Algebraic Attacks on Grain-like Keystream Generators

Matthew Beighton[1][0000−0002−7772−757X], Harry Bartlett[1][0000−0003−4347−0144], Leonie Simpson[1][0000−0001−8434−9741], and Kenneth Koon-Ho Wong[1][0000−0003−1732−6149]

Queensland University of Technology: Brisbane, Queensland, AU, matthew.beighton@hdr.qut.edu.au

Abstract. This paper analyses the resistance of certain keystream generators against algebraic attacks, namely generators consisting of a nonlinear feedback shift register, a linear feedback shift register and a filter function. We show that poorly chosen filter functions make such designs vulnerable to new algebraic attacks, using a divide and conquer approach that targets the LFSR first. This approach provides efficient LFSR initial state recovery followed by partial NFSR initial state recovery.

We apply our algebraic attacks to modified versions of the Grain family of stream ciphers. Our analysis shows that, despite the highly nonlinear filter functions used in these variants, the LFSR state can be recovered using our algebraic attack much faster than exhaustive search. Following this, the NFSR initial state can be partially recovered, leaving a smaller subset of NFSR stages to be exhaustively searched. This investigation highlights the importance of the filter function in keystream generators with a "Grain-like" structure, and demonstrates that many functions previously considered secure are vulnerable to this attack.

Keywords: Algebraic attack · fast algebraic attack · divide and conquer · nonlinear feedback shift register · linear feedback shift register · filter generator · Grain

1 Introduction

Symmetric stream ciphers are used to provide confidentiality for a range of real-time applications. The most common type of stream cipher is the binary additive stream cipher, where encryption and decryption are performed by XORing a binary keystream with the plaintext or ciphertext bitstream, respectively. The reciprocal nature of the XOR operation provides high speed encryption and decryption processes. However, the security provided depends on the properties of the keystream. The security of the keystream generator is therefore crucial.

Many keystream generators are based on shift registers. In 2003, algebraic attacks on keystream generators based on linear feedback shift registers (LFSRs) were proposed [6]. A series of equations relating the keystream bits to underlying state bits can be solved, given known keystream outputs. The nonlinear filter generator was shown to be vulnerable to both algebraic attacks [6] and fast algebraic attacks [5]. The design of keystream generators has subsequently evolved to provide increased resistance to these and other known attacks.

One alternative approach is to use nonlinear feedback shift registers (NFSRs) in the design of keystream generators. The nonlinearity of the shift register update results in a system of equations that increases in degree over time. Solving such a system of nonlinear equations has extreme computational complexity and is often impractical. However, in 2008, Berbain et al. [2] showed that keystream generators consisting of a NFSR with a linear filter function (known as linearly filtered nonlinear feedback shift registers (LF-NFSR)), are also vulnerable to algebraic attacks.

Some contemporary keystream generators use a combination of a NFSR and a LFSR, together with a nonlinear filter function taking inputs from both registers. We refer to these generators as "Grain-like structures", as the well known Grain family of stream ciphers is designed in this

way. There are five variants of Grain: the original proposal Grain-V0 [16], updated versions Grain-V1 [14], Grain-128 [15], Grain-128a [17], and the most recent variant Grain-128AEAD [18]; proposed in 2005, 2006, 2006, 2011 and 2019, respectively.

Berbain et al. [2] showed that a modified version of Grain-128 is vulnerable to algebraic attacks if the filter function takes only linear terms from the NFSR. Despite extensive analysis of the Grain family [3, 20, 7, 22, 21, 30, 29, 8], since the work of Berbain et al. [2] little progress has been made in applying traditional algebraic attacks to Grain-like structures. In this paper, we investigate relationships between the properties of Grain-like structures and their resistance to algebraic attacks.

Motivated by the work of Berbain et al. [2] and Courtois [5], we identify cases where the filter function to a Grain-like structure can be algebraically manipulated to eliminate the involvement of the NFSR. This allows us to apply a divide and conquer approach, targeting the LFSR. We show that poorly chosen filter functions and sets of input positions to the filter functions allow Grain-like structures to be represented as systems of equations containing only LFSR initial state bits. Such a system can then be solved efficiently to recover the initial state of the LFSR. Following this, the NFSR may be partially recovered. This reduces the overall complexity of the exhaustive search required to recover the remaining bits of the NFSR.

We perform simulations to verify the effectiveness of our attack on Grain-like structures with internal states sizes of 40, 60, 80 and 100 bits, as well as on a modified version of Grain-V0. We also demonstrate that minor variations to the current variants of the Grain family render them vulnerable to our attack. Although the proposed attack is not applicable to the current variants of the Grain family, our results clearly show that care must be taken when choosing the filter function of a Grain-like structure. In particular, these structures may still be vulnerable to algebraic attacks, even when the output function incorporates bits form the LFSR and the NFSR both linearly and nonlinearly.

This paper is organised as follows: Section 2 provides background information on shift register based designs. Section 3 discusses current algebraic attack techniques. Section 4 presents our algebraic attack technique for application to Grain-like structures. We introduce the Grain family of stream ciphers in Section 5 and highlight why Grain currently resists algebraic attacks. We then apply our attack technique to modified Grain variants in Section 6. Experimental simulations for proof of concept are reported in Section 7 and discussed in Section 8. Conclusions are drawn in Section 9.

2 Preliminaries

2.1 Feedback Shift Registers

A binary feedback shift register (FSR) of length n is a set of n storage devices called *stages* $(r_0, r_1, ..., r_{n-1})$, each containing one bit, together with a Boolean update function g. We denote the initial state of the register as S_0, consisting of initial state bits $s_0, s_1, ..., s_{n-1}$. The state at any time t is defined to be S_t, where $S_t = s_t, s_{t+1}, ..., s_{t+(n-1)}$. The sequence of state bits that passes through register R over time is denoted S; that is $S = s_0, s_1, .., s_{n-1}, s_n, ...$. All bits s_{n+t}, $t \geq 0$ are referred to as update bits.

The registers investigated in this paper are regularly clocked Fibonacci style, as shown in Figure 1. At time $t + 1$, the content of stage r_i is shifted to stage r_{i-1} for $1 \leq i \leq n - 1$, and the content of stage r_0 is lost from the register. The new content of stage r_{n-1} (referred to as

the feedback) is computed by applying the Boolean feedback function $g(r_0, r_1, .., r_{n-1})$ to S^t. If g is linear, then the register is said to be a linear feedback shift register (LFSR) and if g is nonlinear, then the register is said to be a nonlinear feedback shift register (NFSR).

A binary sequence can be generated from a FSR by applying a Boolean function f to the state S_t, as shown in Figure 1. A simple way to produce output from a FSR is to choose $f = r_0$, so the output sequence is just the sequence of bits shifted out of the register at each state update. Alternatively, the output can be a function of multiple register stages. In general, $y_t = f(s_t, ..., s_{t+n-1})$.

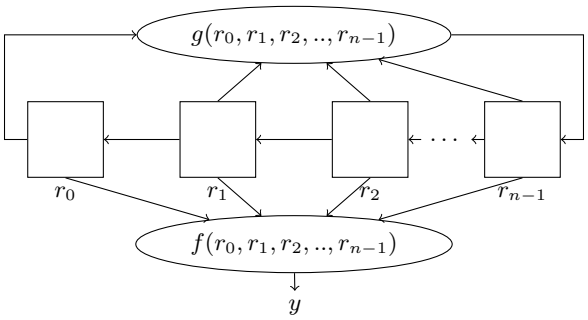

Fig. 1. An n-stage FSR with update function g and filter function f.

2.2 Filter Generators

Keystream generators where f is a function of the contents of multiple stages in R are called *filter generators*. Traditionally, keystream generators for stream ciphers used a LFSR together with a nonlinear filter function f [19]. If f is linear, the filter generator is equivalent to another LFSR. LFSRs, when used by themselves as keystream generators, provide very little security to the plaintext [23]. For this reason, LFSRs were traditionally filtered using a nonlinear Boolean function.

Nonlinear filter generators A keystream generator consisting of a LFSR and a nonlinear filter function f is known as a *nonlinear filter generator* (NLFG) [27]. The security of nonlinear filter generators has been extensively analysed. These designs have been shown to be susceptible to numerous attacks, including correlation attacks [27, 11, 24, 13], algebraic attacks [6, 10, 5] and distinguishing attacks [9]. The properties of the nonlinear filter function are important in determining the resistance of the NLFG to cryptanalysis. The underlying LFSR provides only desirable statistical properties for the binary sequence. As a single nonlinear Boolean function cannot display high levels of all the desirable cryptographic properties, such as high nonlinearity, correlation immunity, algebraic degree and algebraic immunity, choosing a filter function that resists one form of attack may leave the keystream generator vulnerable to other attacks [26].

Linearly filtered nonlinear feedback shift registers In response to the cryptanalysis of NLFGs, designs using NFSRs were proposed. One proposal is the dual construction of the nonlinear filter generator; where the update function g to the register is nonlinear and the filter function f is linear [12]. This is known as a linearly filtered nonlinear feedback shift register (LF-NFSR). Berbain et al. showed that LF-NFSRs are also susceptible to algebraic attacks, resulting in initial state (and possibly secret key) recovery [2]. We provide an overview of this attack in Section 3.3. From the work of Berbain et al. [2] it is obvious that the properties of the filter

function used in a LF-NFSR are critical in providing resistance to a traditional algebraic attack. In this paper, we explore the relationship between the properties of the filter function and the resistance to algebraic attacks for a more recent class of keystream generators.

