Recovering the Key from the Internal State of Grain-128AEAD

Donghoon Chang and Meltem Sönmez Turan

National Institute of Standards and Technology, Gaithersburg, Maryland, USA

Abstract. Grain-128AEAD is one of the second-round candidates of the NIST lightweight cryptography standardization process. There is an existing body of third-party analysis on the earlier versions of the Grain family that provide insights on the security of Grain-128AEAD. Different from the earlier versions, Grain-128AEAD reintroduces the key into the internal state during the initialization. The designers claim that internal state recovery no longer results in key recovery, due to this change. In this paper, we analyze this claim under different scenarios.

Keywords: Grain-128AEAD; lightweight cryptography; key recovery

1 Introduction

Grain-128AEAD [1] is one of the second-round candidates of the National Institute of Standards and Technology (NIST) lightweight cryptography standardization process. To avoid key recovery from the internal state, Grain-128AEAD reintroduces the key into the internal state during the initialization.

In this paper, we analyze the effectiveness of the reintroduction of the key to the state under three different scenarios. In the first scenario, we present a practical method to recover the key from the knowledge of the full internal state, including Nonlinear Feedback Shift Register (NFSR), Linear Feedback Shift Register (LFSR), accumulator and the register, when the state is recovered right after initialization. In the second scenario, we present a key-recovery attack when the state is recovered during encryption, and the states of the accumulation and the register are not available to the attacker. In the last sceneario, we provide the attack in nonce-misuse setting.

2 Key and Nonce Initialization

According to [1], the internal state of Grain consists of two registers: a 128-bit NFSR and a 128-bit LFSR. However, for tag generation, an additional 64-bit accumulator and a 64-bit register are required as shown in Figure 1. We call the combination of NFSR, LFSR, accumulator and the register states as the *full* internal state of Grain-128AEAD.

Let (B_t, S_t, A_t, R_t) [1] be the full internal state of Grain-128AEAD at time t, where

- $B_t = [b_t, b_{t+1}, \dots, b_{t+127}]$ is the NFSR state at time $t \geq 0$,
- $S_t = [s_t, s_{t+1}, \dots, s_{t+127}]$ is the LFSR state at time $t \geq 0$,
- $A_t = [a_0^t, a_1^t, \dots, a_{63}^t]$ is the accumulator state at time $t > 383$,
- $R_t = [r_0^t, r_1^t, \dots, r_{63}^t]$ is the register state at time $t > 383$.

First, the key and the nonce (IV) are loaded to the state, i.e.,

$$B_0 = [b_0, b_1, \dots, b_{127}] \leftarrow [k_0, k_1, \dots, k_{127}]$$
$$S_0 = [s_0, s_1, \dots, s_{127}] \leftarrow [IV_0, \dots, IV_{95}, 1, 1, \dots, 1, 0]$$

Next, the state is clocked 256 times, with feed-forwarding the output of the cipher to the NFSR and the LFSR. The nonlinear Boolean functions used during the initialization are defined as:

$$\begin{aligned} h(x_0, \dots, x_8) =& x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8 \\ g(x_0, \dots, x_{28}) =& x_0 + x_1 + x_2 + x_3 + x_4 + x_5x_6 + x_7x_8 + x_9x_{10} + x_{11}x_{12} \\ & + x_{13}x_{14} + x_{15}x_{16} + x_{17}x_{18} + x_{19}x_{20}x_{21} + x_{22}x_{23}x_{24} \\ & + x_{25}x_{26}x_{27}x_{28} \end{aligned}$$

