Near Collision Attack Against Grain v1

Subhadeep Banik[1], Daniel Collins[2] and Willi Meier[3]

[1]Universita della Svizzera Italiana, Lugano, Switzerland
subhadeep.banik@usi.ch
[2]Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
daniel.collins@epfl.ch
[3]FHNW, Windisch, Switzerland
willimeier48@gmail.com

Abstract. A near collision attack against the Grain v1 stream cipher was proposed by Zhang et al. in Eurocrypt 18. The attack uses the fact that two internal states of the stream cipher with very low hamming distance between them, produce similar keystream sequences which can be identified by simple statistical tests. Such internal states once found in the stream cipher simplify the task of cryptanalysis for the attacker. However this attack has recently come under heavy criticism from Derbez et al. at ToSC 2020:4, who claim that some of the assumptions made in the above paper were not correct. As a result they concluded that the attack presented by Zhang et al. when implemented would take time more than required for a brute force search. In this paper, we take another look at the near collision attack against the Grain v1 stream cipher. We avoid the techniques of the above Eurocrypt paper that have come under criticism, and independently show that a near collision attack can still be applied to Grain v1.

Keywords: Near Collision Attack, Grain v1, LFSR, NFSR, Stream Cipher.

1 Introduction

The Grain family of stream ciphers, proposed by Martin Hell, Thomas Johansson and Willi Meier in 2005, is designed for constrained devices. Grain v1 [HJM05] is included in the final hardware portfolio of the eStream project [RB08]. To meet increased security requirements, the designers proposed a 128-bit version called Grain-128 in ISIT 2006 [HJMM06a] and later the cipher Grain-128a [AHJM11a] to accommodate authentication with encryption. An AEAD version of the cipher Grain-128 AEAD v2 [HJM$^+$19] made it to the 3rd and final round of the NIST Lightweight Competition [lwc]; the underlying stream cipher Grain-128a is standardized in ISO/IEC 29167-13 for use in RFID [iso].

A near collision attack on the Grain v1 stream cipher was first presented in [ZLFL14]. The paper had one flaw that the authors overlooked. The paper uses a result stated in [AM08] to estimate the complexity of NFSR state recovery of Grain v1, once the LFSR state has been recovered. However the result in [AM08]

was actually based on the Grain v0 stream cipher (the very first submission of the authors of Grain to the eStream competition) which has a much simplified algebraic structure, and has been superseded by Grain v1 ever since. The authors in [ZLFL14] most probably mistook the result of [AM08] as based on Grain v1. Actually, recovering the NFSR state of Grain v1 given the LFSR state and the keystream bits generated thereof is also a difficult algebraic problem which requires significant computational resources to solve.

In [ZXM18], Zhang et al. proposed a near collision attack on Grain v1. The authors claimed a time complexity around $2^{75.7}$ ticks for their attack, where one tick was defined as one iteration of the Grain v1 round function computation. This attack was disproved in [DFM20]. To understand the controversy behind the paper let us try to state some of the claims made by the paper.

- Let f be the function that maps any internal state x in Grain v1 to a certain length of keystream z produced by it. The authors in [ZXM18] describe a technique called "Self-refined method" that given a keystream segment z_s outputs a set $X \subset f^{-1}(z_s)$. The authors claimed that if x_s is the actual internal state then $\mathsf{Pr}[x_s \in X] > \frac{|X|}{|f^{-1}(z_s)|}$. The authors of [DFM20] showed that this is simply not possible for any random mapping f.
- Note that here the function f is such that $z = f(x)$ can be rewritten as $z = x_1 \oplus h(x_2)$ and thus, the refined self-contained method was applied on $h(x_2) = 0$. In particular this means that the search space is restricted without the knowledge of any bit of keystream.
- By inspecting the first 20 keystream bits, [ZXM18] claimed to have found a set $\mathbf{X}$ of 118-bit elements of the internal state such that the actual state $\mathbf{x_s} \in \mathbf{X}$ with probability $2^{-54.1}$ and $|\mathbf{X}| = 2^{6.67}$. [DFM20] showed that this probability is actually $2^{-51.7}$ and $|\mathbf{X}|$ has to be around $2^{118-20-51.7} = 2^{47.3}$ and not $2^{6.67}$ as claimed. As a result [DFM20] claimed that the overall complexity of the attack in [ZXM18] would be around $2^{37.24}$ times $2^{75.1}$ ticks which is well above the complexity of brute force search.

The other most prominent attack on the Grain family was the correlation attack reported in [TIM+18]. The attackers try to formulate probabilistic linear equations relating subsets of keystream and LFSR state bits of high enough bias η. Once the attacker has around N such equations with $N \approx \frac{1}{\eta^2}$, the authors use a maximum likelihood decoding algorithm like the Fast Walsh Hadamard transform (FHWT) to find the LFSR state efficiently. Thereafter the key and NFSR state can be found by solving polynomial equations on the keystream bits (see [TIM+18]). Other than this, there have been cube attacks and conditional differential approaches that have attacked round reduced variants of the Grain family. Conditional differential attacks have been reported against round reduced Grain v1 in [KMN10, Ban16]. Cube/conditional attacks have been reported against Grain-128/Grain-128a in [DS11, DGP+11, LM12].

#	Type of Attack	Time	#Table Access	Reference
1	Fast Correlation Attack	$2^{76.7}$ multiplications over $GF(2^{80})$/ 80-bit integer additions		[TIM$^+$18]
2	Near Collision Attack	$2^{74.6}$ encryptions	$2^{80.5}$	Sec 4

Table 1. Comparison of attack complexities

1.1 Contribution and Organization of the Paper

In this paper we outline a near collision attack on Grain v1, without adopting any of the controversial methodologies in [ZXM18]. We show that it is still possible to mount such an attack on Grain v1 which runs in time barely below that of exhaustive search. Thus it plugs a gap in literature that existed ever since [DFM20] was published. Our attack requires $2^{74.6}$ encryptions and $2^{80.5}$ insertions in a table. Unless we have the technology to do memory access very efficiently, the above attack when implemented may take physical time comparable to that of exhaustive search. Hence in the appendices we demonstrate that a simple tradeoff due to the low sampling resistance of Grain v1 allows us to mount the attack using more encryptions while reducing the number of memory accesses by the same multiplicative factor.

Table 1 compares our attack with [TIM$^+$18] which is the only other attack proposed against the full version of Grain v1. Note that the 2 attacks can not be directly compared since [TIM$^+$18] reports attacks assuming that multiplication by a constant over $GF(2^{80})$ and addition/subtraction over 80-bit integers require the same complexity. Although the cryptanalytic approach taken in this paper may not be more efficient than [TIM$^+$18], it serves to highlight an important design issue in the construction of stream ciphers with Grain-like structure.

The attack has mainly been possible due to the fact that the taps which feed the output function in Grain v1 are sparsely distributed over its 160-bit internal state (there are only 12 output taps). This allows us to filter a lot of unnecessary candidates for internal state (and save computational time) as shown in Lemma 2. Intuitively this suggests that even if stream ciphers have internal state twice the size of secret key, the sparseness of output taps may be a source of algebraic weakness. As a countermeasure designers could either opt for denser distribution of output taps, or choose an internal state slightly more than twice the length of secret key. It is however noteworthy that stream ciphers like Atom [BCI$^+$21] which technically have smaller state size, increase the state size artificially by including the key in the state update function,

The rest of the paper is organized in the following manner. We begin with a brief algebraic description of Grain v1 in Section 2. In Section 3 we present some preliminary results of the differential structure of the Grain v1 stream cipher that will help us construct the attack. Section 4 describes the attack in full. Section 5 concludes the paper. In the appendices, we describe a sampling rate based tradeoff that allows us to reduce the number of memory accesses

(Appendix C), and show that our attacks can be analogously extended to Grain-128 and Grain-128a (Appendix D). We note here that the attack on Grain-128a does not directly impact the NIST LWC finalist Grain-128 AEAD v2 due to its authentication mechanism.

2 Algebraic description of Grain v1

Grain v1 consists of an 80-bit LFSR and an 80-bit NFSR. Certain bits of both shift registers are taken as inputs to a combining Boolean function, whence the keystream is produced. The update function of the LFSR is given by the equation $l_{t+80} = f(L_t)$, where $L_t = [l_t, l_{t+1}, \ldots, l_{t+79}]$ is an 80-bit vector that denotes the LFSR state at the t^{th} clock interval and f is a linear function on the LFSR state bits obtained from a primitive polynomial in $GF(2)$ of degree n given by $f(L_t) = l_{t+62} + l_{t+51} + l_{t+38} + l_{t+23} + l_{t+13} + l_t$. The NFSR state is updated as $n_{t+80} = l_t + g(N_t)$. Here, $N_t = [n_t, n_{t+1}, \ldots, n_{t+79}]$ is an 80-bit vector that denotes the NFSR state at the t^{th} clock and g is a non-linear function of the NFSR state bits given by:

$$
\begin{aligned}
g(N_t) = {} & n_{t+62} + n_{t+60} + n_{t+52} + n_{t+45} + n_{t+37} + n_{t+33} + n_{t+28} + n_{t+21} + n_{t+14} \\
& + n_{t+9} + n_t + n_{t+63}n_{t+60} + n_{t+37}n_{t+33} + n_{t+15}n_{t+9} + n_{t+60}n_{t+52}n_{t+45} \\
& + n_{t+33}n_{t+28}n_{t+21} + n_{t+63}n_{t+60}n_{t+52}n_{t+45}n_{t+37} + n_{t+60}n_{t+52}n_{t+37}n_{t+33} \\
& + n_{t+63}n_{t+60}n_{t+21}n_{t+15} + n_{t+63}n_{t+45}n_{t+28}n_{t+9} + n_{t+33}n_{t+28}n_{t+21}n_{t+15}n_{t+9} \\
& + n_{t+52}n_{t+45}n_{t+37}n_{t+33}n_{t+28}n_{t+21}
\end{aligned}
$$

The output keystream is produced by combining the LFSR and NFSR bits as $z_t = h'(N_t, L_t) = \bigoplus_{a \in A} n_{t+a} + h(l_{t+3}, l_{t+25}, l_{t+46}, l_{t+64}, n_{t+63})$, where $A = \{1, 2, 4, 10, 31, 43, 56\}$ and $h(s_0, s_1, s_2, s_3, s_4) = s_1 + s_4 + s_0s_3 + s_2s_3 + s_3s_4 + s_0s_1s_2 + s_0s_2s_3 + s_0s_2s_4 + s_1s_2s_4 + s_2s_3s_4$.

Grain v1 uses an 80-bit key K, and a 64-bit initialization vector IV. The key is loaded in the NFSR and the IV is loaded in the 0^{th} to the 63^{rd} bits of the LFSR. The remaining 16 bits of the LFSR are loaded with the constant `0xFFFF`. Then for the first 160 clocks, the keystream produced at the output point of the function h' is XOR-ed to both the LFSR and NFSR update functions, i.e., during the first 160 clock intervals, the LFSR and the NFSR bits are updated as $l_{t+80} = f(L_t) + z_t$, $n_{t+80} = l_t + z_t + g(N_t)$. After the completion of the KSA, z_t is no longer XOR-ed to the LFSR and the NFSR but it is used as the Pseudo-Random keystream bit. Therefore during this phase, the LFSR and NFSR are updated as $l_{t+80} = f(L_t), n_{t+80} = l_t + g(N_t)$.

3 Preliminaries

Let us look at a few preliminary results which helps build the attack. The first lemma is adapted from [BBI19, Lemma 1].

Lemma 1. *[BBI19] Consider two time instances $t_1,\ t_2$ during the keystream phase (with $t_2 > t_1$ and both less than 2^{80}). Then, given the 80-bit difference vector $\delta = L_{t_1} \oplus L_{t_2}$, t_1 and t_2, it is possible to compute the LFSR states L_{t_1}, L_{t_2} in around $5 \cdot 80^3$ bit operations.*

Proof. Although the lemma was proven in [BBI19], for the completeness of the paper we give another proof. If M is the companion matrix over $GF(2)$ of the connection polynomial $p(x)$ of the LFSR, then we can write L_{t+1} as a matrix-vector product between M and L_t. Thus we have $L_{t+1} = M \cdot L_t$. We thus have $L_{t_2} = M^{t_2 - t_1} \cdot L_{t_1}$. And so we have,

$$\delta = \ L_{t_2} \oplus L_{t_1} = (M^{t_2 - t_1} \oplus I) \cdot L_{t_1}$$

The above is a system of linear equations with the 80 variables in the L_{t_1} vector as unknowns. Further it is known that the minimal polynomial of M is the connection polynomial $p(x)$ of the LFSR itself. Since $p(x)$ is primitive, we know that an isomorphism exists between $GF(2^{80}) \cong \mathbb{F}_2[M] = \{\mathbf{0}, I, M, M^2, \ldots, M^{2^{80}-2}\}$. Define $T = t_2 - t_1$. The matrix $M^T \oplus I$ corresponds to the finite field element $\alpha^T + 1$ in $GF(2^{80})$ (here α denotes any root of $p(x)$). Since $0 < T < 2^{80} - 1$, $\alpha^T + 1$ is a non-zero element in $GF(2^{80})$, and it must have a multiplicative inverse β. The inverse of $M^T \oplus I$ is therefore the image of β in $\mathbb{F}_2[M]$ of the given isomorphic map. Since we have proven that $M^T \oplus I$ is invertible, so the above system of equations can be solved efficiently by Gaussian elimination to compute L_{t_1} and hence L_{t_2}.

How to solve the system of equations: The second question is given $T,\ \delta$ how many operations does it take to find L_{t_1}. Computing M^T using a standard square and multiply approach requires around $log_2\ T$ iterations of the square and multiply routine. The value $log_2\ T$ is naturally bounded by 80 and, estimating conservatively that matrix multiplication requires around n^3 bit-operations, calculating M^T alone takes $2 \cdot 80 \cdot 80^3 \approx 2^{26.3}$ bit-operations in the worst case. However it is possible to do better, by using the isomorphism that exists between $GF(2^{80}) \cong \mathbb{F}_2[M]$. The idea is therefore to compute $\alpha^T = q(\alpha) \bmod p(\alpha)$, where q is a polynomial in $GF(2)$ of degree less than 80 and p is the primitive polynomial whose roots are used to construct the field extension. We then compute $M^T = q(M)$. If the computation of M^T needs to be done for many values of T, one can simply pre-compute $M^i,\ \forall\ i \in [1, 79]$. and then the task boils down to efficiently computing q given T.