2.3 Composite combiners and 'Grain-like' structures

Effective algebraic attacks have been proposed on both NLFG and LF-NFSR keystream generators. A more complex design incorporates both a LFSR and a NFSR, together with a nonlinear filter function taking inputs from both registers, as shown in Figure 2. Keystream generators using this structure include Grain [16] and subsequent variants of Grain [14, 15, 17, 18]. We denote this general design as a "Grain-like" structure. Other ciphers such as Sprout [1] and Plantlet [25] have adopted this structure. For simplicity, in this paper we consider the lengths of the NFSR and LFSR to be the same (n). However, the approach outlined also applies in the case where register lengths differ.

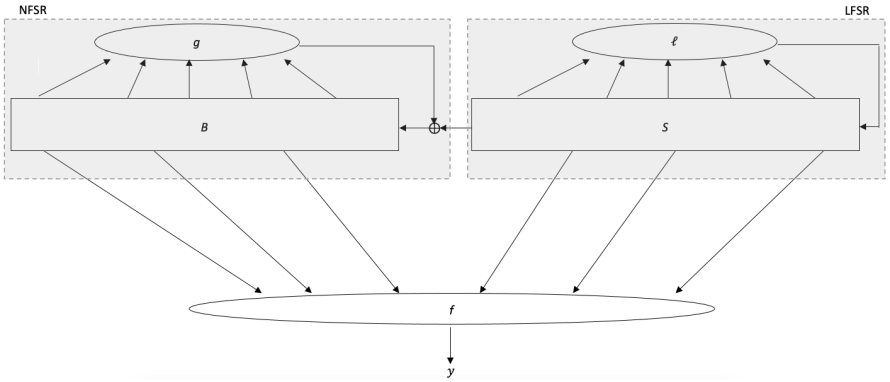

Fig. 2. Grain-like structure.

We denote the initial states of the NFSR and the LFSR as B_0 and S_0, respectively, and the initial state bits $b_0, b_1, ..., b_{n-1}$ and $s_0, s_1, ..., s_{n-1}$, respectively. The states of the registers at any time t are defined to be B_t and S_t. The sequences of state bits that pass through the registers over time are denoted B and S; that is $B = b_0, b_1, .., b_{n-1}, b_n, ...$ and $S = s_0, s_1, .., s_{n-1}, s_n, ...$ All bits b_{n+t} and s_{n+t}, $t \geq 0$ are referred to as update bits for either the NFSR or the LFSR. In the case of Grain-like structures, we denote the nonlinear update function as g, the linear update function as ℓ and the filter function as f. For a Grain-like structure, the LFSR is autonomous when producing output as all of the inputs to ℓ are from the LFSR. The NFSR is not autonomous, as the nonlinear update function g contains one input from the LFSR.

The filter function f can be considered as the XOR sum (here denoted '+') of several different types of monomials. That is, we consider sub-functions of f. We define the following sub-functions, each as a sum of the terms indicated:

- L_B - monomials with linear inputs from NFSR.
- L_S - monomials with linear inputs from LFSR.
- f_S - nonlinear monomials with inputs from LFSR only.
- f_B - nonlinear monomials with inputs from NFSR only.
- f_{BS} - nonlinear monomials with inputs from both NFSR and LFSR.

Thus, any filter function f in a Grain-like structure can be expressed as follows:

$$f(B, S) = L_B + L_S + f_B + f_S + f_{BS}. \tag{1}$$

For example, the output function for the keystream generator of Grain-128AEAD is as follows:

$$y = b_2 + b_{15} + b_{36} + b_{45} + b_{64} + b_{73} + b_{89} + s_{93} + b_{12}s_8 + s_{13}s_{20} + b_{95}s_{42} + b_{95}b_{12}s_{94} + s_{60}s_{79}.$$

This has the form $f(B, S) = L_B + L_S + f_S + f_{BS}$, where $L_B = b_2 + b_{15} + b_{36} + b_{45} + b_{64} + b_{73} + b_{89}$, $L_S = s_{93}$, $f_S = s_{13}s_{20} + s_{60}s_{79}$, $f_{BS} = s_8b_{12} + b_{95}s_{42} + b_{95}b_{12}s_{94}$ (and $f_B = 0$).

3 Current Algebraic Attacks

The goal of an algebraic attack is to create a system of low degree equations that relates the initial state bits of the cipher to some observed output bits and to solve these equations to recover the internal state values. For a binary additive stream cipher, the output may be obtained using a known-plaintext attack.

Algebraic attacks are performed in two phases: pre-computation and online. The pre-computation phase uses knowledge of the structure of the keystream generator to build a system of equations, relating initial state bits and output bits. These are represented as variables. In the online phase, given an observed output sequence, the appropriate substitutions are performed, the system is solved and the initial state recovered.

Algebraic attacks were first introduced by Courtois and Meier [6] on ciphers with linear feedback. An algebraic attack will be successful if the algebraic degree of the equations is low, and enough output can be observed to produce a meaningful solution.

3.1 Algebraic attacks on NLFGs

Courtois and Meier [6] built a system of equations to represent a NLFG using a simple approach: at time $t = 0$ the output bit y_0 is produced by applying f to the state S_0:

$$y_0 = f(S_0) = f(s_0, \ldots, s_{n-1})$$

Similarly, for every subsequent time step an equation can be formed.

$$y_t = f(S_t) = f(s_t, \ldots, s_{t+n-1}) \tag{2}$$

As the NLFG is built on a LFSR, the linear update function g is used to replace state bits s_{t+n-1} with the corresponding linear combination of initial state bits keeping the equation system of a constant degree ($deg(f)$), while maintaining the number of unknown variables in the system.

Courtois and Meier noted that, in many cases, each equation in the system may be multiplied through by a low degree multivariate function h (of degree e) to reduce the overall degree of the system of equations [6]. We refer to h as a reduction function, and using the terminology from [5] note that if $fh = 0$, then h is defined as an annihilator of f. Each equation in the resulting system has the following form:

$$f(S_t)h(S_t) = y_t h(S_t).$$

The degree of this system will be equal to $deg(fh) = d$, where $d < deg(f)$, with n independent variables, where n is the length of the underlying LFSR.

In the online phase, a known output sequence $\{y_t\}_{t=0}^{\infty}$ provides values for y_t to be substituted into the equations and the system is solved using linearisation. For a more detailed explanation,

the reader is referred to Courtois and Meier's paper [6].

The above considerations lead to the algorithms presented in Appendix A.1. In the online phase of the attack, the initial state of the LFSR can be recovered if approximately $\binom{n}{d}$ bits of output are known. This attack has a computational complexity of $\mathcal{O}(n\binom{n}{d} + \binom{n}{d}^{\omega})$, where d is the degree of the system and ω is the Guassian elimination exponent $\omega \approx 2.8$ [6]. If this output requirement cannot be met, it may be possible to solve the system by applying other methods for solving simultaneous equations, such as Gröbner bases or the XL algorithm [4].

3.2 Fast algebraic attacks on NLFGs

Following algebraic attacks on NLFGs, Courtois [5] showed that the attack could be improved if the overall degree of the system of equations could be reduced further, below the degree of fh. This increases the complexity of the precomputation phase, but greatly reduces the complexity of the online phase. However, precomputation only needs to be performed once for a particular cipher. This equation system may then be reused in multiple online phases to recover the states of the cipher corresponding to multiple different keystreams. These attacks are known as fast algebraic attacks.

Fast algebraic attacks make use of a concept Courtois [5] described as "double-decker equations". These equations allow an attacker to equate an expression in terms of initial state bits only to an expression in terms of initial state bits and observed output bits. The existence of "double-decker equations" in a system produced using Equation 2 permits all of the monomials in initial state bits only of degree from $e = deg(h)$ to $d = deg(fh)$ to be separated from the other monomials. Given approximately $\binom{n}{d}$ equations, the monomials of degree e to d will occur in multiple equations. Consequently, linear combinations of the equations in the system can be identified that cancel these monomials. These linear combinations define a new system in n unknowns, of degree $e < d$. This new system can be solved by linearisation, with less computational complexity than for traditional algebraic attacks. For a detailed explanation, the reader is referred to Courtois' paper [5]. The above considerations lead to the algorithm presented in Appendix A.2. Note that the online phase of the attack is the same as for regular algebraic attacks.

When the Berlekamp-Massey algorithm is used to find the linear dependency, the pre-computation phase of the attack has a computational complexity of $\mathcal{O}(\binom{n}{d} log(\binom{n}{d}))$ [5]. The initial state of the LFSR can be recovered in the online phase of the attack by observing approximately $\binom{n}{d}$ bits of output with a computational complexity of $\mathcal{O}(\binom{n}{d}\binom{n}{e} + \binom{n}{e}^{\omega})$, where d is the degree of fh, e is the degree of h and $\omega \approx 2.8$ [5]. Note that at first glance the online complexities for an algebraic attack and a fast algebraic attack look similar. However, when n is much larger than d, as is the case with registers used in practice, $\binom{n}{d}^{\omega}$ is much larger than $\binom{n}{d}\binom{n}{e}$ and $\binom{n}{e}^{\omega}$. Thus, by reducing the degree from d to e, the complexity of the online phase is drastically reduced for registers of practical size.

3.3 Algebraic attacks on LF-NFSRs

Initially, LF-NFSRs were considered resistant to algebraic attacks, due to the use of a nonlinear state update function. Using the nonlinear feedback function to derive equations for the update bits in terms of initial state bits causes the degree of the system of equations to increase over time. However, Berbain et al. [2] showed that it is possible to keep the degree of the system of equations constant by rearranging the linear output function to represent an initial state bit as a linear combination of other initial state bits and an output bit. Thus, an algebraic

attack can be performed. Using this approach, the initial state of the underlying NFSR (and possibly the secret key) can be recovered. We provide an overview of Berbain's attack below.

