0}_n$ and $\mathbf{u}_k = \mathbf{u}$.

The following simple results are easy to verify.

Proposition 18 *Let $\mathbf{X} = (X_1, \ldots, X_m)$, $\mathbf{Y} = (Y_1, \ldots, Y_n)$ and $f(\mathbf{X}, \mathbf{Y}) = g(\mathbf{Y})$, i.e. g is degenerate on $\mathbf{X}$. Then $\mathsf{corr}_f(\mathbf{0}_m, \boldsymbol{\beta}) = \mathsf{corr}_g(\boldsymbol{\beta})$, and $\mathsf{corr}_f(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$ for $\boldsymbol{\alpha} \neq \mathbf{0}_m$.*

Proposition 19 *Let $f(X_1, \ldots, X_n) = h(\psi(X_1, \ldots, X_n))$, where ψ is a bit permutation. Then $\mathsf{corr}_f(\boldsymbol{\alpha}) = \mathsf{corr}_h(\psi(\boldsymbol{\alpha}))$.*

Proposition 20 *Let $f(X_1, \ldots, X_{n_1}, Y_1, \ldots, Y_{n_2}) = g(X_1, \ldots, X_{n_1}) \oplus h(Y_1, \ldots, Y_{n_2})$. Then $\mathsf{corr}_f(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathsf{corr}_g(\boldsymbol{\alpha})\mathsf{corr}_h(\boldsymbol{\beta})$.*

B Bounds on Number of Keystream bits and Physical Memory

In this section, we provide (informal) justification for setting bounds on the number of keystream bits that should be generated from a single key and IV pair, and the number of bits of memory which may be considered physically possible.

Bound on the number of keystream bits generated from a single key and IV pair. Recall that we have set the maximum number of keystream bits that can be generated from a single key and IV pair to be 2^{64}. This may be contrasted with the standardised AES-GCM which permits $2^{32} - 2$ blocks (i.e. less than 2^{39} bits, since a block consists of 128 bits) to be encrypted/authenticated with a single key and IV pair.

To put the figure of 2^{64} in a realistic context, we note that the storage capacity of CERN crossed one exabyte (i.e. $2^3 \cdot 10^{18} \approx 2^{62.8}$ bits) in September 2023[1]. This data is stored at multiple data centres across different countries. It is completely impractical to encrypt the whole CERN storage using a single key and IV pair. One good reason for the impracticality is that the keystream generation does not support random access, i.e. it is not possible to generate later portions of the keystream without generating the earlier portions. So a single monolithic encryption of a huge number of bits using a single key and IV pair will create tremendous inefficiencies in encrypting and decrypting portions that are not close to the start of the message. One of the advantages of using an IV is to avoid such inefficiencies. Using the same secret key, different IVs can be used to encrypt (and decrypt) disjoint reasonable size segments of the message.

[1] `https://home.cern/news/news/computing/exabyte-disk-storage-cern`, (accessed on 7th October, 2025)

Bound on the memory. The number of atoms in the world[2] is about $1.33 \cdot 10^{50} \approx 2^{166.5}$. So any attack requiring memory around or more than this figure is clearly physically impossible. We next provide two informal arguments as to why even memory around 2^{100} or more bits is physically impossible.

The first argument goes as follows. As mentioned above, the storage capacity of CERN crossed one exabyte in 2023. It is reported that Google has 10 to 15 exabytes of storage, and AWS possibly has more storage. Suppose we consider 100 exabytes (i.e. $100 \cdot 2^3 \cdot 10^{18} \approx 2^{69.4}$ bits) of storage. The question is how much physical space is required to store this amount of data? CERN data, as well as Google and AWS data are stored at multiple data centres across the world, making it difficult to estimate the physical area required to store the entire $2^{62.8}$ bits of data. For the sake of concreteness, we make the ad-hoc assumption that the entire 100 exabyte data can be physically stored in a building whose surface area is equal to the size of a FIFA football field which is a 105m $\times$ 68m rectangle[3]. The surface area of Antartica is about 14.2 million square kms[4], and so about $2^{30.9}$ buildings whose surface area is equal to that of a FIFA football field will cover Antartica. If each of these buildings houses $2^{69.4}$ bits of data, then the total amount of data that can be stored in all of these buildings is about $2^{100.3}$ bits. It is clearly absurd to cover an area as large as Antartica in such a manner.

The second argument proceeds somewhat differently and is based on a similar argument appearing in Pages 13 to 15 of [72]. A micro SD card is of size 15mm $\times$ 11mm$\times$ 1mm. Let us suppose that a micro SD card can store 1 terabyte (i.e. $2^3 \cdot 10^{12} \approx 2^{42.9}$ bits). If we tile entire Antartica using micro SD cards, then the number of such cards will be about $2^{56.3}$. The total bits that can be stored in all of these cards is at most $2^{99.2}$ bits. It is clearly absurd to cover an area as large as Antartica with micro SD cards.

Both of the above arguments are based on the limitation of physical size and indicate that memory of 2^{100} or more bits is physically infeasible.

[2] `https://en.wikipedia.org/wiki/Atom`, accessed on 7th October, 2025

[3] `https://publications.fifa.com/de/football-stadiums-guidelines/technical-guideline/stadium-guidelines/pitch-dimensions-and-surrounding-areas/`, accessed on 7th October, 2025

[4] `https://en.wikipedia.org/wiki/Antarctica`, accessed on 7th October, 2025