Again we can take a square and multiply approach to compute $\alpha^T \bmod p(\alpha)$. The idea is to reduce α^j modulo the primitive polynomial $p(\alpha)$ after each square or multiply operation every time the degree of the result exceeds 80. As a result after the k-th iteration we are always left with a polynomial q_k with degree less than 80. Note that for any polynomial r over GF(2) we have $r(\alpha)^2 = r(\alpha^2)$. As a result squaring over GF(2) comes for free. Thereafter reduction modulo $p(\alpha)$ can be done as follows: since we limit the degree of the polynomials to 79 at each stage, $r(\alpha^2)$ has degree at most 158. We can precompute the polynomials $m_i(\alpha) = \alpha^i \bmod p(\alpha), \forall\ i \in [80, 158]$. Thereafter reduction can be done by xoring

$m_i(\alpha)$ to the resultant if $r(\alpha^2)$ has a term of degree i. This step is bounded by 80^2 bit-operations in the worst case. Multiplication between two polynomials using even a naive shift and add approach requires 80^2 bit-operations, after which the reduction modulo $p(\alpha)$ requires another 80^2 operations using the above approach. Hence one iteration of square and multiply will require at worst $3 \cdot 80^2$ operations. Since the number of iterations is bounded by $\log_2 T$ which is 80, the computation of $q(\alpha)$ requires $3 \cdot 80^3$ operations in the worst case. We then compute $M^T \oplus I = q(M) \oplus I$ using the precomputed M^i matrices. Since one matrix addition takes 80^2 bit-operations and $q(\cdot)$ has at most 80 terms, computing $q(M) \oplus I$ again requires 80^3 operations in the worst case. Thereafter Gaussian elimination of $M^T \oplus I$ using even a naive approach would require 80^3 bit-operations. Thus solving this requires (in the worst case) $(3 + 1 + 1) \cdot 80^3 \approx 2^{21.3}$ bit-operations.

Lemma 2. *Consider two internal states in Grain v1, $S_{t_1} = (N_{t_1}, L_{t_1})$ and $S_{t_2} = (N_{t_2}, L_{t_2})$ during the keystream phase such that $S_{t_1} \oplus S_{t_2} = 0^{80}||e_{79}$, i.e. $N_{t_1} = N_{t_2}$ and $L_{t_1} \oplus L_{t_2} = e_{79}$, ($e_i$ is the 80-bit unit hamming weight vector, with 1 at location i). Then consider the vectors Z_{t_1} and Z_{t_2} of the first 140 keystream bits generated by S_{t_1} and S_{t_2} respectively. Also consider the vectors Y_{t_1} and Y_{t_2} of the first 30 keystream bits produced by S_{t_1} and S_{t_2} respectively, in the backward direction, i.e. by running the inverse state update routine. To be more specific*

$$Z_{t_i} = [z_{t_i+0}, z_{t_i+1}, z_{t_i+2}, \ldots, z_{t_i+139}],\ Y_{t_i} = [z_{t_i-1}, z_{t_i-2}, z_{t_i-3}, \ldots, z_{t_i-30}].$$

*for $i = 1, 2$. Then in the 170 bit difference vector $\Delta = Z_{t_1}||Y_{t_1} \oplus Z_{t_2}||Y_{t_2}$, there are **100** bits that take the value 1 or 0 with probability 1, i.e. when the probability is computed over all possible initial states S_{t_1}.*

Proof. The above result is not difficult to verify, if we analyze the differential trail of the difference when introduced in the 79th LFSR location. In the forward direction, for $j \in [0, 129] - S$ where $S = \{15, 33, 44, 51, 54, 57, 62, 69, 72, 73, 75, 76, 80, 82, 83, 87, 90, 91\} \cup [93, 96] \cup [98, 100] \cup \{102, 103, 105, 108, 109\} \cup [111, 113] \cup [115, 121] \cup [123, 126] \cup \{128\}$, and $j \in \{-1\} \cup [-10, -6] \cup \{-13, -16, -22\}$

1. the differences (between the internal states S_{t_1+j} and S_{t_2+j}) sit on tap locations that are not used in the computation of the keystream bit, or
2. the differences (between the internal states S_{t_1+j} and S_{t_2+j}) sit on an *even* number of NFSR locations (with probability 1) that contribute linearly to the output keystream.

Hence for all such j, $z_{t_1+j} = z_{t_2+j}$. Whereas, for $j' \in \{103, -2, -3, -5, -15, -19\}$, the difference appears at an *odd* number of NFSR tap locations that contributes to the keystream equation linearly. For all such j', we have $z_{t_1+j'} \oplus z_{t_2+j'} = 1$, with probability 1. There are, in total, 100 time instances where these events take place, hence a total of 100 bits in the difference vector are guaranteed to be either 0 or 1, with probability 1. □

We present a result relating the differential structure of Grain v1: we will use this and related results to reduce the number of candidates for the internal state in Sec 4.

Lemma 3. *Consider again the conditions in the previous lemma. Define* $c_i = z_{t_1-i} \oplus z_{t_2-i}$. *We have the following identities with probability 1.*

$$c_{11} \oplus c_{12} = 1, \quad c_{11} \oplus c_{14} = 1, \quad c_{14} \oplus c_{18} \oplus c_{20} \oplus c_{21} = 1$$
$$c_{21} \oplus c_{23} \oplus c_{24} \oplus c_{25} \oplus c_{28} = 1$$

Proof. To prove this we have to run the difference trail backwards, i.e. clock the cipher backwards and investigate the propagation of the difference at 79th LFSR bit. We will prove the first identity of the lemma because the proof for the others are similar. At $j = -11, -12, -14$, the differences between the internal states S_{t_1+j} and S_{t_2+j} sit on the following tap locations:

1. At $j = -11$, at NFSR location 1, there is a probabilistic difference (i.e. which occurs with probability less than 1). At $j = -12$, the same difference propagates to NFSR location 2 and at $j = -14$, it propagates to NFSR location 4, all of which contribute to the keystream expression linearly.
2. Additionally at $j = -11$, there is a deterministic difference at NFSR location 10 (i.e. which occurs with probability 1),
3. At $j = -11, -12, -14$, no other tap locations that contribute to the keystream bit contain any difference.

Note that since the difference δ at NFSR location 1 at $j = -11$ is probabilistic and does not occur with probability 1, we have $z_{t_1-11} \oplus z_{t_2-11} = 1 \oplus \delta$ and so it is not guaranteed to be 1 or 0. However the same δ propagates to NFSR locations 2 and 4 at $j = -12, -14$, and so we have

$$z_{t_1-11} \oplus z_{t_2-11} \oplus z_{t_1-12} \oplus z_{t_2-12} = (1 \oplus \delta) \oplus \delta = 1$$

Since the probabilistic difference gets canceled out in the expression, the above identity holds with probability 1. Similar arguments can be made for the other identities. For example, for the third identity, we need to verify that all the probabilistic differences produced at $j = -14, -18, -20, -21$ get linearly canceled out in the respective keystream expressions. □

The following lemma establishes the number of keystream bits that need to be generated, to produce in the process two internal states S_{i_1}, S_{i_2} that differ at only the 79th LFSR location.

Lemma 4. *In the event that we generate* N *iterations of internal states of Grain v1 sequentially:* $S_i \in \{0,1\}^{160}$, *for* $i \in [0, N-1]$, *then the probability that there is at least one tuple* $i_1, i_2 \in [0, N-1]$ *and* $i_2 > i_1$, *such that,* $S_{i_1} \oplus S_{i_2} = 0^{80}||e_{79}$, *is approximately* $p_{coll} = \frac{N^2}{2^{161}}$.

Proof. The lemma is essentially the same as [BBI19, Lemma 4], in which it was proven by birthday bound considerations. We will try to give a more precise proof here. Consider the initial state of Grain v1 $N_0 = [n_0, n_1, \ldots, n_{79}]$, $L_0 = [l_0, l_1, \ldots, l_{79}]$. Initially all the n_i, l_i's are i.i.d according to $Ber(\frac{1}{2})$. Note that any subsequent n_{t+80} is given as $g(n_t, n_{t+1}, \ldots, n_{t+79}) + l_t$. Since all the l_t's are

linear functions of $l_0, l_1, \ldots, l_{79}$ which are independently distributed, in this proof we will assume that all n_i's (even for $i \geq 80$) are i.i.d as per $Ber(\frac{1}{2})$ which seems to be a reasonable assumption to make. If we run Grain v1 for N iterations we have the string $S_N = n_0, n_1, \ldots, n_{79+N}$ of length $N + 80$ over GF(2) obtained by concatenating all NFSR bits. Let us try to answer the simpler question: what is the probability that given i_1, i_2 we have that NFSR states at i_1, i_2 collide, i.e. the substrings $s_{i_1} = n_{i_1}, n_{i_1+1}, \ldots n_{i_1+79}$ and $s_{i_2} = n_{i_2}, n_{i_2+1}, \ldots n_{i_2+79}$ are equal (where the probability is computed over all possible values of S_N). We have two cases here:

Case 1: $i_2 - i_1 \geq 80$**:** This implies that there is no overlap between s_{i_1} and s_{i_2}. Thus $\mathsf{Pr}[s_{i_1} = s_{i_2}] = \prod_{j=0}^{79} \mathsf{Pr}[n_{i_1+j} = n_{i_2+j}] = 2^{-80}$.
Case 2: $i_2 - i_1 < 80$**:** This implies an overlap. We have to calculate the number of strings S_N such that given i_1, i_2 the substrings s_{i_1} and s_{i_2} are equal. Let $i_2 - i_1 = \Delta$, and $Q = \lfloor \frac{80 - i_2 + i_1}{\Delta} \rfloor$. Consider the substring $M = n_{i_1}, \ldots, n_{i_2+79}$ of length $80 + i_2 - i_1$. A necessary and sufficient condition for the substrings s_{i_1} and s_{i_2} to be equal is:

$$n_{i_1+j} = n_{i_1+\Delta+j} = \cdots = n_{i_1+Q\Delta+j}, \ \forall \ j \in [0, i_2 - i_1 - 1]$$

So, since there are only $i_2 - i_1$ bits that we can freely select in M (i.e. n_{i_1} to $n_{i_1+\Delta-1}$), we have $\#M = 2^{i_2-i_1}$. There are $N + 80 - 80 + i_2 - i_1 = N - i_2 + i_1$ bits remaining in S_N which can be selected freely so we have $\#S_N = 2^{i_2-i_1+N-i_2+i_1} = 2^N$. Hence we have $\mathsf{Pr}[s_{i_1} = s_{i_2}] = \frac{2^N}{2^{N+80}} = 2^{-80}$.

So we see that in both cases $\mathsf{Pr}[s_{i_1} = s_{i_2}] = 2^{-80}$. We now turn our attention to the LFSR state. Given i_1, i_2 the probability that $L_{i_1} \oplus L_{i_2} = e_{79}$ is given by Lemma 1. For this to hold, L_{i_1} has to be the unique non-zero solution x of the equation $(M^{i_2-i_1 \bmod 2^{80}-1} \oplus I) \cdot x = e_{79}$[1]. This means that $L_0 = M^{-i_1} \cdot x$. Note that if $L_0 = \mathbf{0}$ then all subsequent L_i will be $\mathbf{0}$ too and so we must exclude this case. Thus the probability that the LFSR states are equal at i_1 and i_2 is given by Bayes theorem as $\mathsf{Pr}[L_0 = M^{-i_1} \cdot x] \approx \frac{1}{2^{80}} \cdot 0 + (1 - \frac{1}{2^{80}})\frac{1}{2^{80}} \approx 2^{-80}$ where the probability is calculated over all values of L_0. Thus given (i_1, i_2) the probability p_{i_1,i_2} that $S_{i_1} \oplus S_{i_2} = 0^{80}||e_{79}$ is given by $2^{-80-80} \approx 2^{-160}$. Given N internal states the probability that we find one such tuple (i_1, i_2) is given as $p_{coll} = \binom{N}{2} \cdot 2^{-160} \approx \frac{N^2}{2^{161}}$ (assuming of course that distribution of each tuple is i.i.d). Note that for $N \approx 2^{80.5}$ we expect to find one such tuple. □

To experimentally confirm the above we generated Grain like ciphers with register lengths n from 8 to 15, with the LFSR update obtained from a random primitive polynomial and the NFSR update a random non-linear polynomial. We tried to find the average number of iterations after which we get one t_1, t_2

[1] We exclude the degenerate case when $i_2 \equiv i_1 \bmod 2^{80} - 1$, as then L_{i_1} must be equal to L_{i_2} and so their difference can not be e_{79}. But this occurs with extremely small probability for the range of values of N we are interested in and so we ignore this event here.

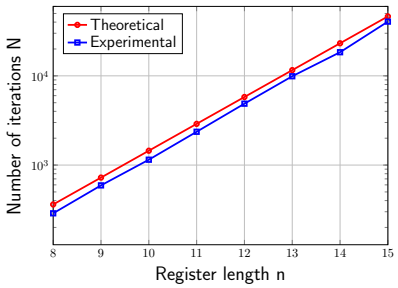

Fig. 1. Theoretical and Experimental estimates for N

with $S_{t_1} \oplus S_{t_2} = 0^n||e_{n-1}$, by generating random initial states updating the state till a collision of the required type is found. For each n we obtained the average over around (a) 1000 randomly generated update functions and then (b) for each function random update function over 10000 random initial states and computed the average #iterations taken to find the required collision. The results are presented, in Fig 1, which shows that theoretical and experimental values of the number of iterations N are quite similar.

4 State recovery attack

Having made some preliminary observations about the differential structure of Grain v1, we are now ready to mathematically describe the attack steps. Note that by the previous lemma, if we generate $N \approx 2^{80.5}$ keystream bits generated by any key/IV pair, giving rise to equal number of internal states S_{t_1}, then we are almost certain to encounter 2 states in the process such that $S_{t_{i_1}} \oplus S_{t_{i_2}} = 0^{80}||e_{79}$. When this happens, then the attacker can identify the states and the corresponding values of t_1, t_2 by looking at the difference keystream vector $\Delta = Z_{t_1}||Y_{t_1} \oplus Z_{t_2}||Y_{t_2}$ (which was defined in the previous section).

However, although it is true that two internal states with difference $0^{80}||e_{79}$ produce keystream bits whose differential is guaranteed to be 0 or 1 at 100 fixed locations (as per Lemma 2), the opposite is not true. In fact there exist, with probability around 2^{-100}, two completely random internal states of Grain v1 that produce a keystream differential of 0/1 at the same 100 locations enumerated in Lemma 2. Thus the attacker, when for some (t_1, t_2), observes a differential keystream with the required 0/1 pattern in the locations enumerated in Lemma 2, may still proceed to the next steps, assuming that they were generated by two Grain v1 states with difference only in the 79th LFSR location. But if his assumption about the state difference is wrong, then in the subsequent steps he would certainly reach a contradiction that would invalidate the assumption. The attacker would then require to repeat the experiment to obtain some other t_1, t_2

until he is successful in getting internal states with required difference. Thus any attack must compensate for these computational overheads.

Thus, at the very top level, the strategy of the attacker will be as follows:

A: Generate $N = 2^{80.5}$ keystream bits by a black-box accessing of the stream cipher with a fixed secret key/IV pair. This generates around N overlapping keystream segments of length 170-bits each.

B: For all $t \in [1, N]$, store in a hash table t, Z_t, Y_t as defined earlier.

C: From this table, try to find, if it exists, t_1, t_2 so that $\Delta = Z_{t_1}||Y_{t_1} \oplus Z_{t_2}||Y_{t_2}$. We refer to such an event as a keystream-collision.

D: If there exists one or multiple such t_1, t_2, then assuming that the state differential in between the states at time t_1, t_2 is in the 79th LFSR location, try to formulate an equation system on the variables of internal system for each keystream bit and try to solve the system.

E: If a contradiction is reached, try other values of t_1, t_2, if they exist. If the attacker does not encounter a contradiction, and is able to solve the equations, he would have computed the entire state.