Berbain's preliminary observation: Consider y_0, the output at time $t = 0$ produced by applying of the linear filter function to the LF-NFSR contents:

$$y_0 = \ell(s_0, ..., s_{n-1}) = \sum_{k=0}^{n-1} a_k s_k,$$

where a_k are coefficients from the set {0,1}.

There will exist a highest indexed term in this sum, s_j, such that $a_j = 1$. Thus, we can write y_0 as

$$y_0 = s_j + \sum_{k=0}^{j-1} a_k s_k.$$

We can rearrange this equation to represent s_j as a sum of an output bit and initial state bits,

$$s_j = \sum_{k=0}^{j-1} a_k s_k + y_0.$$

Repeating this process for all subsequent time steps allows us to express every bit, s_{j+t} for $t \geq 0$, in terms of output bits and initial state bits. This produces a set of equations of the form:

$$s_{j+t} = \sum_{k=0}^{j-1} a_{k+t} s_{k+t} + y_t,$$

for $t \geq 0$. Note that if the latter summations contain any term for which an equation already exists, the term can be replaced by the corresponding linear combination of initial state bits and output bits.

The above considerations lead to the algorithm presented in Appendix A.3. For certain update functions a reduction function of g, say h, may be used to reduce the overall degree of the system. If the degree of gh is d, then the overall system will be of degree at most d. The initial state of the LF-NFSR can be recovered in the online phase of the attack by observing approximately $\binom{n}{d}$ bits of output with a computational complexity of $\mathcal{O}(n\binom{n}{d} + \binom{n}{d}^{\omega})$, where d is the degree of the system and $\omega \approx 2.8$ [2]. Note that fast algebraic techniques are not applicable to LF-NFSRs.

3.4 Algebraic attacks on Grain-like structures

After successfully applying algebraic attacks to LF-NFSRs, Berbain et al. proposed an algebraic attack on Grain-like structures where the output function f is the XOR combination of a LF-NFSR and a NLFG. That is, adopting the notation from Section 2.3, $f(B, S) = L_B + L_S + f_S$.

In this case the output of the keystream generator can be expressed as

$$y_0 = L_B + L_S + f_S = \sum_{k=0}^{n-1} a_k b_k + L_S + f_S. \tag{3}$$

As discussed in Section 3.3, there exists a highest indexed term in L_B (which we denote as b_j). Thus,

$$y_0 = b_j + \sum_{k=0}^{j-1} a_k b_k + L_S + f_S,$$
$$b_j = \sum_{k=0}^{j-1} a_k b_k + L_S + f_S + y_0.$$

Repeating this for $t > 0$ allows for NFSR state bits of index j or higher to be represented as the XOR sum of:

- a linear combination of NFSR initial state bits
- a linear and nonlinear combination of LFSR initial state bits
- a linear combination of observed output bits.

A second system of equations can then be built using the nonlinear update function to the NFSR, making substitutions from the system generated by Equation 3 where applicable. This system will be of degree at most $deg(g)deg(f_S)$. Combining the two systems results in a system of equations of degree $deg(g)deg(f_S)$ in $n + j$ unknown initial state bits, where n is the size of LFSR and j is the index of the highest indexed term in L_B.

The success of this attack in recovering the LFSR and NFSR initial states demonstrated that using the contents of stages in the NFSR linearly in the output function is not sufficient to provide resistance to algebraic attacks; the NFSR contents must also be filtered nonlinearly in some way.

4 Our algebraic attacks on Grain-like structures

Currently, Grain-like structures are considered resistant to algebraic attacks if the filter function is chosen to incorporate inputs from both the LFSR and NFSR, and includes these both linearly and nonlinearly. In this paper, we show that this is insufficient. The filter function must be chosen more carefully than previously considered. We show that Grain-like structures designed with filter functions that use inputs from the LFSR and NFSR both linearly and nonlinearly may still be susceptible to algebraic attacks. We consider the filter functions used in Grain-like structures and identify possible combinations of the sub-functions, each containing some (or all) of the monomials shown in Equation 1. For each case we examine the applicability of algebraic attacks.

4.1 Vulnerable filter functions

The algebraic attack presented by Berbain et al. [2] applies to the case where $f(B, S) = L_B + L_S + f_S$. As the success of this case is already documented, we omit this from our paper and investigate four additional cases as follows.

Case 1: $f(B, S) = L_S + f_S + f_{BS}$ Consider a keystream generator that produces an output bit at each time step by:

$$z = L_S + f_S + f_{BS}. \tag{4}$$

That is, NFSR state bits are only used nonlinearly and only in f_{BS}. Every monomial in f_{BS} will contain both NFSR bits and LFSR bits. Thus, using the idea of annihilators presented by Courtois and Meier [6], we may multiply Equation 4 by a low degree function containing

only LFSR bits that will eliminate f_{BS}. We denote this function as A_{BS}, and consider it to be a "partial annihilator" in as much as A_{BS} only annihilates certain monomials. Note that the degree of the NFSR bits in f_{BS} does not affect the ability to annihilate the monomials containing bits from the NFSR.

Therefore Equation 4 can be rewritten as

$$zA_{BS} = A_{BS}(L_S + f_S), \tag{5}$$

which is an equation containing only LFSR initial state bits. The degree of the system of equations built using Equation 5 will be at most $deg(A_{BS}) + deg(f_S)$. Note, however, that the right hand side of Equation 5 contains only initial state bits from the LFSR. This means that fast algebraic attack methods can be performed in the precomputation phase of the attack to reduce the degree of unknown variables in the system from $deg(A_{BS})+deg(f_S)$ to $deg(A_{BS})$.

To illustrate this point, consider the filter function

$$z = x_1 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_2x_3x_4, \tag{6}$$

where x_0, x_1, x_2, x_3 are inputs from the LFSR and x_4 is taken from the NFSR. Equation 6 may be rewritten as

$$z = x_1 + x_0x_3 + x_0x_1x_2 + x_3x_4(1 + x_2).$$

Multiplying through by x_2 gives:

$$x_2z = x_1x_2 + x_0x_1x_2 + x_0x_2x_3,$$

since $x_2(1 + x_2) = 0$.

The right hand side of this equation now contains only inputs from the LFSR and can therefore be used to mount an algebraic attack on the LFSR. Further processing (clocking the equation forward and taking suitable linear combinations of the resulting equations to eliminate all right hand side terms of degree higher than $deg(A_{BS}) = 1$) would then allow a fast algebraic attack on the LFSR, with much lower online complexity than the original algebraic attack.

Case 2: $f(B, S) = L_B + L_S + f_S + f_{BS}$ The simple idea presented for Case 1 can be extended to the case where $f(B, S) = L_B + L_S + f_S + f_{BS}$, under certain conditions. If the filter function is chosen such that each monomial in L_B uniquely divides a monomial in f_{BS} of degree at most 1 in NFSR bits, then a common factor containing only LFSR bits may be found. It follows that we can annihilate every monomial containing inputs from the NFSR by multiplying through by a low degree function of LFSR initial state bits.

For example, consider a keystream bit produced by

$$z = x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_2x_3x_4, \tag{7}$$

where x_0, x_1, x_2, x_3 are inputs from the LFSR and x_4 is input from the NFSR. This is the function used in the previous example, with the inclusion of x_4 used linearly. We now have:

$$z = x_1 + x_0x_3 + x_0x_1x_2 + x_4(1 + x_3 + x_2x_3).$$

Multiplying through by $x_3 + x_2x_3$ gives:

$$(x_3 + x_2x_3)z = (x_3 + x_2x_3)(x_1 + x_0x_3 + x_0x_1x_2).$$

As before, the right hand side of this equation now contains only inputs from the LFSR.

Case 3: $f(B,S) = L_S + f_S + f_B + f_{BS}$ Case 3 is a simple extension of Case 2. That is, if each of the monomials used in f_B is a factor of some monomial used in f_{BS} then a common factor containing only LFSR bits may again be found and a partial annihilator obtained.

For example, consider a keystream bit produced by

$$z = x_1 + x_4x_5 + x_0x_3 + x_0x_1x_2 + x_2x_3x_4x_5, \tag{8}$$

where x_0, x_1, x_2, x_3 are inputs from the LFSR and x_4, x_5 are inputs from the NFSR. We now have:

$$z = x_1 + x_0x_3 + x_0x_1x_2 + x_4x_5(1 + x_2x_3).$$

Multiplying through by x_2x_3 gives:

$$(x_2x_3)z = (x_2x_3)(x_1 + x_0x_3 + x_0x_1x_2).$$

As before, the right hand side of this equation now contains only inputs from the LFSR.

Case 4: $f(B,S) = L_B + L_S + f_S + f_B + f_{BS}$ Case 4 is a filter function that contains all possible monomial types (monomials that are linear and nonlinear in both the LFSR and the NFSR bits). If the filter function is chosen such that each monomial in L_B and f_B uniquely divides a monomial in f_{BS}, then there exists a partial annihilator that will eliminate L_B, f_B and f_{BS}.

For example, consider a keystream bit produced by

$$z = x_1 + x_4 + x_5 + x_0x_3 + x_0x_1x_2 + x_2x_4 + x_3x_5 + x_6x_7 + x_1x_2x_6x_7, \tag{9}$$

where x_0, x_1, x_2, x_3 are inputs from the LFSR and x_4, x_5, x_6, x_7 are inputs from the NFSR. We now have:

$$z = x_1 + x_4 + x_5 + x_0x_3 + x_0x_1x_2 + x_2x_4 + x_3x_5 + x_6x_7 + x_1x_2x_6x_7$$
$$z = x_1 + x_4(1 + x_2) + x_5(1 + x_3) + x_0x_3 + x_0x_1x_2 + x_2x_4 + x_3x_5 + (1 + x_1x_2)x_6x_7.$$

Multiplying through by $x_1x_2x_3$ gives:

$$(x_1x_2x_3)z = (x_1x_2x_3)(x_1 + x_0x_3 + x_0x_1x_2 + x_2x_4 + x_3x_5).$$

As before, the right hand side of this equation now contains only inputs from the LFSR.

4.2 Generalised algebraic attack algorithm

We present here an attack algorithm based on the analysis above. This attack uses a divide and conquer strategy. We first target the LFSR and recover the LFSR initial state. The NFSR is then targeted, with partial NFSR intitial state recovery possible.

Recovering the LFSR We show that if an output function to a Grain-like structure fits any of the cases discussed in Section 4.1, then a system of equations can be developed that relates observable output bits to just LFSR initial state bits, and does not increase in algebraic degree over time.

The structure of a system of equations built in this way allows for the fast algebraic attack

techniques highlighted in Section 3.2 to be applied. That is, given access to approximately $\binom{n}{d}$ bits of output (where n is the size of the LFSR and d is the algebraic degree of the system relating LFSR initial state bits to observable output bits), a precomputation phase can be performed that allows a new system of equations to be built of degree $e < d$, where e is the degree of A_{BS}. This precomputation phase has a complexity of $\mathcal{O}(\binom{n}{d} log \binom{n}{d} + n\binom{n}{d})$. The initial state of the LFSR can then be recovered in the online phase of the attack by observing approximately $\binom{n}{d}$ bits of output with complexity $\mathcal{O}(\binom{n}{d}\binom{n}{e}) + \binom{n}{e}^{\omega}$, where ω is the Guassian elimination exponent $\omega \approx 2.8$.

Recovering the NFSR Once the LFSR initial state is recovered, every future LFSR state bit will be known, as the LFSR is autonomous. The next stage is to recover the NFSR initial state.

If the output function satisfies Cases 2 or 4 from Section 4.1, then it may be possible to build a system of equations where each equation is linear in NFSR initial state bits. It is likely, however, that for the cases covered in this paper this approach is not applicable. That is, the structure of the functions that allow us to target the LFSR inhibits the ability to build such a system for the NFSR state bits. In these cases, a different approach may be applied to recover the NFSR contents. The idea is best illustrated through an example.

Consider the example filter function used in Case 3 of Section 4.1. That is, at each time step a keystream bit is produced by

$$z = x_1 + x_4x_5 + x_0x_3 + x_0x_1x_2 + x_2x_3x_4x_5, \tag{10}$$

where x_0, x_1, x_2, x_3 are from the LFSR and x_4, x_5 are from the NFSR. Since the LFSR is known, each output bit will have the form

$$z = \alpha x_4x_5 + \beta,$$

where α and β may be 0 or 1, respectively.

Clearly, when $\alpha = 0$ no information about the initial state of the NFSR is leaked. We must therefore utilise the case where $\alpha = 1$. If $z = x_4x_5$ and $z = 1$, then we know $x_4 = x_5 = 1$. Likewise if $z = x_4x_5 + 1$ and $z = 0$, then we know $x_4 = x_5 = 1$. Once we have recovered these state bits, we may then look to equations where $z = x_4x_5$ and $z = 0$, but for which we know either x_4 or x_5 equals 1. We would then know that the unknown state bit is equal to zero. Similarly for the case where $z = x_4x_5 + 1$ and $z = 1$. Continuing in this way, we may be able to recover n consecutive bits of the NFSR. It may then be possible to reverse the NFSR update and therefore recover the NFSR initial state.

For certain filter functions it may not be possible to recover n consecutive state bits. In this case, the partially recovered initial state reduces the exhaustive search required to recover the correct initial state of the NFSR. For instance, suppose m bits of the NFSR can be recovered. This leaves 2^{n-m} possible candidates for the correct NFSR initial state which, for $m > 0$, is better than exhaustively searching the entire register. Each candidate can be used (together with the known LFSR initial state) to produce output. The candidate which produces the correct output sequence can be assumed to be the correct initial state.

Once the correct LFSR and NFSR initial states are recovered, it may be possible to reverse the initialisation process and reveal the secret key.

5 The Grain Family of Stream Ciphers

We now focus on the well known Grain family of stream ciphers. In the following sections we provide an overview of the functions used in the family and highlight why the variants currently resist traditional algebraic attacks. We then demonstrate how an algebraic attack is possible on modified versions of each of the Grain variants.

5.1 Description of the Grain

Grain is a well known family of stream ciphers that has been extensively analysed. There are five main variants: Grain-V0 [16], its revised specification Grain-V1 [14], Grain-128 [15], Grain-128a [17] and Grain-128AEAD [18]. Grain-V0, Grain-V1 and Grain-128 provide only confidentiality to a plaintext message. Grain-128a provides confidentiality, or confidentiality and integrity assurance. Grain-128AEAD provides confidentiality and integrity assurance to every plaintext message.

Keystream generators in the Grain family are based on three main building blocks: a nonlinear feedback shift register (NFSR), a linear feedback shift register (LFSR) and a nonlinear filter function. The length of the registers changes depending on key and initialisation vector (IV) size. Table 1 provides the sizes of the key, IV and component registers for each variant.

Table 1. Inputs, register lengths and internal state sizes of Grain variants.

Variant	Key (bits)	IV (bits)	NFSR length (bits)	LFSR length (bits)	Internal state (bits)
Grain-V0	80	64	80	80	160
Grain-V1	80	64	80	80	160
Grain-128	128	96	128	128	256
Grain-128a	128	96	128	128	256
Grain-128AEAD	128	96	128	128	256

5.2 Phases of Operation

Grain variants can be partitioned into two classes: those that are capable of authentication and those that are not. The phases of operation among the Grain variants are all similar. Each variant produces an initial state by loading a secret key and public IV, and then clocking the cipher a certain number of times. For Grain-128AEAD, this initialisation process uses the secret key to make reversing the initialisation process difficult.

Processing the plaintext Once the initialisation phase is complete, the output from the Grain keystream generator is used to encrypt plaintext to form ciphertext. When processing the plaintext, Grain-V0, Grain-V1, Grain-128 and Grain-128a (not in authentication mode) produce keystream z, which is XORed with the plaintext p to produce ciphertext c, where $c = z + p$. For Grain-128a (with authentication) and Grain-128AEAD, if the index of the output bit is even, then it is used as keystream and XORed with the plaintext to produce ciphertext. If the index of the output bit is odd, then the output is used to help generate the tag. We omit the process by which the tag is generated in this paper as it does not directly affect our findings.

5.3 Resistance against algebraic attacks

All of the Grain variants use filter functions of the same form. In particular, all of the variants have filter functions of the form $f(B, S) = L_B + L_S + f_S + f_{BS}$. Recall from Section 4.1 that a Grain-like structure that uses a filter function of this form will be susceptible to an algebraic attack if each of the monomials in L_B is a factor of some monomial used in f_{BS}. Each Grain variant, however, uses terms in L_B that do not appear in any other monomial. This eliminates the ability to annihilate these terms from the NFSR and thus defeats our attack.

6 Algebraic Attack on Modified Versions of the Grain Family

We now mount an algebraic attack on adapted versions of the Grain family of stream ciphers with modified filter functions. We show that even with bits from the NFSR used nonlinearly in f, a system of equations that does not increase in degree over time can be constructed, enabling successful fast algebraic attacks on the modified variants of V0, V1, 128 and also 128a (without the authentication). We also mount algebraic attacks on the variants with authentication (128a and 128AEAD). We note that recovering the LFSR initial state is the same for all variants, but on average less of the NFSR initial state will be recovered for variants with authentication due to the decimation used on the output when generating keystream.

6.1 Modified version of Grain

We introduce a modified version of the Grain family, where we replace any independent linear term taken from the NFSR by the corresponding term in the LFSR. That is, all filter functions $f(B, S)$ remain the same, except that monomials appearing only in L_B are replaced by monomials in L_S with the same indices. Note that we denote a modified version of Grain by appending the suffix $-m$.

Table 2 highlights the differences between original and modified versions of each Grain variant. To save space, we present the modifications for Grain-V0 in detail here and refer the interested reader to Appendix B for details of the other variants.

Table 2. Modifications to linear combinations in different Grain variants.

Variant	Original linear combination	Modified linear combination
V0	b_0	s_1
V1	$b_1 + b_2 + b_4 + b_{10} + b_{31} + b_{43} + b_{56}$	$s_1 + s_2 + s_4 + s_{10} + s_{31} + s_{43} + s_{56}$
128	$b_2 + b_{15} + b_{36} + b_{45} + b_{64} + b_{73} + b_{89} + b_{93}$	$s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93}$
128a	$b_2 + b_{15} + b_{36} + b_{45} + b_{64} + b_{73} + b_{89} + s_{93}$	$s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93}$
128AEAD	$b_2 + b_{15} + b_{36} + b_{45} + b_{64} + b_{73} + b_{89} + s_{93}$	$s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93}$

Grain-V0$-m$ In the case of Grain-V0-m, the bit b_0 was replaced by s_1 so as to not use the same bit used in the linear update of the LFSR.

For Grain-V0-m, the output function is as follows:

$$\begin{aligned} z = h(B, S) =& s_1 + s_3 + s_{25} + b_{63} + s_3 s_{64} + s_{46} s_{64} + s_{64} b_{63} + s_3 s_{25} s_{46} + s_3 s_{46} s_{64} \\ &+ s_3 s_{46} b_{63} + s_{25} s_{46} b_{63} + s_{46} s_{64} b_{63}. \end{aligned}$$

Note that b_{63} was left as a linear term in f of Grain-V0$-m$ as f satisfies the conditions of Case 2 covered in Section 4.1.

We show that for each variant, it is possible to build systems of equations that are independent of the NFSR. This means that the degree of the system of equations will not vary over time, despite the presence of NFSR bits in the filter function of each variant.