4.1 Online Stage I: Collecting and storing keystream bits

In the online stage, the attacker needs to collect and store keystream bits and store them in a judicious manner. To facilitate detection of keystream-collision, we will use the same data structure for recording collisions as used in [BBI19]. We insert each tuple t, Z_t, Y_t in a table $\mathcal{T}$ as follows:

A: We describe table $\mathcal{T}$ that checks for keystream-collision arising due to internal state differences of form $0^{80}||e_{79}$. We insert t, Z_t, Y_t in table location $I = z_{t+g_0}||z_{t+g_1}||\cdots||z_{t+g_{99}}||z_{t-11} \oplus z_{t-12}||z_{t-11} \oplus z_{t-14}||z_{t-14} \oplus z_{t-18} \oplus z_{t-20} \oplus z_{t-21}||z_{t-21} \oplus z_{t-23} \oplus z_{t-24} \oplus z_{t-25} \oplus z_{t-28}$, where the g_i terms are the locations enumerated in Lemma 2 in which the differential keystream is guaranteed to be 0/1. Thus $(g_0, g_1, \ldots, g_{93}) = (0, 1, 2, \ldots, 14, 16 \ldots, 129, -1, -6, -7 \ldots, -10, -13, -16, -22)$. And we have $(g_{94}, g_{95}, \ldots, g_{99}) = (103, -2, -3, -5, -15, -19)$. Each entry in the table should be able to store multiple entries.

B: It is not difficult to see that a keystream-collision will occur if during an insertion into index I, the attacker checks the index $I^* = z_{t+g_0}||z_{t+g_1}||\cdots||z_{t+g_{93}}||1 \oplus z_{t+g_{94}}||1 \oplus z_{t+g_{95}}||\cdots||1 \oplus z_{t+g_{99}}||1 \oplus z_{t-11} \oplus z_{t-12}||1 \oplus z_{t-11} \oplus z_{t-14}||1 \oplus z_{t-14} \oplus z_{t-18} \oplus z_{t-20} \oplus z_{t-21}||1 \oplus z_{t-21} \oplus z_{t-23} \oplus z_{t-24} \oplus z_{t-25} \oplus z_{t-28}$, and finds one or multiple tuples already stored at I^*. This follows from Lemma 2, 3. For each such keystream-collision pair in (I, I^*), the attacker proceeds to the next steps of the attack.

It takes 80 bits to store t and 170 bits to store Z_t, Y_t and so the table contains around $250 \cdot N \approx 2^{88.5}$ bits of memory on average.

4.2 Online Stage II: Further filtering

For each keystream-collision pair obtained in the previous step, the attacker can first compute $T = t_2 - t_1$, and retrieve the LFSR state as follows. The attacker can find $L_{t_1},\ L_{t_2}$ by solving the equation $[M^T \oplus I] \cdot L_{t_1} = e_{79}$ as described in Lemma 1 in the previous section. The following lemma shows how he can reject a pair after computing the LFSR state.

Lemma 5. *If two internal states in Grain v1, $S_{t_1} = (N_{t_1}, L_{t_1})$ and $S_{t_2} = (N_{t_2}, L_{t_2})$ satisfy $S_{t_1} \oplus S_{t_2} = 0^{80}||e_{79}$, then we have the following identities. Define $d_i = z_{t_1+i} \oplus z_{t_2+i}$ for conciseness.*

$$d_{112} = l_{t_1+115} \cdot l_{t_1+158} \oplus l_{t_1+137} \cdot l_{t_1+158} \oplus l_{t_1+158} \cdot l_{t_1+176} \oplus l_{t_1+158} \oplus l_{t_1+176} \oplus 1$$
$$d_{125} = l_{t_1+128} \cdot l_{t_1+171} \oplus l_{t_1+150} \cdot l_{t_1+171} \oplus l_{t_1+171} \cdot l_{t_1+189} \oplus l_{t_1+189} \oplus 1$$

Proof. The Grain v1 output function $h(\mathbf{x}) = h(x_0, x_1, x_2, x_3, x_4)$ is such that only x_4 takes input from an NFSR location. The function has the additional differential properties that:

$$h(\mathbf{x}) \oplus h(\mathbf{x} \oplus [0,0,0,0,1]) = x_0 \cdot x_2 \oplus x_1 \cdot x_2 \oplus x_2 \cdot x_3 \oplus x_3 \oplus 1$$
$$h(\mathbf{x}) \oplus h(\mathbf{x} \oplus [1,1,0,0,0]) = x_0 \cdot x_2 \oplus x_1 \cdot x_2 \oplus x_2 \cdot x_3 \oplus x_2 \oplus x_3 \oplus 1$$

which are functions of only the LFSR bits. Also note that both the above functions are balanced. An analysis of the differential trails tells us the differences between the internal states S_{t_1+j} and S_{t_2+j} sit on the following tap locations:

1. at $j = 125$, at NFSR location 63 (corresponding x_4 in h), and at no other location that contribute to keystream bit.
2. at $j = 112$, at LFSR locations 3, 25 (corresponding x_0, x_1 in h), and at no other location that contribute to keystream.

□

By Lemma 5, there are 2 other bits in $Z_{t_1} \oplus Z_{t_2}$ that are directly related to L_{t_1}. Since during the keystream stage the LFSR evolves independently, all l_{t_1+i} can be computed with the knowledge of L_{t_1} alone. This provides us with an opportunity to further filter the keystream-collision pairs obtained from the stage. For example, by Lemma 5, if $z_{t_1+112} \oplus z_{t_2+112} \neq l_{t_1+115} \cdot l_{t_1+158} \oplus l_{t_1+137} \cdot l_{t_1+158} \oplus l_{t_1+158} \cdot l_{t_1+176} \oplus l_{t_1+158} \oplus l_{t_1+176} \oplus 1$, the attacker can discard the keystream pair.

4.3 Online Stage III: 3rd filtering

The knowledge of the LFSR state allows some more filtering steps as shown in the following lemmas.

Lemma 6. *Consider again the conditions in the previous lemma. At $j \in \{15, 33, 44, 51, 54, 57, 62, 69, 72, 73, 75, 76, 80, 82, 83, 87, 90, 91\} \cup [93, 96] \cup [98, 100] \cup \{102, 105, 108, 109, 111, 115\} \cup [117, 119] \cup \{121, 126, 128, 132, 135\} \cup \{-4\}$, we have*

$$d_j = p_{t_1+j} \cdot n_{t_1+j+63} \oplus q_{t_1+j},$$

where $p_{t_1+j},\ q_{t_1+j}$ are functions on only the LFSR bits.

Proof. Note that except for $[0,0,0,0,1]$, $[1,1,0,0,0]$ and $[1,1,0,0,1]$, for all the 28 other non-zero 5-bit vectors $\mathbf{v}$ the expression $h(\mathbf{x}) \oplus h(\mathbf{x} \oplus \mathbf{v})$ is of the form $x_4 \cdot f_1(x_0, x_1, x_2, x_3) \oplus f_2(x_0, x_1, x_2, x_3)$. It is an elementary exercise to verify that at the given values of j, the difference between the states at t_1, t_2 are such that the input to the function h differ by exactly one of the other 28 non-zero vectors mentioned above. Thus the above lemma follows. In Table 2 in Appendix B we tabulate the difference vectors, and the forms of the functions p_{t_1+j}, q_{t_1+j} for each j. □

Lemma 7. *Consider again the conditions in the previous lemma. We have*

$$z_{t_1-17} \oplus z_{t_2-17} \oplus z_{t_1-18} \oplus z_{t_2-18} = p' \cdot n_{t_1+46} \oplus q'$$

where $p' = l_{t_1+29}$ *and* $q' = l_{t_1+8} \cdot l_{t_1+29} \oplus l_{t_1+29} \cdot l_{t_1+47} \oplus l_{t_1+47}$.

Proof. The proof is similar to that of Lemma 3. At $j = -17, -18$, between N_{t_1+j}, L_{t_1+j} and N_{t_2+j}, L_{t_2+j} we have the following differences:

1. at $j = -17$, at NFSR location 1 there is probabilistic difference δ' and a probability 1 difference at LFSR location 3.
2. at $j = -18$, δ' propagates to NFSR location 2, and there is no other difference in any location that feeds z_{t_1-18}. Thus we have

$z_{t_1-17} \oplus z_{t_2-17} = \delta' \oplus h(l_{t_1+29}, \ldots) \oplus h(1 \oplus l_{t_1+29}, \ldots) = \delta' \oplus p' \cdot n_{t_1+46} \oplus q'$. Since $z_{t_1-18} \oplus z_{t_2-18} = \delta'$ the above lemma follows. □

Lemma 8. *Consider again the conditions in the previous lemma. We have*

$$\bigoplus_{j \in \{18,21,23,25,26,27\}} z_{t_1-j} \oplus z_{t_2-j} = l_{t_1+19} \cdot n_{t_1+36} \oplus l_{t_1+20} \cdot n_{t_1+37} \oplus q''$$

where q'' is a function only on the LFSR bits.

Proof. We need to verify that all the probabilistic differences produced in the NFSR at $j = -18, -21, -23, -25, -26, -27$ get canceled out among each other. However at $j = -26, -27$ deterministic differences are produced in the LFSR according to which the $l_{t_1+19} \cdot n_{t_1+36} \oplus l_{t_1+20} \cdot n_{t_1+37} \oplus q''$ terms are produced due to the contribution of $h(\mathbf{x}) \oplus h(\mathbf{x} \oplus \mathbf{v})$ terms during these time instances. □

Lemma 9. *Consider again the conditions in the previous lemma. We have*

$$\begin{aligned} z_{t_1+124} \oplus z_{t_2+124} &= (n_{t_1+187} \oplus n_{t_2+187}) \cdot f_l(l_{t_1+127}, l_{t_1+149}, l_{t_1+170}, l_{t_1+188}) \\ 1 \oplus z_{t_1+139} \oplus z_{t_2+139} &= (n_{t_1+202} \oplus n_{t_2+202}) \cdot f_l(l_{t_1+142}, l_{t_1+164}, l_{t_1+185}, l_{t_1+203}) \\ z_{t_1+142} \oplus z_{t_2+142} &= (n_{t_1+205} \oplus n_{t_2+205}) \cdot f_l(l_{t_1+145}, l_{t_1+167}, l_{t_1+188}, l_{t_1+206}) \end{aligned}$$

where $f_l(x_0, x_1, x_2, x_3) = x_0x_2 \oplus x_1x_2 \oplus x_2x_3 \oplus x_3 \oplus 1$ *is a balanced function only on the LFSR bits.*

Proof. Note that if u is any variable over $GF(2)$ then we have the following relation: $h(x_0, x_1, x_2, x_3, x_4) \oplus h(x_0, x_1, x_2, x_3, u \oplus x_4) = u \cdot f_l$. At $j = 124, 139, 142$, between N_{t_1+j}, L_{t_1+j} and N_{t_2+j}, L_{t_2+j} we have the following differences:

1. There exist no differences on any tap locations that contribute to the output keystream bit except NFSR location 63 which corresponds to the variable x_4 in $h(\cdot)$.
2. At NFSR location 63, the difference however is probabilistic and may or may not be 1, i.e. at $j = 124$, for example, the difference at NFSR location 63 $n_{t_1+187} \oplus n_{t_2+187}$ is not guaranteed to be either 0 or 1.
3. Additionally at $j = 139$, there exists a deterministic difference at NFSR location 56 which only linearly affects the keystream.

The above facts thus completely prove the lemma statements. It is straightforward to verify that f_l is a balanced function. □

We use the results in Lemma 6, 7, 8 and 9 to further filter any keystream pair that has survived the previous filters. This is how one can do this:

a) By Lemma 6, we know that if a keystream pair is produced by 2 internal states that differ by $0^{80}||e_{79}$ then there are 40 values of j for which $z_{t_1+j} \oplus z_{t_2+j} = p_{t_1+j} \cdot n_{t_1+j+63} \oplus q_{t_1+j}$, where p_{t_1+j}, q_{t_1+j} are functions on only of L_{t_1}. For example if for a given keystream pair that has survived the stage 2 filter, if for some value of j the attacker finds that $z_{t_1+j} \oplus z_{t_2+j} \oplus q_{t_1+j} = 1$ and $p_{t_1+j} = 0$ then the attacker can discard the keystream pair, since they could not have been generated by internal states with difference only in the 79th LFSR location.
b) Also due to Lemma 7, if that attacker finds $z_{t_1-17} \oplus z_{t_2-17} \oplus z_{t_1-18} \oplus z_{t_2-18} \oplus q' = 1$ and $p' = 0$, then too the keystream pair can be discarded.
c) Lemma 8 states that $\bigoplus_{j \in \{18,21,23,25,26,27\}} (z_{t_1-j} \oplus z_{t_2-j}) \oplus q'' = l_{t_1+19} \cdot n_{t_1+36} \oplus l_{t_1+20} \cdot n_{t_1+37}$. So if the attacker finds that $l_{t_1+19} = l_{t_1+20} = 0$ and $\bigoplus_{j \in \{18,21,23,25,26,27\}} (z_{t_1-j} \oplus z_{t_2-j}) \oplus q'' = 1$, then too he can discard the pair.
d) Similarly Lemma 9 relates the difference of 2 keystream bits with the product of an NFSR difference and the function f_l calculable on only the LFSR bits. For example if the attacker finds that $z_{t_1+124} \oplus z_{t_2+124} = 1$ and $f_l(l_{t_1+127}, l_{t_1+149}, l_{t_1+170}, l_{t_1+188}) = 0$, he can discard the pair.