6.2 Stage 1: LFSR recovery

In this section we apply the algorithm from Section 4. We provide the details for Grain-V0 and Grain-128a-m/128AEAD-m. The details for the other variants can be found in Appendix C. The theoretical data and computational complexity requirements to recover the LFSR initial state for each variant are summarised in Table 3. In Section 7, we provide experimental results for the modified version of Grain-V0.

Grain-V0$-m$
At time $t = 0$ an output bit in Grain-V0-m is produced as follows:

$$z_0 = s_1 + s_3 + s_{25} + s_3s_{64} + s_{46}s_{64} + s_3s_{25}s_{46} + s_3s_{46}s_{64} + b_{63}(1 + s_{64} + s_3s_{46} + s_{25}s_{46} + s_{46}s_{64})$$

Multiplying this equation by $(s_{64} + s_3s_{46} + s_{25}s_{46} + s_{46}s_{64})$ gives

$$\begin{aligned}(s_{64} + s_3s_{46} + s_{25}s_{46} + s_{46}s_{64})z_0 = &s_1s_3s_{46} + s_1s_{25}s_{46} + s_1s_{46}s_{64} + s_3s_{46}s_{64} + s_1s_{64} + s_3s_{46} \\ &+ s_{25}s_{46} + s_{25}s_{64}\end{aligned}$$

where the right hand side of the equation contains only LFSR initial state bits and is of degree 3. Thus, by observing at least $\binom{80}{3}$ keystream bits, fast algebraic techniques may be applied in the precomputation phase of the attack to reduce the overall degree of the system to the degree of the left hand side (which is of degree 2 in the unknown LFSR initial state bits) [5].

Grain-128a$-m$ (with authentication)/Grain-128AEAD$-m$
The process for recovering the LFSR initial state in Grain-128a-m (with authentication) and Grain128AEAD-m is very similar to Grain-128a-m (without authentication). In keystream generation, Grain-128a-m (with authentication) and Grain-128AEAD-m use a decimated version of the output sequence as keystream. Therefore, every second equation produced using the output function will contain an unknown output bit (used for authentication and so not visible in a known plaintext scenario). To avoid this problem, an attacker simply builds the system following the process for Grain-128a-m (without authentication), and then builds a new system by taking all the even indexed equations. Note that this requires the attacker to produce twice as many equations as they would for Grain-128a-m (without authentication), but still only requires the same amount of keystream output.

The highest degree monomial is of order 5 (see Appendix C). Thus, by observing at least $\binom{80}{5}$ keystream bits, fast algebraic techniques may be applied in the precomputation phase of the attack to reduce the overall degree of the system to the degree of the left hand side (which is of degree 3 in the unknown LFSR initial state bits) [5]. Note that due to the keystream decimation used in these variants, the precomputation phase requires a higher complexity (as discussed by Courtois [5]).

6.3 Stage 2: NFSR recovery

Once the LFSR initial state is recovered, the output function will contain only unknown initial state bits from the NFSR. Due to the structure of the output function for the Grain variants,

Table 3. Resource requirements for recovering the LFSR of the modified Grain variants.

Variant	Grain-V0-m	Grain-V1-m	Grain-128-m	Grain-128a-m (no authentication)	Grain-128a-m (authentication)/128AEAD
Precomputation phase					
Degree of system before applying fast algebraic techniques	3	3	5	5	5
Complexity	$\mathcal{O}(2^{19})$	$\mathcal{O}(2^{19})$	$\mathcal{O}(2^{31})$	$\mathcal{O}(2^{31})$	$\mathcal{O}(2^{78})$
Degree of system after applying fast algebraic techniques	2	2	3	3	3
Online phase					
Data	2^{17}	2^{17}	2^{28}	2^{28}	2^{28}
Complexity	$\mathcal{O}(2^{33})$	$\mathcal{O}(2^{33})$	$\mathcal{O}(2^{52})$	$\mathcal{O}(2^{52})$	$\mathcal{O}(2^{52})$

we adopt the second approach described in Section 4.2. We provide details for Grain-V0-m in this section. Details for the other variants can be found in Appendix D.

The data requirement for this stage will utilise the data collected for LFSR state recovery. The computational complexity is considered to be negligible [3]. The number of NFSR initial state bits recovered through application of this method is hard to estimate and will vary depending on the particular initial state. However, some guidance based on experimental results is provided in Section 7.4. Moreover, if at least one bit of the NFSR can be recovered, the remaining exhaustive search is better than exhaustively searching the key space. Due to the low computational complexity of partial NFSR recovery we provide experimental results for this in the following section.

Grain-V0$-m$
At time $t = 0$ an output bit in Grain-V0 is produced as follows:

$$\begin{aligned} z_0 =& s_1 + s_3 + s_{25} + s_3 s_{64} + s_{46} s_{64} + s_3 s_{25} s_{46} + s_3 s_{46} s_{64} \\ &+ b_{63}(1 + s_{64} + s_3 s_{46} + s_{25} s_{46} + s_{46} s_{64}) \end{aligned}$$

This function is linear in the single NFSR bit b_{63}. At each time step we have:

$$z = \alpha b_{63+t} + \beta,$$

where α and β can be 0 or 1, respectively.

When $\alpha = 1$, an NFSR initial state bit will be recovered. This can be used for simple partial state recovery of the NFSR. The remaining stages of the NFSR initial state can then be found through exhaustive search. An estimate of the average exhaustive search requirement for each modified Grain variant is provided in Table 5 of Section 7.4.

7 Experimental simulations

We have performed computer simulations of our algebraic attack, applying it to toy versions of Grain-like structures with total internal states of 40, 60, 80 and 100 bits, to demonstrate proof of concept. We have also performed simulations of the full modified version of Grain-V0. The details of these versions, the simulation setup and results are provided in the following sections. We also provide experimental results in Section 7.4 for the partial NFSR recovery of each of the full modified versions of Grain; this is possible because of the low time complexity required to partially recover these states.

7.1 Specifications

The toy Grain-like structures used in our simulations were formed using registers each of length 20, 30, 40 and 50 respectively. The details for the structures are as follows. Note that we use subscripts on the functions to distinguish between the registers of the different sizes. The modified version of Grain-V0 is denoted using the subscript 80.

The LFSR update functions correspond to primitive polynomials of the relevant order and are as follows:

$$\begin{aligned}
\ell_{20} &= s_0 + s_{11} + s_{15} + s_{17} \\
\ell_{30} &= s_0 + s_7 + s_{28} + s_{29} \\
\ell_{40} &= s_0 + s_{35} + s_{36} + s_{37} \\
\ell_{50} &= s_0 + s_{46} + s_{47} + s_{48} \\
\ell_{80} &= s_0 + s_{13} + s_{23} + s_{38} + s_{51} + s_{62}
\end{aligned}$$

The NFSR update functions $g_{20}, g_{30}, g_{40}, g_{50}$ are modified versions of the nonlinear update used in Sprout [1]. The modified functions use the same number of inputs and have the same algebraic degree of 4. The update function g_{80} is the same function used in Grain-V0. The nonlinear update functions are as follows:

$$\begin{aligned}
g_{20} &= s_0 + b_0 + b_{13} + b_{19} + b_{15} + b_2b_{15} + b_3b_5 + b_7b_8 + b_{14}b_{19} + b_{10}b_{11}b_{12} + b_6b_{13}b_{17}b_{18} \\
g_{30} &= s_0 + b_0 + b_{19} + b_{28} + b_{22} + b_4b_{22} + b_5b_7 + b_{11}b_{12} + b_{21}b_{28} + b_{15}b_{17}b_{19} + b_9b_{19}b_{25}b_{27} \\
g_{40} &= s_0 + b_0 + b_{27} + b_{38} + b_{28} + b_5b_{28} + b_6b_9 + b_{15}b_{14} + b_{31}b_{38} + b_{23}b_{25}b_{27} + b_{15}b_{27}b_{33}b_{37} \\
g_{50} &= s_0 + b_0 + b_{33} + b_{47} + b_{38} + b_6b_{38} + b_8b_{13} + b_{17}b_{20} + b_{35}b_{47} + b_{25}b_{28}b_{30} + b_{15}b_{33}b_{42}b_{45}
\end{aligned}$$

$$\begin{aligned}
g_{80} = {} & s_0 + b_0 + b_9 + b_{15} + b_{21} + b_{28} + b_{33} + b_{37} + b_{45} + b_{52} + b_{60} + b_{63} + b_{9)}b_{15} + b_{33}b_{37} + \\
& + b_{60}b_{63} + b_{21}b_{28}b_{33} + b_{45}b_{52}b_{60} + b_9b_{28}b_{45}b_{63} + b_{15}b_{21}b_{60}b_{63} + b_{33}b_{37}b_{52}b_{60} + b_9b_{15}b_{21}b_{28}b_{33} + \\
& + b_{37}b_{45}b_{52}b_{60}b_{63} + b_{21}b_{28}b_{33}b_{37}b_{45}b_{52}
\end{aligned}$$

The output functions used in the simulations are modified versions of the one used in Grain-V0 [16] with the same number of taps and algebraic degree. The output functions are as follows:

$$\begin{aligned}
f_{20} &= s_1 + s_2 + s_6 + s_2s_{16} + s_{12}s_{16} + s_2s_6s_{12} + s_2s_{12}s_{16} + b_{15}(s_{16} + s_2s_{12} + s_6s_{12} + s_{12}s_{16} + 1) \\
f_{30} &= s_1 + s_2 + s_9 + s_2s_{24} + s_{17}s_{24} + s_2s_9s_{17} + s_2s_{17}s_{24} + b_{23}(s_{24} + s_2s_{17} + s_9s_{17} + s_{17}s_{24} + 1) \\
f_{40} &= s_1 + s_2 + s_{13} + s_2s_{32} + s_{23}s_{32} + s_2s_{13}s_{23} + s_2s_{23}s_{32} + b_{31}(s_{32} + s_2s_{23} + s_{13}s_{23} + s_{23}s_{32} + 1) \\
f_{50} &= s_1 + s_2 + s_{16} + s_2s_{40} + s_{29}s_{40} + s_2s_{16}s_{29} + s_2s_{29}s_{40} + b_{39}(s_{40} + s_2s_{29} + s_{16}s_{29} + s_{29}s_{40} + 1) \\
f_{80} &= s_1 + s_3 + s_{25} + s_3s_{64} + s_{46}s_{64} + s_3s_{25}s_{46} + s_3s_{46}s_{64} + b_{63}(s_{64} + s_3s_{46} + s_{25}s_{46} + s_{46}s_{64} + 1)
\end{aligned}$$

7.2 Experimental approach

For each simulation, random NFSR and LFSR states were produced. Output from the Grain-like structure was then produced. The attack from Section 4 was then applied to fully recover the LFSR initial state and partially recover the NFSR initial state. The remaining NFSR bits were then exhaustively searched. Each initial state candidate was used to produce output, which was checked against the correct output sequence. A candidate that produced the correct output was considered the correct initial state. The computed initial state was then checked against the correct initial state.