So, let us calculate the probability that a given IV produces a keystream-collision pair that survives all the filter levels described above. Note that since a single IV can produce N tuples, the total number of pairs of tuples are $D = \frac{N(N-1)}{2} \approx 2^{160}$. Denote $\alpha_i = z_{t_1+g_i} \oplus z_{t_2+g_i}$ for all $i \in [0, 99]$ and $\alpha_{100} = z_{t_1-11} \oplus z_{t_1-12} \oplus z_{t_2-11} \oplus z_{t_2-12}$, $\alpha_{101} = z_{t_1-11} \oplus z_{t_1-14} \oplus z_{t_2-11} \oplus z_{t_2-14}$, $\alpha_{102} = \bigoplus_{j \in \{14,18,20,21\}} (z_{t_1-j} \oplus z_{t_2-j})$, $\alpha_{103} = \bigoplus_{j \in \{21,23,24,25,28\}} (z_{t_1-j} \oplus z_{t_2-j})$. And also define the following notations:

- $\beta_0 = z_{t_1+112} \oplus z_{t_2+112} \oplus l_{t_1+115} \cdot l_{t_1+158} \oplus l_{t_1+137} \cdot l_{t_1+158} \oplus l_{t_1+158} \cdot l_{t_1+176} \oplus l_{t_1+158} \oplus l_{t_1+176} \oplus 1$
- $\beta_1 = z_{t_1+125} \oplus z_{t_2+125} \oplus l_{t_1+128} \cdot l_{t_1+171} \oplus l_{t_1+150} \cdot l_{t_1+171} \oplus l_{t_1+171} \cdot l_{t_1+189} \oplus l_{t_1+189} \oplus 1$

- Define $h_0, h_1, \ldots, h_{39} = 15, 33, \ldots, 135, -4$ which are mentioned in Lemma 6.
- Define $\gamma_j = z_{t_1+h_j} \oplus z_{t_2+h_j} \oplus q_{t_1+h_j}$, and $\eta_j = p_{t_1+h_j}$ for all values of $j \in [0, 39]$
- Define $\gamma_{40} = z_{t_1-17} \oplus z_{t_2-17} \oplus z_{t_1-18} \oplus z_{t_2-18} \oplus q'$ and $\eta_{40} = p'$
- Define $\gamma_{41} = \bigoplus_{j\in\{18,21,23,25,26,27\}}(z_{t_1-j} \oplus z_{t_2-j}) \oplus q''$ and $\eta_{41} = l_{t_1+19}$, $\eta'_{41} = l_{t_1+20}$.
- Define $\mu_0 = z_{t_1+124} \oplus z_{t_2+124}$, $\mu_1 = z_{t_1+139} \oplus z_{t_2+139} \oplus 1$ and $\mu_2 = z_{t_1+142} \oplus z_{t_2+142}$. And $\bar{\mu}_0 = f_l(l_{t_1+127}, l_{t_1+149}, l_{t_1+170}, l_{t_1+188})$, $\bar{\mu}_1 = f_l(l_{t_1+142}, l_{t_1+164}, l_{t_1+185}, l_{t_1+203})$ and $\bar{\mu}_2 = f_l(l_{t_1+145}, l_{t_1+167}, l_{t_1+188}, l_{t_1+206})$

The probability that a pair is not rejected is given as

$$\rho = \prod_{i=0}^{93} \mathsf{Pr}(\alpha_i = 0) \cdot \prod_{i=94}^{103} \mathsf{Pr}(\alpha_i = 1) \cdot \prod_{i=0}^{1} \mathsf{Pr}(\beta_i = 0) \cdot \prod_{j=0}^{40} \Big(1 - \mathsf{Pr}(\gamma_j = 1 \wedge \eta_j = 0)\Big)$$
$$\cdot \Big(1 - \mathsf{Pr}(\gamma_{41} = 1 \wedge \eta_{41} = 0 \wedge \eta'_{41} = 0)\Big) \cdot \prod_{j=0}^{2} \Big(1 - \mathsf{Pr}(\mu_j = 1 \wedge \bar{\mu}_j = 0)\Big)$$
$$= 2^{-94} \cdot 2^{-10} \cdot 2^{-2} \cdot \left(\frac{3}{4}\right)^{38} \cdot \frac{7}{8} \cdot \left(\frac{3}{4}\right)^{3} = 2^{-123.21}$$

Note that all the probabilities in the above equation have been calculated over all values of S_{t_1}, S_{t_2} generated during the experiment. Additionally we make the following assumptions:

1. We have assumed that the events (a) $\mathsf{Pr}(\gamma_j = 1)$ and $\mathsf{Pr}(\eta_j = 0)$, (b) $\mathsf{Pr}(\mu_j = 1)$ and $\mathsf{Pr}(\bar{\mu}_j = 0)$ are iid according to $Ber(\frac{1}{2})$.
2. We have also used the fact that $\mathsf{Pr}(\eta_j = 1) = 1$, for $j = 72, 108, 118$ according to Table 2. For all other j, $\mathsf{Pr}(\eta_j = 1) = \frac{1}{2}$ since the p_{t_1+j}'s are all linear functions and hence balanced.
3. Note we have assumed $\mathsf{Pr}(\bar{\mu}_j = 0) = \frac{1}{2}$ since the function f_l is balanced.

Let X_{t_1,t_2} be the indicator variable that is 1 when the tuples at t_1, t_2 are not rejected by the filters, and zero otherwise. Then we have shown that $E(X_{t_1,t_2}) = \rho$. Let P_s be the expected number of pairs that survive during processing keystream generated a single IV. We have

$$P_s = \sum_{i=1}^{N} \sum_{j=i+1}^{N} E[X_{i,j}] = \binom{N}{2} \cdot \rho = D \cdot \rho = 2^{36.79}$$

This is the number of pairs that proceed to the next stage of the attack.

Remark. Note the results presented in Lemma 2, 5, 6, 9 may appear to be arbitrary, but there is a simple technique to find these by analyzing the progression of differences in Grain v1 by a system of integer operations over the algebraic structure (similar to the Δ-GRAIN tool presented in [Ban16]). To model the progression of a difference on LFSR location 79, consider a Grain-like shift-register

structure $\mathsf{N}_t, \mathsf{L}_t$ over the integers, where all $\mathsf{N}_t, \mathsf{L}_t$ are 0 for $t \in [0, 79]$ except $\mathsf{L}_{79} = 1$. The register updates as follows

$$\mathsf{L}_{t+1} = \mathsf{L}_t + \mathsf{L}_{t+13} + \mathsf{L}_{t+23} + \mathsf{L}_{t+38} + \mathsf{L}_{t+51} + \mathsf{L}_{t+62} \bmod 2$$
$$\mathsf{N}_{t+1} = (\mathsf{L}_t + \mathsf{N}_t + \mathsf{N}_{t+14} + \mathsf{N}_{t+62} \bmod 2) + 2 \cdot \mathsf{OR}((\mathsf{N}_{t+60},\ \mathsf{N}_{t+52},\ \mathsf{N}_{t+45},\ \mathsf{N}_{t+37},\ \mathsf{N}_{t+33},\ \mathsf{N}_{t+28},\ \mathsf{N}_{t+21},\ \mathsf{N}_{t+9},\ \mathsf{N}_{t+63},\ \mathsf{N}_{t+15})$$

where OR maps to zero iff all its arguments are 0 else to 1. It can be deduced that any if $\mathsf{N}_t/\mathsf{L}_t$ equals

- 0 $\Rightarrow$ the corresponding difference in the actual Grain v1 state bits are deterministically equal,
- 1 $\Rightarrow$ the corresponding difference is deterministically unequal,
- 2,3 $\Rightarrow$ the corresponding difference is probabilistic.

Define $\chi_t = [\mathsf{N}_{t+1}, \mathsf{N}_{t+2}, \mathsf{N}_{t+4}, \mathsf{N}_{t+10}, \mathsf{N}_{t+31}, \mathsf{N}_{t+43}, \mathsf{N}_{t+56}]$ and $\lambda_t = [\mathsf{L}_{t+3}, \mathsf{L}_{t+25}, \mathsf{L}_{t+46}, \mathsf{L}_{t+64}, \mathsf{N}_{t+63}]$. To deduce Lemmas 2, 5, 6, we run the system forwards and backwards and filter out the time instances χ_t and λ_t have only 0s and 1s. Lemma 2 additionally requires that $\lambda_t = 0^5$. Lemma 5 requires that $\lambda_t \in \{00001, 11000\}$. Lemma 6 requires that $\lambda_t \notin \{00001, 11000, 110001\}$. For Lemma 9 we need $\lambda_t \in \{00002, 00003\}$ and of course that χ_t has only 1s or 0s. To find the higher dimensional differences in Lemma 3, 7, 8, closer scrutiny of the underlying algebraic system is required.

4.4 Online Stage IV: Solving Equation System

Once we have brought down the candidate LFSR states at t_1, t_2 to a much smaller P_s candidates, it is now the ideal time to formulate a set of equations in the NFSR and LFSR states to solve for the entire state. The expression for the output keystream bit at any time t is given as:

$$z_t = n_{t+1} \oplus n_{t+2} \oplus n_{t+4} \oplus n_{t+10} \oplus n_{t+31} \oplus n_{t+43} \oplus n_{t+56} \oplus n_{t+63} \cdot p_t \oplus q_t$$

If L_t is known then p_t and q_t can be easily computed, thus the above expression is a linear equation in either 7 or 8 terms depending on whether p_t is 0 or 1. Make a list of these equations from $t = t_0 - 1$ to $t_0 + 78$ for some value of t_0. This then gives us a set of 80 linear equations in n_{t_0} to n_{t_0+141}. If we guess the 62 values n_{t_0+80} to n_{t_0+141} then it is straightforward to find the values of n_{t_0} to n_{t_0+79}. For example, from the equation

$$z_{t_0+78} = n_{t_0+79} \oplus \bigoplus_{j \in \{80,82,88,109,121,134\}} n_{t_0+j} \oplus \quad n_{t_0+141} \cdot p_{t_0+78} \oplus q_{t_0+78}$$

it is easy to get n_{t_0+79} since all the other variables in the equation are either known or have been guessed. Similarly in the next equation

$$z_{t_0+77} = n_{t_0+78} \oplus \bigoplus_{j \in \{79,81,87,108,120,133\}} n_{t_0+j} \oplus \quad n_{t_0+140} \cdot p_{t_0+77} \oplus q_{t_0+77}$$

n_{t_0+78} is the only unknown and can be found easily. Pursuing a mathematical induction based argument, that shows that all n_{t_0+j} can be similarly computed sequentially.

How many values need to be guessed on average? Note that n_{t_0+136} to n_{t_0+141} appear at most once in these equations, and may not appear at all if the corresponding p_{t_0+72} to p_{t_0+78} are respectively 0. For the remaining values, we can take help of Lemma 6 to reduce the number of guesses, since we are still chasing keystream pairs that are generated by internal states with the single difference in the 79th LFSR location. If the keystream vector pairs are indeed produced by such internal states, then if for some j we have $\gamma_j = 0 \wedge \eta_j = 1$, this immediately implies $n_{t_1+h_j+63} = 0$. Similarly $\gamma_j = 1 \wedge \eta_j = 1$ implies $n_{t_1+h_j+63} = 1$. So we get the value of $n_{t_+h_j+63}$ for free at this stage when $\eta_j = 1$. Again we hope that if the keystream pair is not generated by such a pair of internal states the value of $n_{t_1+h_j+63}$ inferred thus will lead to a contradiction along the line and will be eventually discarded.

4.5 Actual algorithm for Solving

We will select the 80 successive equations z_{t_1+46} to z_{t_1+125}. Denote by $x_i := n_{t_1+47+i}$ for all $i \in [0, 141]$. The idea is to solve for all $x_0 - x_{79}$ by guessing all $x_{80} - x_{141}$. We divide these equations into three distinct segments of 18, 55, 7 equations as shown below:

Segment 1: 18 equations

$$
\begin{aligned}
z_{t_1+46} &= x_0 \oplus x_1 \oplus x_3 \oplus x_9 \oplus x_{30} \oplus x_{42} \oplus x_{55} \oplus x_{62} \cdot p_{t_1+46} \oplus q_{t_1+46} \\
z_{t_1+47} &= x_1 \oplus x_2 \oplus x_4 \oplus x_{10} \oplus x_{31} \oplus x_{43} \oplus x_{56} \oplus x_{63} \cdot p_{t_1+47} \oplus q_{t_1+47} \\
&\vdots \\
z_{t_1+63} &= x_{17} \oplus x_{18} \oplus x_{20} \oplus x_{26} \oplus x_{47} \oplus x_{59} \oplus x_{72} \oplus x_{79} \cdot p_{t_1+63} \oplus q_{t_1+63}
\end{aligned}
$$

Segment 2: 55 equations

$$
\begin{aligned}
z_{t_1+64} &= x_{18} \oplus x_{19} \oplus x_{21} \oplus x_{27} \oplus x_{48} \oplus x_{60} \oplus x_{73} \oplus x_{80} \cdot p_{t_1+64} \oplus q_{t_1+64} \\
z_{t_1+65} &= x_{19} \oplus x_{20} \oplus x_{22} \oplus x_{28} \oplus x_{49} \oplus x_{61} \oplus x_{74} \oplus x_{81} \cdot p_{t_1+65} \oplus q_{t_1+65} \\
&\vdots \\
z_{t_1+118} &= x_{72} \oplus x_{73} \oplus x_{75} \oplus x_{81} \oplus x_{102} \oplus x_{114} \oplus x_{127} \oplus x_{134} \cdot p_{t_1+118} \oplus q_{t_1+118}
\end{aligned}
$$

Segment 3: 7 equations

$$
\begin{aligned}
z_{t_1+119} &= x_{73} \oplus x_{74} \oplus x_{76} \oplus x_{82} \oplus x_{103} \oplus x_{115} \oplus x_{128} \oplus x_{135} \cdot p_{t_1+119} \oplus q_{t_1+119} \\
z_{t_1+120} &= x_{74} \oplus x_{75} \oplus x_{77} \oplus x_{83} \oplus x_{104} \oplus x_{116} \oplus x_{129} \oplus x_{136} \cdot p_{t_1+120} \oplus q_{t_1+120} \\
&\vdots \\
z_{t_1+125} &= x_{79} \oplus x_{80} \oplus x_{82} \oplus x_{88} \oplus x_{109} \oplus x_{121} \oplus x_{134} \oplus x_{141} \cdot p_{t_1+125} \oplus q_{t_1+125}
\end{aligned}
$$

The obvious question is why divide these equations in 3 segments. In Segment 3, if any of p_{t_1+119} to p_{t_1+125} is 0, then respectively the corresponding x_{135} to

x_{141} do not occur in the set of equations. If out of these 7 equations any T_3 of the p_{t_1+j}'s are 0, then this means there are exactly T_3 less variables to guess in this segment. It is easy to see that T_3 is distributed according to $\mathsf{Binomial}(7, \frac{1}{2})$.

In Segment 2, there are 3 values of j: $72, 108, 118$ where $p_{t_1+j} = 1$ with probability 1, and so the corresponding values of x_{88}, x_{124}, x_{134} can be found with certainty as explained above. For example if $z_{t_1+118} \oplus z_{t_2+118} \oplus q_{t_1+118} = 1$, then x_{134} is guaranteed to be 1 too. This being the case the number of variables to guess in this segment comes down to $55 - 3 = 52$. Note according to Table 2, there are 23 other values of j in this segment for which $z_{t_1+j} \oplus z_{t_2+118} \oplus q_{t_1+j}$ is of the form $p_{t_1+j} \cdot n_{t_1+63+j} = p_{t_1+j} \cdot x_{j+16}$. If $p_{t_1+j} = 1$, then we can directly find the value of x_{j+16}. Again if there are T_2 out of 23 p_{t_1+j}'s turn out to be 1 then we have T_2 less variables to guess in this segment.

The question is what fraction of equation systems have $p_{t_1+j} = 1$. Note that at the point we are starting to solve the equations we have P_s candidate keystream pairs and corresponding LFSR states which we assume differ only in location 79. To make things easier, consider a single value of $j = 69$, say. For each of the P_s candidates we have either of the three following equations holding:

1. $p_{t_1+69} = \eta_7 = 1$ and $z_{t_1+69} \oplus z_{t_2+69} \oplus q_{t_1+69} = \gamma_7 = 0$, OR
2. $p_{t_1+69} = \eta_7 = 1$ and $z_{t_1+69} \oplus z_{t_2+69} \oplus q_{t_1+69} = \gamma_7 = 1$, OR
3. $p_{t_1+69} = \eta_7 = 0$ and $z_{t_1+69} \oplus z_{t_2+69} \oplus q_{t_1+69} = \gamma_7 = 0$.

This is because we have already filtered out all those candidates with $p_{t_1+69} = 0$ and $z_{t_1+69} \oplus z_{t_2+69} \oplus q_{t_1+69} = 1$ in one of the previous filtering stages. Thus the fraction θ of the P_s candidates that has $\eta_7 = 1$ is given as

$$\theta = \mathsf{Pr}\Big[\eta_7 = 1 | \sim (\eta_7 = 0 \wedge \gamma_7 = 1)\Big] = \frac{\mathsf{Pr}[\eta_7 = 1 \wedge \sim (\eta_7 = 0 \wedge \gamma_7 = 1)]}{\mathsf{Pr}[\sim (\eta_7 = 0 \wedge \gamma_7 = 1)]}$$
$$= \frac{\mathsf{Pr}[\eta_7 = 1]}{\mathsf{Pr}[\sim (\eta_7 = 0 \wedge \gamma_7 = 1)]} = \frac{\frac{1}{2}}{1 - \frac{1}{4}} = \frac{2}{3}$$

The above follows since the Boolean expression $A \wedge \sim (\sim A \wedge B) = A$. Following this logic it is reasonable to deduce that T_2 is distributed according to $\mathsf{Binomial}(23, \frac{2}{3})$.

In Segment 1, if we have $p_{t_1+j} = 1$ this helps us deduce the value of one of the variables between x_{62} to x_{79}. Table 2 shows that there are 4 such values of j: $51, 54, 57, 62$ where such deduction can occur, i.e. we can deduce the value of $x_{67}, x_{70}, x_{73}, x_{78}$. This does not directly reduce the number of variables to guess. However we can leverage this to reduce the number of guesses. Let $\mathcal{P}, \mathcal{QZ}$ be the 13-element vectors $[p_{t_1+113}, p_{t_1+114}, \ldots, p_{t_1+125}]$, $[q_{t_1+113} \oplus z_{t_1+113}, \ldots, q_{t_1+125} \oplus z_{t_1+125}]$. Note that without any external deduction aided by Table 2, $x_{67}, x_{70}, x_{73}, x_{78}$ is computed by the following 13 steps sequentially, for $i = 0$ to 12:

$$x_{79-i} \leftarrow z_{t_1+125-i} \oplus x_{80-i} \oplus x_{82-i} \oplus x_{88-i} \oplus x_{109-i} \oplus x_{121-i} \oplus x_{134-i} \oplus$$
$$x_{141-i} \cdot p_{t_1+125-i} \oplus q_{t_1+125-i}$$

From here it is straightforward to deduce that $x_{67}, x_{70}, x_{73}, x_{78}$ are affine expressions in $x_{80} - x_{141}$: call them $A_{67}, A_{70}, A_{73}, A_{78}$. Each of these depend on only the actual value of the vectors $\mathcal{P},\ \mathcal{QZ}$. Since there are only 2^{26} possible values of these vectors, one can even precompute all 2^{26} possible values of the affine expressions. Now suppose for example we have deduced the value of $x_{67} = 1, x_{70} = 0$ using the fact that $p_{t_1+51} = p_{t_1+54} = 1$. In that case we know that the variables $x_{80} - x_{141}$ need to be guessed so that $A_{67} = 1$ and $A_{70} = 0$. This reduces the dimension of the total set of guesses of $x_{80} - x_{141}$ by 2. Hence if T_1 of the 4 values are deduced then the dimension of the guess space is reduced by T_1. Additionally it is clear that $T_1 \sim \mathsf{Binomial}(4, \frac{2}{3})$. So the total number of guesses for each of the P_s candidates is $2^{62-3-T_1-T_2-T_3} = 2^{59-\sum T_i}$ with probability $\mathbf{p} = \binom{7}{T_3} \cdot \binom{23}{T_2} \cdot \binom{4}{T_1} \cdot (\frac{1}{2})^7 \cdot (\frac{2}{3})^{T_1+T_2} (\frac{1}{3})^{23+7-T_1-T_2}$. Hence the expected number of candidate internal states after this process is given by

$$N_s = \sum_{T_1=0}^{4} \sum_{T_2=0}^{23} \sum_{T_3=0}^{7} P_s \cdot 2^{59-\sum T_i} \cdot \mathbf{p} \approx 2^{77.09}$$

Note that in all the above process we have ignored the fact that the x_i's so found are additionally related by the Boolean function g. More specifically we must have that $x_{80+i} = g(x_i, x_{i+9}, \ldots, x_{i+63}) \oplus l_{t_1+47+i}$. It is reasonable to assume that each of the N_s candidates for the complete internal state will fail the above equation (for each i) with probability $\frac{1}{2}$, in which case the candidate can be simply eliminated. How many computations of g on average must be done for each candidate before they are discarded? We try to answer this question. The probability that any candidate survives i such equations and is eliminated at the $i+1$-th equation is roughly 2^{-i-1}. Thus in the first stage we have N_s evaluations of g. In the second stage the number of candidates remaining are $\frac{N_s}{2}$. In the third stage we have $\frac{N_s}{4}$ and so on. So the expected evaluations of g is the sum $\sum_{i=0}^{\infty} N_s \cdot 2^{-i} \approx 2 \cdot N_s = 2^{78.09}$, after which we are left with a single surviving candidate for the internal state of Grain v1.

4.6 Formal algorithm and time complexity

Before we end the paper let us state the steps of the algorithm of the Near Collision Attack and compute its time complexity.

A) Precomputation: We do the following pre-computational steps: **1)** Compute the matrices $M^i,\ \forall\ i \in [1, 79]$, and **2)** For the 2^{26} values of the vector $\mathcal{P},\ \mathcal{QZ}$ compute the affine expressions $A_{67}, A_{70}, A_{73}, A_{78}$. These steps take negligible time complexity in comparison with the complexity required in the online stage of the algorithm.

B) Generating Equations: From Lemma 4, we know that we need to generate around $2^{80.5}$ keystream bits to ensure with probability close to one that we do come across with internal states with a single difference in the 79th LFSR location. This obviously means that we need to run $2^{80.5}$ iterations of the Grain v1 update function. As argued in [LN15, EK15, ZLFL14, MAM16] one

stream cipher encryption should be equal to the average number of rounds of the cipher required to be executed per trial with a guessed value of the key (in a brute force search). This comes to 160 initialization rounds and 4 rounds in the keystream generation phase. We have given a proof of this in the appendices of this paper. Thus one Grain v1 round is equivalent to around $\frac{1}{164} = 2^{-7.36}$ encryptions. Thus the task of generating keystream requires time equivalent to around $T_{Gen} = 2^{80.5-7.36} \approx 2^{73.14}$ Grain v1 encryptions.

C) Filtering: In the process of filtering we need to do around $2^{80.5}$ insertions in a hash table. We then need to compute the LFSR state L_{t_1} for every keystream pair that survives the first stage of filtering, the number of which is around $2^{160-104} = 2^{56}$. We have already argued in Section 3, that each such computation is bounded above by at most $2^{21.3}$ bit-operations. And so the the total complexity for this operation is around $2^{56+21.3} = 2^{77.3}$ bit-operations. This is also much less than the time complexity required to generate keystream data. Note that the #bit-operations in one Grain v1 encryption can be taken as 164 times the number of bit-operations required to compute one Grain v1 round, i.e. to compute the functions f, g, h on the Grain v1 internal state. This can be estimated as 164 times the sum of the straight line complexities of evaluating f, g, h. Since each round of Grain v1 requires xor of around 36 different monomials, the number of bit-level operations required in one Grain v1 encryption is likely be in excess of $164 * 36 \approx 2^{12.5}$ by any conservative estimation. Thus we can conclude that computing the LFSR state requires less than $T_{LFSR} \approx 2^{56+21.3-12.5} \approx 2^{64.8}$ encryptions.

Thereafter for each of these candidates we compute all **(a)** L_{t_1+j} for all $j \in [-30, 140]$, which requires running the LFSR part over around 170 iterations, **(b)** then compute $p_{t_1+j},\ q_{t_1+j}$ for all the above values of j. Since $p,\ q$ are component functions of h, the complexity of doing these operations is roughly equal to computing $f,\ h$ over around 170 iterations which is almost equal to the amount of Grain v1 rounds in one encryption. Since computing $f,\ h$ is much easier than computing g, we can estimate that each such computation needs operations less than $\frac{1}{2}$ of a Grain v1 encryption, and so the total complexity of this part is around $T_{F_1} = 2^{56-1} = 2^{55}$ encryptions.

Thereafter for activating the filters in the 2nd/3rd Stage the attacker needs to compute a series of expressions, most of which require only a few bit-xors i.e. $\beta_0, \beta_1, \{\gamma_j\}_{j=0}^{41}$, $\{\mu_j, \bar{\mu}_j\}_{j=0}^{2}$. Computing these is much less than computing the $L_{t_1+j}, p_{t_1+j}, q_{t_1+j}$'s and so T_{F_1} is the most dominant complexity term.

D) Solve equations: Note that only $P_s \approx 2^{36.79}$ candidates survive this stage. For each of these candidates, we first determine the dimension of guess space by first determining T_2, T_3 with the help of the p_{t_1+j}'s and then determining T_1 with the precomputed expressions $A_{67}, A_{70}, A_{73}, A_{78}$ with the help of the vectors $\mathcal{P},\ \mathcal{QZ}$. This generates around N_s candidates each consisting of **(a)** the keystream vector pairs, **(b)** the corresponding $L_{t_1+j}, p_{t_1+j}, q_{t_1+j}$'s, and **(c)** valid guesses of $x_{80} - x_{141}$. From each candidate computing $x_0 - x_{79}$ can be done in sequence one after the other with around 7-8 bit-xor operations for each x_i, so around $80 * 8 = 640$ bit-xors in total. This is much less than

the number of bit-operations in $640 * 2^{-12.5} \approx 2^{-3.2}$ encryptions and so this complexity is bounded above by $T_{Eq} \approx N_s \cdot 2^{-3.2} \approx 2^{73.89}$ encryptions.

E) Eliminate candidates: As already explained this required $2 \cdot N_s \approx 2^{78.09}$ evaluations of g. Since one evaluation of g is computationally less than that required in $\frac{1}{164} \approx 2^{-7.36}$ encryptions, the complexity of this part is bounded from above by $T_{El} = 2 \cdot N_s \cdot 2^{-7.36} \approx 2^{70.73}$ encryptions.

The total complexity of the algorithm is given by $T_{total} = T_{Gen} + T_{LFSR} + T_{F_1} + T_{Eq} + T_{El}$. The most dominant terms in the sum are T_{Gen}, T_{Eq} and so the total complexity is around $T_{Total} \approx 2^{74.6}$ encryptions.

4.7 Extending attacks to Grain-128 and Grain-128a

The attacks described in this and the previous sections can be analogous applied to Grain-128 [HJMM06a] and Grain-128a [AHJM11a]. Since the core steps are very similar we move the full description of the attack to the appendices (Appendix D).

5 Conclusion

In this paper we look back at near collision style attacks on Grain v1. After a similar attack on Grain v1 [ZXM18] was disproved in [DFM20], it has been a matter of debate whether such attacks can be applied to Grain v1. In this paper we answer the question in the affirmative, but just barely. The attack we propose takes $2^{74.6}$ Grain v1 encryptions and settles this open question. The paper shows that sparseness of taps in the output function as in Grain v1 can be a weakness even if the state size is twice that of the key size.

References

AHJM11a. Martin Agren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *International Journal of Wireless and Mobile Computing*, 5(1):48–59, 2011.

ÅHJM11b. 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.

AM08. Mehreen Afzal and Ashraf Masood. Algebraic Cryptanalysis of A NLFSR Based Stream Cipher. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6, 2008.

Ban16. Subhadeep Banik. Conditional differential cryptanalysis of 105 round Grain v1. *Cryptogr. Commun.*, 8(1):113–137, 2016.

BBI19. Subhadeep Banik, Khashayar Barooti, and Takanori Isobe. Cryptanalysis of Plantlet. *IACR Trans. Symmetric Cryptol.*, 2019(3):103–120, 2019.

BCI+21. Subhadeep Banik, Andrea Caforio, Takanori Isobe, Fukang Liu, Willi Meier, Kosei Sakamoto, and Santanu Sarkar. Atom: A stream cipher with double key filter. *IACR Trans. Symmetric Cryptol.*, 2021(1):5–36, 2021.

Bjo08. Tor E. Bjorstad. Cryptanalysis of Grain using Time/Memory/Date Tradeoffs. eSTREAM, ECRYPT Stream Cipher Project, Report 2008/012, 2008. `https://www.ecrypt.eu.org/stream/papersdir/2008/012.pdf`.

DFM20. Patrick Derbez, Pierre-Alain Fouque, and Victor Mollimard. Fake near collisions attacks. *IACR Trans. Symmetric Cryptol.*, 2020(4):88–103, 2020.

DGP+11. 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.

DS11. 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.

EK15. Muhammed F. Esgin and Orhun Kara. Practical Cryptanalysis of Full Sprout with TMD Tradeoff Attacks. In *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 67–85, 2015.

HJM05. Martin Hell, Thomas Johansson, and Willi Meier. Grain - A Stream Cipher for Constrained Environments. eSTREAM, ECRYPT Stream Cipher Project Report, 2005. `http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf`.

HJM+19. Martin Hell, Thomas Johansson, Alexander Maximov, Willi Meier, Jonathan Sönnerup, and Hirotaka Yoshida. Grain-128aeadv2 - a lightweight aead stream cipher. *NIST Lightweight Cryptography Project*, 2019.

HJMM06a. 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.

HJMM06b. 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.

iso. Iso/iec 29167-13:2015 information technology – automatic identification and data capture techniques – part 13: Crypto suite grain-128a security services for air interface communications.

KMN10. 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.

LM12. 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.

LN15. Virginie Lallemand and María Naya-Plasencia. Cryptanalysis of Full Sprout. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 663–682, 2015.

lwc. Nist lightweight cryptography project. `https://csrc.nist.gov/projects/lightweight-cryptography`.

MAM16. 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.

RB08. Matthew J. B. Robshaw and Olivier Billet, editors. *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*. Springer, 2008.

TIM+18. 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 *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 129–159, 2018.

ZLFL14. 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*, pages 518–538, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

ZXM18. Bin Zhang, Chao Xu, and Willi Meier. Fast Near Collision Attack on the Grain v1 Stream Cipher. In *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*, pages 771–802, 2018.

A Cost of executing one round of Grain v1

To do an exhaustive search, first an initialization phase has to be run for 160 rounds, after which 80-bits of keystream is generated to do a unique match. However, since each keystream bit generated matches the correct one with probability $\frac{1}{2}$, 2^{80} keys are tried for 1 clock and roughly half of them are eliminated, 2^{79} for 2 clocks and half of the remaining keys are eliminated, and so on. This means that in the process of brute force search, the probability that for any random key, $(i+1)$ Grain v1 keystream phase rounds need to be run, is $\frac{1}{2^i}$. Hence, the expected number of Grain v1 rounds per trial is

$$\sum_{i=0}^{79} \frac{(i+1)2^{80-i}}{2^{80}} = \sum_{i=0}^{79} (i+1)\frac{1}{2^i} \approx 4$$

Adding to this the 160 rounds in the initialization phase, the average number of Grain v1 rounds per trial is 164. As a result, we will assume that clocking the registers once will cost roughly $\frac{1}{160+4} = 2^{-7.36}$ encryptions.