The code used for the simulations was written using the SageMath software package [28] and all calculations were performed using QUT's High Performance Computing facility. We used a single node from the cluster with an Intel Xeon core capable of 271 TeraFlops.

7.3 Results on toy Grain-like structure

In precomputation, the initial system of equations was built, the linear dependency was found and the reduced system of equations was built. For $n = 20$, approximately 2^{11} bits of output were used in the precomputation phase of the attack. For $n = 30$, approximately 2^{13} bits of output were used, approximately 2^{14} bits of output were used for $n = 40$ and approximately 2^{15} bits of output were used for $n = 50$. For the modified version of Grain-V0, approximately 2^{17} bits of output were used. The majority of the computational complexity required for the precomputation comes from applying the linear dependency to produce the reduced system of equations. On average, precomputation was completed in 3 seconds, 37 seconds, 8 minutes and 1 hour, respectively for the toy versions of Grain. The precomputation phase of the attack for the modified version of Grain-V0 took 24 hours on average.

A total of 100 simulations were performed for each structure. In every simulation the full LFSR initial state was recovered. Each simulation for the toy version required on average 10 seconds to recover the LFSR initial state, regardless of the register size. For the modified version of Grain-V0, 30 seconds were required on average to recover the LFSR state in the online phase.

For each simulation, partially recovering the NFSR took less than a second. Table 4 provides a tally (across the 100 simulations) of how many times a certain number of state bits were recovered from the NFSR. For each simulation, the full available keystream was used. That is, the NFSR state was partially recovered using 2^{11}, 2^{13}, 2^{14}, and 2^{15} bits of keystream for registers of length 20, 30, 40 and 50 respectively. Note that the results for NFSR recovery of the modified version of Grain-V0 can be found in Section 7.4.

We see from Table 4 that the number of NFSR initial state bits that were recovered varied. On average, 18, 25, 34 and 45 bits were recovered for the respective NFSR initial states of size 20, 30, 40 and 50. The remaining 2, 3, 5 and 8 bits were recovered by exhaustive search and used to produce output. The output was then compared against the correct output. This process took a maximum of a few seconds in each case.

Table 4. Distribution table for NFSR bits recovered over 100 simulations for Grain-like structures of length $n = 20, n = 30, n = 40$ and $n = 50$.

$n = 20$										
No. bits recovered	**0**	...	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Frequency	0	...	0	0	0	0	34	59	7	0

$n = 30$										
No. bits recovered	**0**	...	**22**	**23**	**24**	**25**	**26**	**27**	...	**30**
Frequency	0	...	0	3	9	72	16	0	...	0

$n = 40$										
No. bits recovered	**0**	...	**32**	**33**	**34**	**35**	**36**	**37**	...	**40**
Frequency	0	...	0	21	50	18	11	0	...	0

$n = 50$										
No. bits recovered	**0**	...	**43**	**44**	**45**	**46**	**47**	**48**	...	**50**
Frequency	0	...	0	10	53	33	4	0	...	0

7.4 NFSR recovery on full modified Grain family

Due to the low computational complexity of partially recovering the NFSR, we have performed simulations for the full modified versions of the Grain family. A total of 1000 simulations were performed for each variant. For each simulation, 2^{17} bits of output were produced and used to partially recover the NFSR initial state. Each simulation required less than a second to partially recover the NFSR state. Figure 3 and Figure 4 provide tallies (across the 1000 simulations) of how many times a certain number of state bits were recovered for the NFSR of each variant,

respectively. Grain-V0 and Grain-V1 have been graphed as a single data set due to the extreme similarity between the results for each variant. Similarly for Grain-128 and Grain-128a (without authentication), and Grain-128a (with authentication) and Grain-128AEAD. Table 5 provides an estimate for the exhaustive search requirements for each modified Grain variant, based on Figures 3 and 4.

Fig. 3. Histogram for NFSR bits recovered over 1000 simulations for each modified Grain variant, using 2^{17} bits of output.

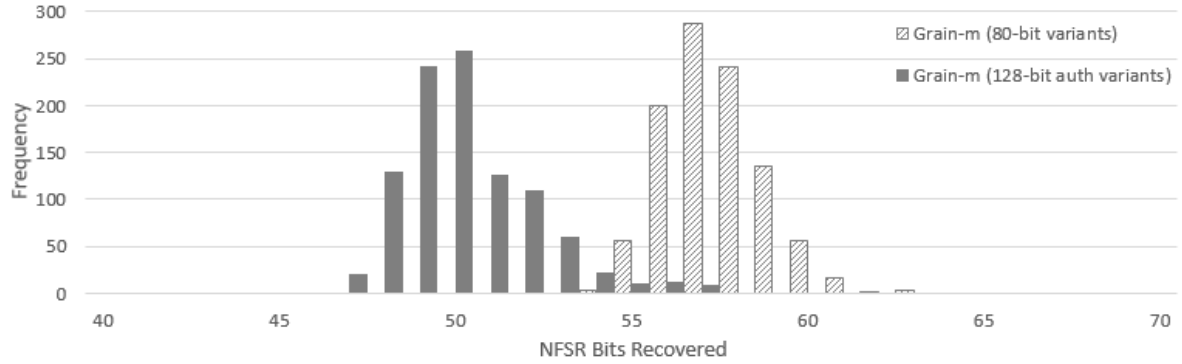

Fig. 4. Histogram for NFSR bits recovered over 1000 simulations for each modified Grain variant, using 2^{17} bits of output.

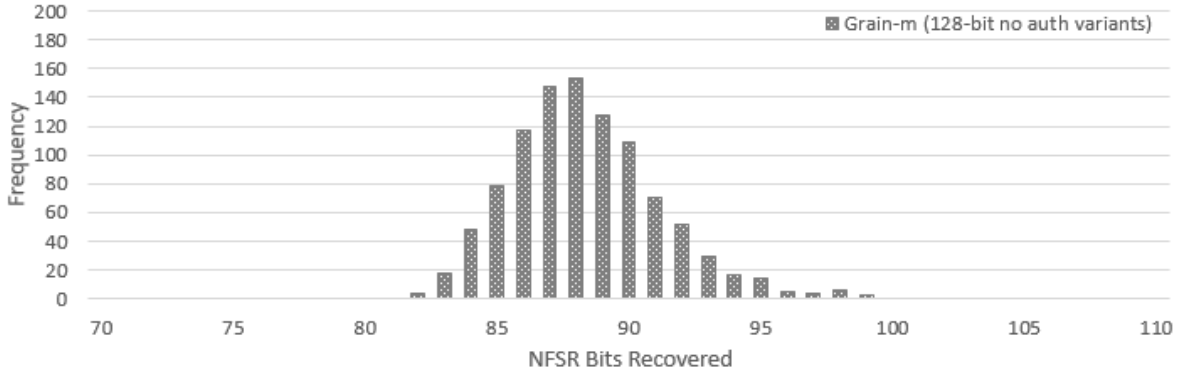

Table 5. Average exhaustive search requirement for the NFSR of each modified Grain variant using 2^{17} bits of output.

Variant	Grain-V0-m	Grain-V1-m	Grain-128-m	Grain-128a-m (no auth.)	Grain-128a-m (auth.)	Grain-128AEAD-m
Average NFSR exhaustive search complexity	2^{23}	2^{23}	2^{41}	2^{41}	2^{78}	2^{78}

It is worth noting that for the simulations presented in this paper, a limited number of output bits were used for the partial recovery of the NFSR. In practice, the NFSR only needs to be recovered once and so, an attacker may use all the available keystream for NFSR recovery. Using more keystream will increase the number of state bits recovered by an attacker, in general, but will take longer. For instance, Table 6 shows the average number of NFSR state bits recovered in Grain-128a-m, across 10 trials, when using 2^9, 2^{10}, 2^{13}, 2^{17}, 2^{19} and 2^{20} bits of keystream. By increasing the amount of keystream used for NFSR recovery from 2^9 bits to 2^{20} bits, 26 more NFSR state bits were recovered on average. This decreases the exhaustive search requirement for the remaining NFSR bits from 2^{62} to 2^{36}. Similar decreases in the exhaustive search complexity would also apply to the other variants in Table 5.

Table 6. Number of NFSR bits recovered for increasing amounts of keystream in Grain-128a-m (no authentication).