B Difference Vectors

#	j	$\mathbf{v}$	p_{t_1+j}	q_{t_1+j}
0	15	$[0,0,0,1,0]$	$l_{t_1+61} \oplus 1$	$l_{t_1+18} \cdot l_{t_1+61} \oplus l_{t_1+18} \oplus l_{t_1+61}$
1	33	$[0,0,1,1,0]$	$l_{t_1+36} \oplus l_{t_1+58} \oplus l_{t_1+79} \oplus l_{t_1+97}$	$l_{t_1+36} \cdot l_{t_1+58} \oplus l_{t_1+36} \cdot l_{t_1+79} \oplus$ $l_{t_1+36} \cdot l_{t_1+97} \oplus l_{t_1+79} \oplus l_{t_1+97} \oplus 1$
2	44	$[0,0,0,1,0]$	$l_{t_1+90} \oplus 1$	$l_{t_1+47} \cdot l_{t_1+90} \oplus l_{t_1+47} \oplus l_{t_1+90}$
3	51	$[0,0,1,1,0]$	$l_{t_1+54} \oplus l_{t_1+76} \oplus l_{t_1+97} \oplus l_{t_1+115}$	$l_{t_1+54} \cdot l_{t_1+76} \oplus l_{t_1+54} \cdot l_{t_1+97} \oplus$ $l_{t_1+54} \cdot l_{t_1+115} \oplus l_{t_1+97} \oplus l_{t_1+115} \oplus 1$
4	54	$[0,1,0,0,0]$	l_{t_1+100}	$l_{t_1+57} \cdot l_{t_1+100} \oplus 1$
5	57	$[0,0,0,1,0]$	$l_{t_1+103} \oplus 1$	$l_{t_1+60} \cdot l_{t_1+103} \oplus l_{t_1+60} \oplus l_{t_1+103}$
6	62	$[0,0,1,0,0]$	$l_{t_1+65} \oplus l_{t_1+87} \oplus l_{t_1+126}$	$l_{t_1+65} \cdot l_{t_1+87} \oplus l_{t_1+65} \cdot l_{t_1+126} \oplus l_{t_1+126}$
7	69	$[0,0,1,1,0]$	$l_{t_1+72} \oplus l_{t_1+94} \oplus l_{t_1+115} \oplus l_{t_1+133}$	$l_{t_1+72} \cdot l_{t_1+94} \oplus l_{t_1+72} \cdot l_{t_1+115} \oplus$ $l_{t_1+72} \cdot l_{t_1+133} \oplus l_{t_1+115} \oplus l_{t_1+133} \oplus 1$
8	72	$[0,1,0,1,0]$	1	$l_{t_1+75} \oplus l_{t_1+118} \oplus 1$
9	73	$[0,0,0,1,0]$	$l_{t_1+119} \oplus 1$	$l_{t_1+76} \cdot l_{t_1+119} \oplus l_{t_1+76} \oplus l_{t_1+119}$
10	75	$[0,0,1,0,0]$	$l_{t_1+78} \oplus l_{t_1+100} \oplus l_{t_1+139}$	$l_{t_1+78} \cdot l_{t_1+100} \oplus l_{t_1+78} \cdot l_{t_1+139} \oplus l_{t_1+139}$
11	76	$[1,0,0,0,0]$	l_{t_1+122}	$l_{t_1+101} \cdot l_{t_1+122} \oplus l_{t_1+122} \cdot l_{t_1+140} \oplus l_{t_1+140}$
12	80	$[0,0,0,1,0]$	$l_{t_1+126} \oplus 1$	$l_{t_1+83} \cdot l_{t_1+126} \oplus l_{t_1+83} \oplus l_{t_1+126}$
13	82	$[0,0,0,1,0]$	$l_{t_1+128} \oplus 1$	$l_{t_1+85} \cdot l_{t_1+128} \oplus l_{t_1+85} \oplus l_{t_1+128}$
14	83	$[0,1,0,0,0]$	l_{t_1+129}	$l_{t_1+86} \cdot l_{t_1+129} \oplus 1$
15	87	$[0,0,1,1,0]$	$l_{t_1+90} \oplus l_{t_1+112} \oplus l_{t_1+133} \oplus l_{t_1+151}$	$l_{t_1+90} \cdot l_{t_1+112} \oplus l_{t_1+90} \cdot l_{t_1+133} \oplus$ $l_{t_1+90} \cdot l_{t_1+151} \oplus l_{t_1+133} \oplus l_{t_1+151} \oplus 1$
16	90	$[0,1,1,0,0]$	$l_{t_1+93} \oplus l_{t_1+115} \oplus l_{t_1+136} \oplus l_{t_1+154} \oplus 1$	$l_{t_1+93} \cdot l_{t_1+115} \oplus l_{t_1+93} \cdot l_{t_1+136} \oplus$ $l_{t_1+93} \cdot l_{t_1+154} \oplus l_{t_1+93} \oplus l_{t_1+154} \oplus 1$
17	91	$[0,0,1,1,0]$	$l_{t_1+94} \oplus l_{t_1+116} \oplus l_{t_1+137} \oplus l_{t_1+155}$	$l_{t_1+94} \cdot l_{t_1+116} \oplus l_{t_1+94} \cdot l_{t_1+137} \oplus$ $l_{t_1+94} \cdot l_{t_1+155} \oplus l_{t_1+137} \oplus l_{t_1+155} \oplus 1$
18	93	$[0,0,0,1,0]$	$l_{t_1+139} \oplus 1$	$l_{t_1+96} \cdot l_{t_1+139} \oplus l_{t_1+96} \oplus l_{t_1+139}$
19	94	$[1,0,0,0,0]$	l_{t_1+140}	$l_{t_1+119} \cdot l_{t_1+140} \oplus l_{t_1+140} \cdot l_{t_1+158} \oplus l_{t_1+158}$
20	95	$[0,0,0,1,0]$	$l_{t_1+141} \oplus 1$	$l_{t_1+98} \cdot l_{t_1+141} \oplus l_{t_1+98} \oplus l_{t_1+141}$
21	96	$[0,1,0,0,1]$	l_{t_1+142}	$l_{t_1+121} \cdot l_{t_1+142} \oplus l_{t_1+142} \cdot l_{t_1+160} \oplus l_{t_1+142} \oplus l_{t_1+160}$
22	98	$[0,0,1,0,0]$	$l_{t_1+101} \oplus l_{t_1+123} \oplus l_{t_1+162}$	$l_{t_1+101} \cdot l_{t_1+123} \oplus l_{t_1+101} \cdot l_{t_1+162} \oplus l_{t_1+162}$
23	99	$[0,0,0,1,0]$	$l_{t_1+145} \oplus 1$	$l_{t_1+102} \cdot l_{t_1+145} \oplus l_{t_1+102} \oplus l_{t_1+145}$
24	100	$[0,0,1,0,0]$	$l_{t_1+103} \oplus l_{t_1+125} \oplus l_{t_1+164}$	$l_{t_1+103} \cdot l_{t_1+125} \oplus l_{t_1+103} \cdot l_{t_1+164} \oplus l_{t_1+164}$
25	102	$[0,0,0,1,0]$	$l_{t_1+148} \oplus 1$	$l_{t_1+105} \cdot l_{t_1+148} \oplus l_{t_1+105} \oplus l_{t_1+148}$
26	105	$[1,0,1,1,0]$	$l_{t_1+108} \oplus l_{t_1+130} \oplus l_{t_1+169} \oplus 1$	$l_{t_1+108} \cdot l_{t_1+130} \oplus l_{t_1+108} \cdot l_{t_1+151} \oplus$ $l_{t_1+108} \cdot l_{t_1+169} \oplus l_{t_1+130} \cdot l_{t_1+151} \oplus$ $l_{t_1+130} \oplus l_{t_1+151} \cdot l_{t_1+169} \oplus l_{t_1+169} \oplus 1$
27	108	$[0,1,0,1,0]$	1	$l_{t_1+111} \oplus l_{t_1+154} \oplus 1$
28	109	$[0,0,1,0,0]$	$l_{t_1+112} \oplus l_{t_1+134} \oplus l_{t_1+173}$	$l_{t_1+112} \cdot l_{t_1+134} \oplus l_{t_1+112} \cdot l_{t_1+173} \oplus l_{t_1+173}$
29	111	$[0,1,1,0,0]$	$l_{t_1+114} \oplus l_{t_1+136} \oplus l_{t_1+157} \oplus l_{t_1+175} \oplus 1$	$l_{t_1+114} \cdot l_{t_1+136} \oplus l_{t_1+114} \cdot l_{t_1+157} \oplus$ $l_{t_1+114} \cdot l_{t_1+175} \oplus l_{t_1+114} \oplus l_{t_1+175} \oplus 1$
30	115	$[0,0,0,1,0]$	$l_{t_1+161} \oplus 1$	$l_{t_1+118} \cdot l_{t_1+161} \oplus l_{t_1+118} \oplus l_{t_1+161}$
31	117	$[0,0,1,1,0]$	$l_{t_1+120} \oplus l_{t_1+142} \oplus l_{t_1+163} \oplus l_{t_1+181}$	$l_{t_1+120} \cdot l_{t_1+142} \oplus l_{t_1+120} \cdot l_{t_1+163} \oplus$ $l_{t_1+120} \cdot l_{t_1+181} \oplus l_{t_1+163} \oplus l_{t_1+181} \oplus 1$
32	118	$[1,0,0,1,0]$	1	$l_{t_1+121} \cdot l_{t_1+164} \oplus l_{t_1+121} \oplus l_{t_1+143} \cdot l_{t_1+164} \oplus$ $l_{t_1+164} \cdot l_{t_1+182} \oplus l_{t_1+182} \oplus 1$
33	119	$[0,1,0,0,0]$	l_{t_1+165}	$l_{t_1+122} \cdot l_{t_1+165} \oplus 1$
34	121	$[0,1,0,0,0]$	l_{t_1+167}	$l_{t_1+124} \cdot l_{t_1+167} \oplus 1$
35	126	$[0,1,1,0,0]$	$l_{t_1+129} \oplus l_{t_1+151} \oplus l_{t_1+172} \oplus l_{t_1+190} \oplus 1$	$l_{t_1+129} \cdot l_{t_1+151} \oplus l_{t_1+129} \cdot l_{t_1+172} \oplus$ $l_{t_1+129} \cdot l_{t_1+190} \oplus l_{t_1+129} \oplus l_{t_1+190} \oplus 1$
36	128	$[0,0,0,1,0]$	$l_{t_1+174} \oplus 1$	$l_{t_1+131} \cdot l_{t_1+174} \oplus l_{t_1+131} \oplus l_{t_1+174}$
37	132	$[0,1,0,0,1]$	l_{t_1+178}	$l_{t_1+157} \cdot l_{t_1+178} \oplus l_{t_1+178} \cdot l_{t_1+196} \oplus l_{t_1+178} \oplus l_{t_1+196} \oplus 1$
38	135	$[0,0,1,0,0]$	$l_{t_1+138} \oplus l_{t_1+160} \oplus l_{t_1+199}$	$l_{t_1+157} \cdot l_{t_1+178} \oplus l_{t_1+178} \cdot l_{t_1+196} \oplus l_{t_1+178} \oplus l_{t_1+196}$
39	-4	$[1,0,0,0,0]$	l_{t_1+42}	$l_{t_1+21} \cdot l_{t_1+42} \oplus l_{t_1+42} \cdot l_{t_1+60} \oplus l_{t_1+60}$

Table 2. Table of the functions $p_{t_1+j},\ q_{t_1+j}$ for various j.

C Reducing the memory footprint

The attack in the main body requires $2^{80.5}$ hash table insertions. In general, the time to access a table could be considered to be more than a typical cryptographic operation, and unless we have the technology or means to perform disk

access in short enough time, it is unclear whether the physical time needed to do table insertions is less than the that required of exhaustive search.

In this section we try to reduce the number of table accesses using the algebraic structure of the Grain v1. We use the fact that the sampling resistance of Grain v1 is very low. In general if it is efficient to enumerate all internal states of a stream cipher that generates keystream segments with some constant R-bit prefix, then the sampling resistance of a stream cipher is said to be 2^{-R}. The following result is well known:

Lemma 10. *[Bjo08] The sampling resistance of Grain v1 is at most* 2^{-18}.

Proof. Although it was proven in [Bjo08], for completeness, we give a brief proof sketch. From the output equation of Grain v1, we can see that

$$\begin{aligned} n_{t+10} &= z_t \quad \oplus_{i\in\{1,2,4,31,43,56\}} n_{t+i} \oplus h(l_{t+3}, l_{t+25}, l_{t+46}, l_{t+64}, n_{t+63}) \\ &\vdots \\ n_{t+25} &= z_{t+15} \oplus_{i\in\{16,17,19,46,58,71\}} n_{t+i} \oplus h(l_{t+18}, l_{t+40}, l_{t+61}, l_{t+79}, n_{t+78}) \end{aligned} \tag{1}$$

Now, fixing the keystream prefix $z_t, z_{t+1}, \ldots z_{t+17}$ to some constant, say 0^{18}, we can determine 16 state bits n_{t+10} to n_{t+25} sequentially, by simply guessing all the other state bits that appear on the right side of the above equation list, whose values have not been already determined in any of the previous steps. This kind of enumeration is further made possible since at each step the computed NFSR bit (i.e. on the left side of each equation) does not occur in any of the previous equations. In [Bjo08], it was additionally shown that n_{t+26}, n_{t+27} can also be deduced using this technique by using $z_{t+\{16,17\}}$ and some other state bits. Hence the result follows.

The technique we will use to reduce the hash table insertions is similar to collision search with privileged points, in the sense that the strategy will be to only store keystream vectors with some easily identifiable property: let's say during the search for keystream collision we only insert in table the tuple (t, Z_t, Y_t) iff Z_t begins with r consecutive 0s, for some $r \leq 15$ (it will be clear shortly why we choose 15 as a bound for r instead of 18). If this is the strategy, we have to work out how many key stream bits we have to generate before we detect two states S_{t_1}, S_{t_2} such that $S_{t_1} \oplus S_{t_2} = 0^{80}||e_{79}$.

C.1 The effect of low sampling resistance

In general what low sampling resistance does is that it allows us to have a shorter description of all internal states that produce keystream with a constant r bit prefix, for any r less than R, where 2^{-R} is the sampling resistance. For example, in Grain v1, we know that for all states generating keystream with the 0^{18} prefix, the 18 state bits n_{t+10} to n_{t+27} can be uniquely determined from values of the other 142 state bits. Thus given any 142-bit input X we can define

a bijective map $B : \{0,1\}^{142} \to \{0,1\}^{160}$ such that $B(X)$ represents the 160-bit internal state of Grain v1 that produces keystream starting with 18 zeros. Thus X is in a sense a shorter description of $B(X)$ which is of only 142 bits. This exercise can be carried out for any r less than 18, i.e. we can define bijective maps $B_r : \{0,1\}^{160-r} \to \{0,1\}^{160}$ similarly.

When we limit $r \leq 15$, we only use up to the first 15 equations in Equation (1) to construct the bijective map, i.e. from n_{t+10} to n_{t+24}. This ensures that l_{t+79} is not involved in any of the expressions. Thus, if we have two short states X_1, X_2, then

$$B_r(X_1) \oplus B_r(X_2) = 0^{80}||e_{79} \Leftrightarrow X_1 \oplus X_2 = 0^{80-r}||e_{79}.$$

Now when we store only keystream vectors that begin with 0^r, we will get a collision $B_r(X_1) \oplus B_r(X_2) = 0^{80}||e_{79}$ if and only if we get a collision between the shorter states X_1, X_2 such that $X_1 \oplus X_2 = 0^{80-r}||e_{79}$. By standard Birthday assumptions, if the X_i's are generated randomly we can see that this occurs when we generate $\sqrt{2 \cdot 2^{160-r}} = 2^{80.5-r/2}$ short states. This now becomes the number of times we have to insert keystream tuples in the table. By standard randomness assumptions, we get a 0^r keystream prefix every 2^r rounds of Grain v1 on average. Thus to generate $\sqrt{2 \cdot 2^{160-r}}$ short states, we have to run the cipher for $2^r \cdot \sqrt{2 \cdot 2^{160-r}} = 2^{80.5+r/2}$ iterations. Thus we see that it gives a clear tradeoff: we decrease the number of memory accesses by a factor of $2^{r/2}$ at the cost of increasing the run time of the cipher by the same factor.

C.2 Modified algorithm and time complexity

The modified algorithm is exactly the same as the original algorithm with the only difference that instead of all t, we only insert a tuple (t, Z_t, Y_t) in the table if Z_t begins with r consecutive 0s for some $r \leq 15$. Note that $z_t, z_{t+1}, \ldots, z_{t+14}$ are not involved in any of the filtering processes and so the rest of that algorithm can be exactly the same as the algorithm detailed in Section 4.6. Thus the only modifications in the complexity estimates are as follows:

1. We have seen that T_{Gen} increases by a factor $2^{r/2}$. Taking $r = 10$ gives us $T_{Gen} \approx 2^{78.14}$. The total attack complexity is given by $T_{total} = T_{Gen} + T_{LFSR} + T_{F_1} + T_{Eq} + T_{El} \approx 2^{78.14}$, since T_{Gen} is the dominant term.
2. The number of table insertions decrease by a factor $2^{r/2}$, Again taking $r = 10$, reduces the number of insertions to $2^{75.5}$.
3. We can do away with storing the first r bits of Z_t. Thus the total memory required is $(250 - r) \cdot 2^{75.5} \approx 2^{83}$ bits for $r = 10$.

C.3 Comparison with generic attacks on stream ciphers

We consider generic key recovery and state recovery attacks on Grain v1. The generic key recovery attack proceeds by the adversary testing every key in the keyspace and comparing the generated key stream with some key stream from an encryption oracle. It requires 2^{80} encryptions in the worst case.

We also need to calculate the complexity of generic state recovery attacks on stream ciphers. The generic attack proceeds as follows:

- Generate N bits of keystream.
- Generate 160 bits of keystream from T random internal states.
- Look for a collision.

The cost of this attack is N data, $160T + N$ iterations of the stream cipher, and we must have $NT \geq 2^{160}$. By minimizing $160T + N$, one can see that there is an attack in $2\sqrt{160} \cdot 2^{80}$ iterations. This is equivalent to around $2^{77.3}$ Grain v1 encryptions using around $2^{83.7}$ bits of memory with $2^{76.3}$ hash table insertions. By contrast, our attack with sampling resistance $r = 8$ requires $2^{77.14}$ encryptions, $2^{84.4}$ bits of memory and $2^{76.5}$ insertions in the hash table.

D Extending the attack to Grain-128 and Grain-128a

In this appendix, we apply similar techniques to perform a state recovery attack on the 128-bit cipher Grain-128 [HJMM06b] and Grain-128a [AHJM11a]. That is, we first generate a near collision on the keystream. Then, for each candidate LFSR state determined by algebraic relations induced by the near collision, we try to solve a set of equations to either recover the entire NFSR/LFSR state or reach a contradiction, then repeat if necessary.

We generalize the notation introduced in Section 2, e.g. letting $L_t = [l_t, l_{t+1}, \ldots, l_{t+127}]$ be the LFSR state at the t-th clock interval. Grain-128 consists of a 128-bit LFSR and a 128-bit NFSR, and uses an 128-bit key K. Grain-128's LFSR is defined by the update function f given by

$$f(Y_t) = l_{t+96} + l_{t+81} + l_{t+70} + l_{t+38} + l_{t+7} + l_t$$

The NFSR state is updated as $n_{t+128} = l_t + g(\cdot)$ for NFSR update function g, which is given by

$$g(X_t) = n_{t+96} + n_{t+91} + n_{t+56} + n_{t+26} + n_t + n_{t+3}n_{t+67} + n_{t+11}n_{t+13} + n_{t+17}n_{t+18} + n_{t+27}n_{t+59} + n_{t+40}n_{t+48} + n_{t+61}n_{t+65} + n_{t+68}n_{t+84}$$

The output function is of the form

$$z_t = h'(X_t, Y_t) = \bigoplus_{a \in A} n_{t+a} + h(s_0, \ldots, s_8) + l_{93}$$

where $A = \{2, 15, 36, 45, 64, 73, 89\}$, $h(s_0, \ldots, s_8) = s_0s_1 + s_2s_3 + s_4s_5 + s_6s_7 + s_0s_4s_8$, and $(s_0, \ldots, s_8) = (n_{t+12}, l_{t+8}, l_{t+13}, l_{t+20}, n_{t+95}, l_{t+42}, l_{t+60}, l_{t+79}, l_{t+95})$. 256 clocks of initialization are executed before entering the keystream phase.

We first note that an analogous result to Lemma 1 can be shown which allows us to efficiently compute the LFSR states L_{t_1} and L_{t_2} given t_1, t_2 and $\delta = L_{t_1} \oplus L_{t_2}$. By essentially the same analysis as in Section 3, one can solve for L_{t_1} and L_{t_2} in at most $5 \cdot 128^3 \approx 2^{23.3}$ bit-operations. We can also state an analogous result on the probability of finding a suitable near collision.

Lemma 11. *In the event that we generate N iterations of internal states of Grain-128 sequentially: $S_i \in \{0,1\}^{256}$, for $i \in [0, N-1]$, then the probability that there is at least one tuple $i_1, i_2 \in [0, N-1]$ and $i_2 > i_1$, such that, $S_{i_1} \oplus S_{i_2} = 0^{128}||e_{127}$, is approximately $p_{coll} = \frac{N^2}{2^{257}}$.*

Thus for $N \approx 2^{128.5}$ we expect to find the desired near collision once. We first state the core result analogous to Lemma 2.

Lemma 12. *Consider two internal states in Grain-128, $S_{t_1} = (N_{t_1}, L_{t_1})$ and $S_{t_2} = (N_{t_2}, L_{t_2})$ during the keystream phase such that $S_{t_1} \oplus S_{t_2} = 0^{128}||e_{127}$, i.e. $N_{t_1} = N_{t_2}$ and $L_{t_1} \oplus L_{t_2} = e_{127}$. Then consider*

$$Z_{t_i} = [z_{t_i+0}, z_{t_i+1}, z_{t_i+2}, \ldots, z_{t_i+255}],\ Y_{t_i} = [z_{t_i-1}, z_{t_i-2}, z_{t_i-3}, \ldots, z_{t_i-11}].$$

*for $i = 1, 2$. Then in the 267 bit difference vector $\Delta = Z_{t_1}||Y_{t_1} \oplus Z_{t_2}||Y_{t_2}$, there are **150** bits that take the value 1 or 0 with probability 1.*

Proof. Considering the forward difference vector, we have $z_{t_1+j} \oplus z_{t_2+j} = 0$ for $j \in [0, 31] \cup \{33\} \cup [35, 47] \cup [49, 62] \cup \{65\} \cup [68, 78] \cup [82, 84] \cup [86, 89] \cup \{91, 93, 94, 97\} \cup [100, 105] \cup [108, 111] \cup \{113, 115, 116, 118, 120, 121, 123, 127, 129\} \cup [133, 137] \cup \{140, 141, 147, 152, 162, 167, 168, 176, 182, 185, 187, 190, 198, 199, 205, 231\}$, and $z_{t_1+j} + z_{t_2+j} = 1$ for $j \in \{34, 66, 81, 92, 98, 124, 130, 145, 150, 155, 156, 191, 194, 214\}$. In the backwards direction, we have $z_{t_1-j} \oplus z_{t_2-j} = 0$ for $j \in \{1, 2, 4, 5, 7, 8, 11\}$ and $z_{t_1-j} + z_{t_2-j} = 1$ for $j \in \{3, 10\}$. □

We now state some lemmas containing additional relations induced by the near collision. These properties can be argued to hold analogously as done for Grain v1 in Section 3 and 4. Similarly to with Grain v1, we use Lemmas 12 to 17 for filtering and Lemmas 17 to 19 to assist with equation solving in the attack.

Lemma 13. *Consider again the conditions in the previous lemma. We have $z_{t_1+99} \oplus z_{t_2+99} \oplus z_{t_1+165} \oplus z_{t_2+165} = 0$, $z_{t_1+106} \oplus z_{t_2+106} \oplus z_{t_1+146} \oplus z_{t_2+146} = 0$ and $z_{t_1+125} \oplus z_{t_2+125} \oplus z_{t_1+144} \oplus z_{t_2+144} = 0$ with probability 1.*

Lemma 14. *Consider again the conditions in the previous lemma. For the 29 pairs $(i, j) \in \{(48, 108), (67, 146), (80, 140), (95, 155), (99, 178), (106, 166), (107, 120), (112, 172), (125, 204), (131, 210), (138, 198), (139, 152), (142, 202), (144, 204), (146, 166), (157, 236), (159, 219), (163, 242), (164, 224), (165, 178), (169, 229), (170, 230), (171, 184), (172, 192), (174, 324), (193, 272), (200, 260), (202, 262), (225, 245)\}$, we have $z_{t_1+i} \oplus z_{t_2+i} = l_{t_1+j}$. For the 5 pairs $(i, j) \in \{(188, 267), (203, 216), (219, 298), (222, 282), (253, 332)\}$, we have $z_{t_1+i} \oplus z_{t_2+i} = l_{t_1+j} \oplus 1$.*

Lemma 15. *Consider again the conditions in the previous lemma. We have the following 7 identities with probability 1.*

$$z_{t_1+114} \oplus z_{t_2+114} = l_{t_1+134} \oplus l_{t_1+193}, \quad z_{t_1+161} \oplus z_{t_2+161} = l_{t_1+181} \oplus l_{t_1+240}$$
$$z_{t_1+178} \oplus z_{t_2+178} = l_{t_1+198} \oplus l_{t_1+257}, \quad z_{t_1+189} \oplus z_{t_2+189} = l_{t_1+249} \oplus l_{t_1+268} \oplus 1$$
$$z_{t_1+208} \oplus z_{t_2+208} = l_{t_1+228} \oplus l_{t_1+268} \oplus l_{t_1+287}$$
$$z_{t_1+228} \oplus z_{t_2+228} = l_{t_1+241} \oplus l_{t_1+288} \oplus 1,\ z_{t_1+236} \oplus z_{t_2+236} = l_{t_1+256} \oplus l_{t_1+296}$$

Lemma 16. *Consider again the conditions in the previous lemma. For the 10 pairs* $(i,j) \in \{(85,180),(117,212),(119,131),(132,227),(149,244),(151,163),(177,189),(179,274),(181,276),(237,332)\}$*, we have* $z_{t_1+i} \oplus z_{t_2+i} = n_{t_1+j}$*. For the 3 pairs* $(i,j) \in \{(166,178),(175,270),(209,221)\}$*, we have* $z_{t_1+i} \oplus z_{t_2+i} = n_{t_1+j} \oplus 1$*.*

Lemma 17. *We have the following 6 identities with probability 1.*

$$\begin{aligned}
z_{t_1+160} \oplus z_{t_2+160} \oplus l_{t_1+202} \oplus n_{t_1+172} \cdot l_{t_1+255} &= 1 \\
z_{t_1+197} \oplus z_{t_2+197} \oplus l_{t_1+210} \oplus l_{t_1+239} \oplus n_{t_1+209} \cdot l_{t_1+292} &= 1 \\
z_{t_1+224} \oplus z_{t_2+224} \oplus l_{t_1+266} \oplus n_{t_1+236} \cdot l_{t_1+319} &= 1 \\
z_{t_1+232} \oplus z_{t_2+232} \oplus l_{t_1+274} \oplus l_{t_1+292} \oplus n_{t_1+244} \cdot l_{327} &= 0 \\
z_{t_1+234} \oplus z_{t_2+234} \oplus l_{t_1+276} \oplus l_{t_1+294} \oplus n_{t_1+246} \cdot l_{329} &= 0 \\
z_{t_1+255} \oplus z_{t_2+255} \oplus l_{t_1+268} \oplus l_{t_1+275} \oplus l_{t_1+297} \oplus l_{t_1+334} \oplus n_{t_1+267} \cdot l_{t_1+350} &= 0
\end{aligned}$$

Lemma 18. *Consider again the conditions in the previous lemma. For the 19 values* $i \in \{32,64,79,90,96,122,126,148,153,158,173,180,184,192,195,212,216,217,238\}$*, we have* $z_{t_1+i} \oplus z_{t_2+i} = n_{t_1+12+i} \cdot n_{t_1+95+i}$*. For the 3 values* $i \in \{128,186,250\}$*, we have* $z_{t_1+i} \oplus z_{t_2+i} = n_{t_1+12+i} \cdot n_{t_1+95+i} \oplus 1$*. For the 2 values* $i \in \{143,206\}$*, we have:* $z_{t_1+i} \oplus z_{t_2+i} = (1 \oplus n_{t_1+12+i}) \cdot n_{t_1+95+i}$*.*

Lemma 19. *Consider again the conditions in the previous lemma. We have the following 11 identities with probability 1.*

$$\begin{aligned}
z_{t_1+154} \oplus z_{t_2+154} &= l_{t_1+167} \oplus n_{t_1+166} \cdot n_{t_1+249} \\
z_{t_1+183} \oplus z_{t_2+183} &= l_{t_1+262} \oplus n_{t_1+195}, \quad z_{t_1+196} \oplus z_{t_2+196} = l_{t_1+256} \oplus n_{t_1+291} \\
z_{t_1+207} \oplus z_{t_2+207} &= 1 \oplus n_{t_1+302} \oplus l_{t_1+249} \\
&\quad \oplus n_{t_1+219} \cdot n_{t_1+302} \oplus l_{t_1+302} \cdot n_{t_1+219} \oplus n_{t_1+219} \\
z_{t_1+201} \oplus z_{t_2+201} &= l_{t_1+214} \oplus n_{t_1+296}, \; z_{t_1+211} \oplus z_{t_2+211} = l_{t_1+271} \oplus n_{t_1+306} \\
z_{t_1+218} \oplus z_{t_2+218} &= 1 \oplus l_{t_1+231} \oplus l_{t_1+260} \oplus n_{t_1+230} \cdot n_{t_1+313} \\
&\quad \oplus n_{t_1+230} \cdot l_{t_1+313} \oplus n_{t_1+230} \\
z_{t_1+230} \oplus z_{t_2+230} &= 1 \oplus n_{t_1+242} \oplus l_{t_1+250} \oplus l_{t_1+309} \\
z_{t_1+239} \oplus z_{t_2+239} &= 1 \oplus n_{t_1+334} \oplus l_{t_1+281} \oplus n_{t_1+251} \cdot l_{t_1+334} \\
z_{t_1+244} \oplus z_{t_2+244} &= l_{t_1+286} \oplus n_{t_1+256} \cdot n_{t_1+339} \oplus n_{t_1+256} \cdot l_{t_1+339} \oplus n_{t_1+256}
\end{aligned}$$

The base attack. We now calculate the complexity of a state recovery attack. We first consider the probability ρ that a keystream collision as in Lemma 12 that also satisfies the relations in Lemmas 13 to 15 was induced by two states that do *not* differ only in bit 127 of the LFSR. By application of the aforementioned lemmas and a similar counting exercise to that of Section 4.3, making comparable independence assumptions along the way, we have $\rho = (\frac{1}{2})^{150+3+29+5+7} \cdot (\frac{3}{4})^6 \approx 2^{-196.49}$. Then the expected number of pairs that proceed to the equation solving stage of the attack is $P_s = \binom{N}{2} \cdot \rho \approx 2^{59.51}$ where $N \approx 2^{128.5}$ as described above.