Number of output bits	2^9	2^{10}	2^{13}	2^{17}	2^{19}	2^{20}
Average number of NFSR bits recovered	66	66	74	87	91	92
Average exhaustive search complexity	2^{62}	2^{62}	2^{54}	2^{41}	2^{37}	2^{36}

8 Discussion/observations

Berbain et al. [2] proposed two attacks, one of which was successfully applied to a modified version of Grain-128, with an output function of the form $f(B, S) = L_B + L_S + f_S$. This attack required a system of equations of degree 6 in 256 unknowns to be solved, requiring approximately 2^{39} bits of keystream and with a computational complexity of 2^{105}.

In this paper, we present a new divide and conquer algebraic attack on Grain-like structures. Unlike the attack of Berbain et al. [2], our method can be applied to Grain-like structures with filter functions containing taps from the NFSR that are used nonlinearly. We show that Grain-like structures with an output function of the form

$$f(B, S) = L_S + f_S + f_{BS}$$

are always susceptible to our attack. It is therefore not sufficient to include NFSR bits nonlinearly as components of f_{BS} alone. Furthermore, we showed that output functions satisfying any of the forms:

- $f(B, S) = L_B + L_S + f_S + f_{BS}$
- $f(B, S) = L_S + f_S + f_B + f_{BS}$
- $f(B, S) = L_B + L_S + f_S + f_B + f_{BS}$

are all susceptible to a divide and conquer attack, if the filter function is chosen poorly. We showed in Section 6 that this is the case for a modified version of each Grain variant.

For example, Grain-128-m with output function of the form $f(B, S) = L_S + f_S + f_{BS}$ is susceptible to attack. First, the LFSR initial state is recovered via a fast algebraic attack requiring approximately 2^{28} bits of keystream and complexity of $2^{51.51}$. Then the NFSR initial state is partially recovered using the keystream produced for the LFSR recovery. Partial recovery of the NFSR takes about a second. From Table 6 we see that, if 2^{20} bits of output are used an average of 92 bits from the NFSR are recovered. The remainder of the state can be recovered in 2^{36} operations. Thus the entire initial state can be recovered with a complexity of about $2^{51.51}$, by observing approximately 2^{28} bits of keystream.

The results of Berbain et al. [2] provide full initial state recovery, whereas our approach provides full recovery of the LFSR and partial recovery of the NFSR state, with the remaining state bits of the NFSR being recovered by exhaustive search. We see, however, that in all cases without authentication, the exhaustive search of the NFSR does not significantly add to the complexity of the attack. Furthermore, our attack approach applies to a much larger set of filter functions. We identified an additional four cases of such functions that are vulnerable to algebraic attacks providing efficient partial state recovery. Moreover, these attacks are, in general, much more efficient than the attack proposed by Berbain et al. [2], and require less keystream.

This analysis highlights that it is not only the choice of a filter function that includes the NFSR

contents nonlinearly and linearly in the output function that currently provides resistance against algebraic attacks in the Grain variants. Rather, it is both the use of the NFSR bits linearly and nonlinearly, together with the careful choice of input stages that provides the resistance. The authors of Grain-128a state "Grain-128a does use an NFSR, which introduces much more nonlinearity together with the nonlinear filter function. Solving equations for the initial 256 bit state is not possible due to the nonlinear update of the NFSR and the NFSR state bits used nonlinearly in the nonlinear filter function [17]." We have shown that although Grain itself is currently resistant to algebraic attacks, the statement made by the authors is over simplified, and not always accurate. We have provided examples where functions satisfy this criterion, but the keystream generators are still susceptible to algebraic attacks.

9 Conclusion

This paper investigated the security of a certain type of contemporary keystream generator design against algebraic attacks. These keystream generators use a nonlinear feedback shift register, a linear feedback shift register and a nonlinear output function taking inputs from both registers. We refer to these designs as "Grain-like" structures, as the well known Grain family of stream ciphers has this structure.

Motivated by the work of Berbain et al. [2] we looked for approaches that eliminate nonlinear contributions from the NFSR to the filter function. Courtois' method in [5] shows how to reduce the degree of a function by multiplying through by a "annihilator". We took a similar approach, but used it to annihilate the nonlinear monomials in the filter function that take inputs from the NFSR. This allowed us to build a system of algebraic equations taking inputs from LFSR bits only, permitting a divide and conquer approach, first targeting the LFSR in an algebraic attack. Following this, NFSR recovery is possible. Note that our attack is applicable to a much larger set of filter functions than the attack presented by Berbain et al. [2], since we are not constrained to filter functions in which NFSR bits are only present linearly.

To illustrate the effectiveness of the attack, we applied our attack method to modified versions of the Grain family of stream ciphers. We performed simulations of our attack on Grain-like structures with internal state sizes of 40, 60, 80 and 100 bits, as well as on a modified version of Grain-V0. In our experiments, the full LFSR initial state was always correctly recovered. The NFSR initial state was partially recovered, dramatically reducing the exhaustive search requirement to obtain the remaining NFSR bits. Having recovered the initial state, it may then be possible to recover the secret key.

These results are relevant to designers of keystream generators as they clearly demonstrate that even when the output function incorporates bits from the LFSR and the NFSR both linearly and nonlinearly, the keystream generator may still be susceptible to attack, resulting in state recovery faster than brute force. We emphasise that our attack method is not applicable to the original Grain family as the filter functions included in those designs do not meet the conditions that permit this attack. However, this paper clearly demonstrates that some output functions previously thought to be secure against algebraic attacks are, in fact, insecure. Designers should carefully assess their chosen functions with this in mind.

References

1. Frederik Armknecht and Vasily Mikhalev. On lightweight stream ciphers with shorter internal states. In *International Workshop on Fast Software Encryption*, pages 451–470. Springer, 2015.
2. Côme Berbain, Henri Gilbert, and Antoine Joux. Algebraic and correlation attacks against linearly filtered non linear feedback shift registers. In *International Workshop on Selected Areas in Cryptography*, pages 184–198. Springer, 2008.
3. Côme Berbain, Henri Gilbert, and Alexander Maximov. Cryptanalysis of Grain. In *International Workshop on Fast Software Encryption*, pages 15–29. Springer, 2006.
4. Nicolas T Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt. In *International Conference on Information Security and Cryptology*, pages 182–199. Springer, 2002.
5. Nicolas T Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Annual International Cryptology Conference*, pages 176–194. Springer, 2003.
6. Nicolas T Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 345–359. Springer, 2003.
7. Christophe De Cannière, Özgül Küçük, and Bart Preneel. Analysis of Grain's initialization algorithm. In *International Conference on Cryptology in Africa*, pages 276–289. Springer, 2008.
8. Itai Dinur and Adi Shamir. Breaking Grain-128 with dynamic cube attacks. In *International Workshop on Fast Software Encryption*, pages 167–187. Springer, 2011.
9. Håkan Englund and Thomas Johansson. A new simple technique to attack filter generators and related ciphers. In *International Workshop on Selected Areas in Cryptography*, pages 39–53. Springer, 2004.
10. Jean-Charles Faugere and Gwénolé Ars. An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases. Report, INRIA, 2003.
11. Réjane Forré. A fast correlation attack on nonlinearly feedforward filtered shift-register sequences. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 586–595. Springer, 1989.
12. Berndt M Gammel and Rainer Göttfert. Linear filtering of nonlinear shift-register sequences. In *International Workshop on Coding and Cryptography*, pages 354–370. Springer, 2005.
13. J Dj Golić, Mahmoud Salmasizadeh, Leonie Simpson, and Ed Dawson. Fast correlation attacks on nonlinear filter generators. *Information Processing Letters*, 64(1):37–42, 1997.
14. A Hell, T Johansson, A Maximov, and W Meier. The Grain family of stream ciphers. In M Robshaw and O Billet, editors, *New Stream Cipher Deisgns*, volume 4986, pages 179–190. LNCS, 2008.
15. Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. In *2006 IEEE International Symposium on Information Theory*, pages 1614–1618. IEEE, 2006.
16. Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing*, 2(1):86–93, 2007.
17. Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *Int. J. Wirel. Mob. Comput.*, 5:48–59, 2011.
18. Martin Hell, Thomas Johansson, Willi Meier, Jonathan Sönnerup, and Hirotaka Yoshida. Grain-128AEAD - a lightweight AEAD stream cipher. *NIST Lightweight Cryptography Competition*, 2019.
19. Jonathan Katz, Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of Applied Cryptography*. CRC press, 1996.
20. Shahram Khazaei, Mehdi Hassanzadeh, and Mohammad Kiaei. Distinguishing attack on Grain. *ECRYPT Stream Cipher Project Report*, 71:2005, 2005.
21. Özgül Küçük. Slide resynchronization attack on the initialization of Grain 1.0. *eSTREAM, ECRYPT Stream Cipher Project, Report*, 44:2006, 2006.
22. Yuseop Lee, Kitae Jeong, Jaechul Sung, and Seokhie Hong. Related-key chosen IV attacks on Grain-v1 and Grain-128. In *Australasian Conference on Information Security and Privacy*, pages 321–335. Springer, 2008.
23. James Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.
24. Willi Meier and Othmar Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989.
25. Vasily Mikhalev, Frederik Armknecht, and Christian Müller. On ciphers that continuously access the non-volatile key. *IACR Transactions on Symmetric Cryptology*, pages 52–79, 2016.
26. William Millan. *Analysis and Design of Boolean Functions for Cryptographic Applications*. PhD Thesis, Queensland University of Technology, 1997.
27. Thomas Siegenthaler. Cryptanalysts representation of nonlinearly filtered ml-sequences. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 103–110. Springer, 1985.
28. William Stein and David Joyner. Sage: System for algebra and geometry experimentation. *ACM Bulletin*, 39(2):61–64, 2005.
29. Yosuke Todo, Takanori Isobe, Willi Meier, Kazumaro Aoki, and Bin Zhang. Fast correlation attack revisited. In *Annual International Cryptology Conference*, pages 129–159. Springer, 2018.
30. Haina Zhang and Xiaoyun Wang. Cryptanalysis of stream cipher Grain family. *IACR Cryptology ePrint Archive*, 2009:109, 2009.

A Algorithms

A.1 Algorithm for NLFG algebraic attack

Precomputation phase:

Step 1 Use $f(S_0) = y_0$ to relate initial state bits $(s_0, s_1, \ldots, s_{n-1})$ to observed output bit y_0.
Step 2 Multiply f by a function h (if applicable) to reduce overall degree to d.
Step 3 Clock forward using $f(S_t) = y_t$ to build a system of equations of constant algebraic degree, applying the linear update as required.

Online phase:

Step 4 Substitute observed output bits $\{y_t\}_{t=0}^{\infty}$ into the system of equations.
Step 5 Solve the system of equations by linearisation, to recover $S_0 = s_0, s_1, \ldots, s_{n-1}$.

A.2 Algorithm for Fast algebraic attack

The precomputation phase is similar to a regular algebraic attack, with Step 3 replaced by three steps (3a, 3b and 3c) as follows.

Step 3a Identify the combination of equations that will eliminate monomials of degree e to d in the initial state bits.
Step 3b Use this linear dependency to build a new general equation.
Step 3c Use this general equation to build a system of equations of degree e in the initial state bits.

A.3 Algorithm for LF-NFSR algebraic attack

Precomputation phase:

Step 1 A system of equations is developed using the linear filter function to represent every state bit as a linear combination of a subset of the initial state bits and some output bits. We denote this system of equation by system $\mathcal{L}$.
Step 2 A second system of equations is developed using the nonlinear update function g to represent update bits as a nonlinear combination of a subset of initial state bits. We denote this system by system $\mathcal{G}$. Substitutions are made for state bits in system $\mathcal{G}$ using system $\mathcal{L}$ where applicable to reduce the number of unknown state variables while keeping the degree of system $\mathcal{G}$ constant.
Step 3 The two systems are combined by aligning the equations from each system that represent the same state bit. The resulting system contains only initial state bits and observed output bits. We denote this system as system $\mathcal{L} + \mathcal{G}$.

Online phase:

Step 4 Substitute observed output bits $\{y_t\}_{t=0}^{\infty}$ into the system of equations
Step 5 Solve the system of equations by linearisation.

B Modified Version of Grain

Grain-V1$-m$

$$z = h(B,S) = s_1 + s_2 + s_4 + s_{10} + s_{31} + s_{43} + s_{56} + s_{25} + b_{63} + \\ + s_3 s_{64} + s_{46} s_{64} + s_{46} s_{64} + s_3 s_{25} s_{46} + s_3 s_{46} s_{64} + s_3 s_{46} b_{63} + s_{25} s_{46} b_{63} + s_{46} s_{64} b_{63}$$

As with Grain-V0$-m$, f of Grain-V1$-m$ satisfies Case 2 and so b_{63} was left in the function.

Grain-128−*m*

$$\begin{aligned} z = h(B,S) =& s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93} + \\ & s_8 b_{12} + s_{13} s_{20} + b_{95} s_{42} + s_{60} s_{79} + b_{12} b_{95} s_{95} \end{aligned}$$

Grain-128a−*m*/Grain-128AEAD−*m*

$$\begin{aligned} z = h(B,S) =& s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93} + \\ & s_8 b_{12} + s_{13} s_{20} + b_{95} s_{42} + s_{60} s_{79} + b_{12} b_{95} s_{94} \end{aligned}$$

Note that the structure of the filter function used Grain-128 is identical to the structure of the filter functions in Grain-128a, except that s_{95} in the final term for Grain-128 was changed to s_{94} in Grain-128a. This change is reflected in the modified versions shown here.

C Recovering the LFSR initial state of Grain

Grain-V1−*m*

At time $t = 0$ an output bit in Grain-V1 is produced as follows:

$$\begin{aligned} z_0 =& s_1 + s_2 + s_3 + s_4 + s_{10} + s_{31} + s_{43} + s_{56} + s_{25} + \\ & s_3 s_{64} + s_{46} s_{64} + s_3 s_{25} s_{46} + s_3 s_{46} s_{64} + b_{63}(1 + s_{64} + s_3 s_{46} + s_{25} s_{46} + s_{46} s_{64}) \end{aligned}$$

Multiplying this equation by $(s_{64} + s_3 s_{46} + s_{25} s_{46} + s_{46} s_{64})$ gives

$$\begin{aligned} (s_{64} + s_3 s_{46} + s_{25} s_{46} + s_{46} s_{64}) z_0 =& (s_1 + s_2 + s_4 + s_{10} + s_{31} + s_{43} + s_{56} + s_{25} + \\ & s_3 s_{64} + s_{46} s_{64} + s_3 s_{25} s_{46} + s_3 s_{46} s_{64})(s_{64} + s_3 s_{46} + s_{25} s_{46} + s_{46} s_{64}), \end{aligned}$$

where the right hand side of the equation contains only LFSR initial state bits. When the right hand side is expanded, the highest degree monomial is of order 3. Thus, by observing at least $\binom{80}{3}$ keystream bits, fast algebraic techniques may be applied in the precomputation phase of the attack to reduce the overall degree of the system to the degree of the left hand side (which is of degree 2 in the unknown LFSR initial state bits) [5].

Grain-128−*m*

At time $t = 0$ an output bit in Grain-128 is produced as follows:

$$z_0 = s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93} + s_8 b_{12} + s_{13} s_{20} + b_{95} s_{42} + s_{60} s_{79} + b_{12} b_{95} s_{95}$$

Multiplying this equation by $(s_8 + 1)(s_{42} + 1)(s_{95} + 1)$ gives

$$\begin{aligned} (s_8 + 1)(s_{42} + 1)(s_{95} + 1) z_0 =& (s_8 + 1)(s_{42} + 1)(s_{95} + 1)(s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + \\ & s_{93} + s_{13} s_{20} + s_{60} s_{79}), \end{aligned}$$

where the right hand side of the equation contains only LFSR initial state bits. When the right hand side is expanded, the highest degree monomial is of order 5. Thus, by observing at least $\binom{80}{5}$ keystream bits, fast algebraic techniques may be applied in the precomputation phase of the attack to reduce the overall degree of the system to the degree of the left hand side (which is of degree 3 in the unknown LFSR initial state bits) [5].

Grain-128a−m (without authentication)
At time $t = 0$ an output bit in Grain-128a is produced as follows:

$$z_0 = s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93} + s_8 b_{12} + s_{13} s_{20} + b_{95} s_{42} + s_{60} s_{79} + b_{12} b_{95} s_{94}$$

Multiplying this equation by $(s_8 + 1)(s_{42} + 1)(s_{94} + 1)$ gives

$$\begin{aligned}(s_8 + 1)(s_{42} + 1)(s_{94} + 1) z_0 = &(s_8 + 1)(s_{42} + 1)(s_{94} + 1)(s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + \\ &s_{93} + s_{13} s_{20} + s_{60} s_{79}),\end{aligned}$$

where the right hand side of the equation contains only LFSR initial state bits. When the right hand side is expanded, the highest degree monomial is of order 5. Thus, by observing at least $\binom{80}{5}$ keystream bits, fast algebraic techniques may be applied in the precomputation phase of the attack to reduce the overall degree of the system to the degree of the left hand side (which is of degree 3 in the unknown LFSR initial state bits) [5].

D Recovering the NFSR Initial State of Grain

Grain-V1−m
At time $t = 0$ an output bit in Grain-V1 is produced as follows:

$$\begin{aligned}z_0 = &s_1 + s_2 + s_4 + s_{10} + s_{31} + s_{43} + s_{56} + s_{25} + \\ &s_3 s_{46} + s_{25} s_{46} + s_3 s_{25} s_{46} + s_3 s_{46} s_{64} + b_{63}(1 + s_{64} + s_3 s_{46} + s_{25} s_{46} + s_{46} s_{64})\end{aligned}$$

Similarly to Grain-V0-m, this output function is already linear in b_{63} and the state can be partially recovered in a similar way.

Grain-128−m
At time $t = 0$ an output bit in Grain-128 is produced as follows:

$$z_0 = s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93} + s_8 b_{12} + s_{13} s_{20} + b_{95} s_{42} + s_{60} s_{79} + b_{12} b_{95} s_{95}$$

There is one monomial $(b_{12} b_{95} s_{95})$ that is of degree 2 in NFSR initial state bits. At each time step we have:

$$z_t = \alpha b_{12} + \beta b_{95} + \gamma b_{12} b_{95} + \zeta$$

As described in Section 4.2, these equations can be used to gain information about individual NFSR state bits when not all of α, β and γ are 0. This information can in turn be used to partially recover the NFSR initial state.

Grain-128a−m (without authentication)
At time $t = 0$ an output bit in Grain-128a is produced as follows:

$$z_0 = s_2 + s_{15} + s_{36} + s_{45} + s_{64} + s_{73} + s_{89} + s_{93} + s_8 b_{12} + s_{13} s_{20} + b_{95} s_{42} + s_{60} s_{79} + b_{12} b_{95} s_{94}$$

There is one monomial $(b_{12} b_{95} s_{94})$ that is of degree 2 in NFSR initial state bits. The possible output equations will be the same for Grain-128a−m (without authentication) as it is for Grain-128-m. The state can then be partially recovered in the same way as Grain-128-m.

Grain-128a−m (with authentication)/Grain-128AEAD-m
The possible output equations will be the same for Grain-128a−m (with authentication) as it is for Grain-128-m. In the case of Grain-128a−m (with authentication)/Grain-128AEAD-m, we may only utilise even index output bits to recover NFSR initial state bits. This will result in less of the state being recovered overall.