To calculate the cost of equation solving, we take the alternate approach described but not followed in Section 4.4. Recall that, after the filtering stages of the attack, we are required to essentially determine which of the approximately P_s remaining candidate states is correct. Following [BBI19], we compare the real-time cost of performing an encryption of Grain-128 with the cost of accepting/rejecting one such candidate state. Letting C_u be the cost of rejecting a candidate state and C_s the cost of accepting a candidate state in the number of Grain-128 encryptions, the cost of equation solving is therefore around $[(P_s - 1) \cdot C_u + C_s]$ Grain-128 encryptions.

To this end, we formulated the problem as a SAT problem and ran experiments using Cryptominisat and computer algebra software SAGE 9.6. More precisely, we generated a series of equations in 256 + 11 keystream bits additionally subject to the constraints from Lemmas 17 to 19. Note that in particular that using relations above with $n_{t_1+12+i} \cdot n_{t_1+95+i}$ terms allows us to linearize equations with degree 2 terms and thus speed up solving. We used a laptop running an Intel i7-8565U processor with 16GB of RAM to perform the experiments. Firstly, we estimated the time for the solver to return SAT when the attacker *correctly* assumes that a given differential keystream which satisfies filters derived from relations from Lemmas 12 to 17 is from the desired near collision state. We then estimated the time for the solver to return UNSAT when the attacker *incorrectly* assumes the above (by enforcing the constraints from Lemmas 12 to 17 on an otherwise random internal state). Finally, we estimated the amount of time required for one Grain-128 encryption, i.e. to perform 256 initialization rounds and 4 keystream clocks (one can argue that 4 keystream clocks is appropriate as in Appendix A). Our results are as follows:

- Encryption time: Encryption took an average of 1.448ms to perform, taking the average over 10000 runs.
- SAT time: To speed up the SAT solver, we guessed 55 of the NFSR state bits in each experiment (thus the cost of this portion attack increases by a factor of 2^{55}). 600 experiments yielded an average time of 16.70 seconds.
- UNSAT time: We guessed less bits of the NFSR state for UNSAT instances, namely 35 bits (inducing an increase in running time of 2^{35} per iteration); an average of 12.57 seconds was used over 600 experiments.

Attack complexity. We consider the attack cost T_{Total}, which is dominated by the two terms T_{Gen} to generate the near collision and T_{Eq} to perform equation solving and ultimately recover the internal state. During equation solving we also need to solve for each candidate LFSR state. Here, for around $2^{59.51}$ candidate states we need to perform around $2^{23.3}$ bit-operations, i.e. perform around $2^{82.81}$ bit-operations, a number that is dwarfed by the other two terms below.

For finding the near collision, we require around $2^{128.5}$ keystream bits, which costs around $2^{128.5}/(256+4) \approx 2^{120.48}$ Grain-128 encryptions. We insert elements of the form (t, Z_i, Y_i) in a hash table as before where t denotes number of elapsed clocks of the Grain keystream ($|t| = 128$), Z_i comprises 256 bits of keystream from position t and Y_i comprises 11 bits of keystream backwards from t. We

require around $2^{128.5}$ hash table insertions, and thus $2^{128.5} \cdot (128 + 256 + 11) \approx 2^{137.13}$ bits of memory. We note that since keystream bits overlap in the table and thus there is redundancy, we can simply store keystream bits in a separate ordered table and thus use $2^{128.5} \cdot (128+1)+(256+11) \approx 2^{135.51}$ bits of memory, but optimizing via sampling resistance allows for even less memory consumption.

We now consider the cost of equation solving. By properties of the LFSR and NFSR feedback polynomials, one can easily design a hardware circuit which performs 32 clocks of Grain-128 per cycle with less than 32 times overhead versus running the circuit serially [HJMM06b]. We thus divide the time that we have measured by 32 when estimating the attack complexity to conservatively estimate the amount of time required to perform the encryption. The cost is around $T_{Solve} = (P_s - 1) \cdot C_u + C_s$ Grain-128 encryptions. We deduce that $C_u \approx 2^{35} \cdot \frac{12.70}{0.001448} \cdot 32 \approx 2^{53.08}$ and $C_s = 2^{55} \cdot \frac{17.36}{0.001448} \cdot 32 \approx 2^{73.46}$. Correspondingly, equation solving costs around $T_{Solve} \approx 2^{112.59}$ Grain-128 encryptions and relatively negligible memory. Thus the overall attack cost of around $2^{120.48}$ Grain-128 encryptions is dominated by the cost of generating the collision.

Sampling resistance. To reduce memory consumption we consider sampling resistance as in Section C. It is argued in [Bjo08] that the sampling resistance of Grain-128 is at most 2^{-22} by considering the spacing between taps n_{15} and n_{36} in the output function. As before, we make use of this property and insert tuples (t, Z_t, Y_t) into our hash table if and only if Z_t is prefixed by r zero bits, where we can safely consider $r \leq 22$ for our purposes. By similar analysis to Section C.1, we require $2^{128.5+r/2}$ keystream bits, i.e. $2^{120.48+r/2}$ Grain-128 encryptions, and $2^{128.5-r/2}$ hash table insertions to find the near collision (consuming around $2^{137.1-r/2}$ bits of memory). By comparison, by the same logic as used for Grain v1, the state recovery attack that minimises the number of Grain-128 encryptions requires around $2^{124.98}$ Grain-128 encryptions, $2^{123.98}$ hash table insertions and 2^{132} bits of data and memory.

D.1 Attacking Grain-128a

The NFSR that Grain-128a [AHJM11a] uses is defined by the following (relatively non-linear and longer) update function g:

$$g(X_t) = g_{128}(X_t) + n_{t+88}n_{t+92}n_{t+93}n_{t+95} + n_{t+22}n_{t+24}n_{t+25} + n_{t+70}n_{t+78}n_{t+82}$$

where g_{128} is Grain-128's update function (c.f. the previous subsection). The output function is defined as $z_t = h'(X_t, Y_t) = \bigoplus_{a \in A} n_{t+a} + h(n_{t+12}, l_{t+8}, l_{t+13}, l_{t+20}, n_{t+95}, l_{t+42}, l_{t+60}, l_{t+79}, l_{t+94})$; it only differs from that of Grain-128 in that h's last argument is now l_{t+94} instead of l_{t+95}. We note that Grain-128a supports optional authentication given the first bit of the IV is 1; we assume in our attack hereafter that authentication is disabled (i.e. we assume $IV_0 = 0$).

Since the attack is very similar to that of Grain-128, we defer describing the algebraic relations we use to Appendix D.2. As before, we find a near collision by processing and storing around $2^{128.5}$ bits of keystream which takes around $2^{120.48}$

Grain-128a encryptions in time. Here, we use around $2^{128.5} \cdot (128 + 226 + 11) \approx 2^{137.01}$ bits of memory naively. Filtering using the relations from Lemmas 20 to 24 and making comparable arguments to before, the probability that the attacker finds a hash table collision that was not the result of the desired near collision is around $\rho = (\frac{1}{2})^{153+2+24+1+4} \cdot (\frac{3}{4})^2 \approx 2^{-184.83}$. Similarly, around $\binom{N}{2} \cdot \rho = 2^{71.17}$ candidate LFSR states remain for equation solving. As before, solving for L_{t_1} and L_{t_2} for every such state requires around $2^{71.17+23.3} = 2^{94.47}$ bit-operations which is again a relatively small cost.

We touch on experimental results for equation solving. We performed the same three classes of experiments as for Grain-128. One Grain-128a encryption took around 1.687 milliseconds, averaging over 10000 runs. As before, we guessed 55 and 35 bits of the NFSR state respectively for SAT and UNSAT instances, and took averages over 600 runs. SAT instances took on average 21.32 seconds and UNSAT instances took on average 13.22 seconds to terminate. Thus, the time to solve equations can be estimated to be around $(P_s - 1) \cdot C_u + C_s \approx 2^{124.18}$ Grain-128a encryptions, where $C_u = 2^{35} \cdot \frac{13.51}{0.001687} \cdot 32 \approx 2^{52.94}$ and $C_s = 2^{55} \cdot \frac{21.41}{0.001687} \cdot 32 \approx 2^{73.63}$. In this case, the attack cost is around $2^{124.01}$ Grain-128a encryptions. With our cost estimates, the attack is dominated by the equation solving phase and the margin is tighter than for Grain-128. We also note that the same tradeoff using sampling resistance as in Grain-128 can be applied here.

D.2 Relations for Grain-128a near collision

Lemma 20. *Consider two internal states in Grain-128a, $S_{t_1} = (N_{t_1}, L_{t_1})$ and $S_{t_2} = (N_{t_2}, L_{t_2})$ during the keystream phase such that $S_{t_1} \oplus S_{t_2} = 0^{128}||e_{127}$. Then consider*

$$Z_{t_i} = [z_{t_i+0}, z_{t_i+1}, z_{t_i+2}, \ldots, z_{t_i+225}],\ Y_{t_i} = [z_{t_i-1}, z_{t_i-2}, z_{t_i-3}, \ldots, z_{t_i-11}].$$

*for $i = 1, 2$. Then in the 237 bit difference vector $\Delta = Z_{t_1}||Y_{t_1} \oplus Z_{t_2}||Y_{t_2}$, there are **153** bits that take the value 1 or 0 with probability 1, i.e. when the probability is computed over all possible initial states S_{t_1}.*

Lemma 21. *We have $z_{t_1+99} \oplus z_{t_2+99} \oplus z_{t_1+165} \oplus z_{t_2+165} = 0$ and $z_{t_1+106} \oplus z_{t_2+106} \oplus z_{t_1+146} \oplus z_{t_2+146} = 0$ with probability 1.*

Lemma 22. *For the 24 pairs $(i, j) \in \{(48, 108), (67, 146), (80, 140), (95, 155), (99, 178), (106, 166), (107, 120), (112, 172), (125, 204), (131, 210), (138, 198), (139, 152), (142, 202), (146, 166), (157, 236), (163, 242), (164, 224), (165, 178), (169, 229), (170, 230), (171, 184), (172, 192), (203, 216), (225, 245)\}$, we have $z_{t_1+i} \oplus z_{t_2+i} = l_{t_1+j}$. We also have $z_{t_1+188} \oplus z_{t_2+188} = l_{t_1+267} \oplus 1$.*

Lemma 23. *We have the following 4 identities with probability 1.*

$$z_{t_1+114} \oplus z_{t_2+114} = l_{t_1+134} \oplus l_{t_1+193}, \quad z_{t_1+161} \oplus z_{t_2+161} = l_{t_1+181} \oplus l_{t_1+240}$$
$$z_{t_1+178} \oplus z_{t_2+178} = l_{t_1+198} \oplus l_{t_1+257}, \quad z_{t_1+189} \oplus z_{t_2+189} = l_{t_1+249} \oplus l_{t_1+268} \oplus 1$$

Lemma 24. *We have* $z_{t_1+160} \oplus z_{t_2+160} \oplus l_{t_1+202} \oplus n_{t_1+172} \cdot l_{t_1+254} = 1$ *and* $z_{t_1+197} \oplus z_{t_2+197} \oplus l_{t_1+210} \oplus l_{t_1+239} \oplus n_{t_1+209} \cdot l_{t_1+291} = 1$ *with probability 1.*

Lemma 25. *For the 8 pairs* $(i, j) \in \{(85, 180), (117, 212), (119, 131), (132, 227), (143, 238), (151, 163), (177, 189), (179, 274)$ *we have* $z_{t_1+i} \oplus z_{t_2+i} = n_{t_1+j}$*. For the 3 pairs* $(i, j) \in \{(166, 178), (175, 270), (209, 221)\}$*, we have* $z_{t_1+i} \oplus z_{t_2+i} = n_{t_1+j} \oplus 1$*.*

Lemma 26. *For the 9 values* $i \in \{33, 91, 97, 123, 127, 129, 185, 187\}$*, we have* $z_{t_1+i} \oplus z_{t_2+i} = n_{t_1+12+i} \cdot n_{t_1+95+i}$*. We also have* $z_{t_1+149} \oplus z_{t_2+149} = n_{t_1+244} \cdot (1 \oplus n_{t_1+161})$*,* $z_{t_1+181} \oplus z_{t_2+181} = n_{t_1+276} \cdot (1 \oplus n_{t_1+193})$ *and* $z_{t_1+213} \oplus z_{t_2+213} = 1 \oplus n_{t_1+225} \cdot (1 \oplus n_{t_1+308})$*.*

Lemma 27. *We have the following 9 identities with probability 1.*

$$
\begin{aligned}
z_{t_1+144} \oplus z_{t_2+144} &= l_{t_1+204} \oplus n_{t_1+239} \cdot n_{t_1+156} \\
z_{t_1+154} \oplus z_{t_2+154} &= l_{t_1+167} \oplus n_{t_1+166} \cdot n_{t_1+249} \\
z_{t_1+159} \oplus z_{t_2+159} &= l_{t_1+219} \oplus n_{t_1+171} \cdot n_{t_1+254} \\
z_{t_1+174} \oplus z_{t_2+174} &= l_{t_1+234} \oplus n_{t_1+186} \cdot n_{t_1+269} \\
z_{t_1+183} \oplus z_{t_2+183} &= l_{t_1+262} \oplus n_{t_1+195} \\
z_{t_1+207} \oplus z_{t_2+207} &= 1 \oplus n_{t_1+302} \oplus l_{t_1+249} \oplus n_{t_1+219} \cdot n_{t_1+302} \\
&\quad \oplus l_{t_1+301} \cdot n_{t_1+219} \oplus n_{t_1+219} \\
z_{t_1+208} \oplus z_{t_2+208} &= l_{t_1+228} \oplus l_{t_1+268} \oplus l_{t_1+287} \oplus n_{t_1+220} \cdot n_{t_1+303} \\
z_{t_1+211} \oplus z_{t_2+211} &= n_{t_1+306} \oplus l_{t_1+271} \\
z_{t_1+219} \oplus z_{t_2+219} &= 1 \oplus l_{t_1+298} \oplus n_{t_1+231} \cdot n_{t_1+314}
\end{aligned}
$$