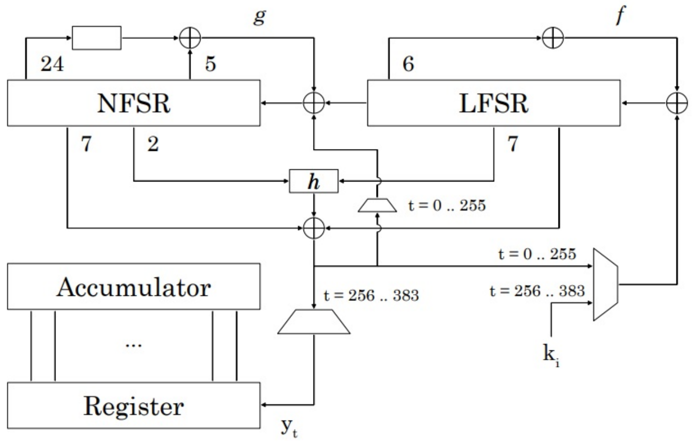

Fig. 1. Overview of the Initialization of Grain-128AEAD

This part of the initialization is invertible. In the second part of the initialization (i.e., $256 \leq t \leq 383$), the accumulator and the register are initialized, and the key is reintroduced back to the internal state. Algorithm 1 describes the initialization of the cipher.

[1] For our purposes, we use a slightly different notation from [1].

Algorithm 1 Initialization (K,IV)

```
1: $[b_0, b_1, \ldots, b_{127}] \leftarrow [k_0, k_1, \ldots, k_{127}]$
2: $[s_0, s_1, \ldots, s_{127}] \leftarrow [IV_0, \ldots, IV_{95}, 1, 1, \ldots, 1, 0]$
3: for $t = 0, 1, \ldots, 383$ do
4:     $y_t = h(b_{t+12}, s_{t+8}, s_{t+13}, s_{t+20}, b_{t+95}, s_{t+42}, s_{t+60}, s_{t+79}, s_{t+94}) + s_{t+93} + b_{t+2} + b_{t+15} + b_{t+36} + b_{t+45} + b_{t+64} + b_{t+73} + b_{t+89}$
5:     $b_{t+128} = s_t + b_t + b_{t+26} + b_{t+56} + b_{t+91} + b_{t+96} + b_{t+3}b_{t+67} + b_{t+11}b_{t+13} + b_{t+17}b_{t+18} + b_{t+27}b_{t+59} + b_{t+40}b_{t+48} + b_{t+61}b_{t+65} + b_{t+68}b_{t+84} + b_{t+22}b_{t+24}b_{t+25} + b_{t+70}b_{t+78}b_{t+82} + b_{t+88}b_{t+92}b_{t+93}b_{t+95} + y_t$
6:     $s_{t+128} = s_t + s_{t+7} + s_{t+38} + s_{t+70} + s_{t+81} + s_{t+96} + y_t$
7:     if $t > 255$ then
8:         $s_{t+128} = s_{t+128} + y_t + k_{t-256}$
9:         $b_{t+128} = b_{t+128} + y_t$
10:    end if
11: end for
12: $B_{384} \leftarrow [b_{384}, \ldots, b_{511}]$
13: $S_{384} \leftarrow [s_{384}, \ldots, s_{511}]$
14: $A_{384} \leftarrow [y_{256}, y_{257}, \ldots, y_{319}]$
15: $R_{384} \leftarrow [y_{320}, y_{321}, \ldots, y_{383}]$
```

3 Recovering Key from the Internal State

Suppose that the attacker has access to message M and the corresponding ciphertext C and the tag T, encrypted under the secret key K. Let L be the length of the padded message. First, in Section 3.1, we present a practical method to recover the K, from the full internal state right after initialization. This method does not require the knowledge of the message and the ciphertext. Next, in Section 3.2. we present a method to recover the key only using the LFSR and NFSR states.

3.1 Using full internal state at t=384

Let's assume that the attacker recovers the state of the NFSR, LFSR, accumulator and the register, right after initialization at time $t = 384$. Here, the aim is to recover the key from $(B_{384}, S_{384}, A_{384}, R_{384})$. As shown in Figure 2, there are two unknown bits represented by ?, namely b_{383} and s_{383} and the last bit of the register is undefined and represented by $*$, when the cipher is clocked backwards from $t = 384$ to $t = 383$.

t=384		t=383
$B_{384} = (b_{384}, \dots, b_{511})$	$\rightarrow$	$B_{383} = (?, b_{384}, \dots, b_{510})$
$S_{384} = (s_{384}, \dots, s_{511})$	$\rightarrow$	$S_{383} = (?, s_{384}, \dots, s_{510})$
$A_{384} = (y_{256}, \dots, y_{319})$	$\rightarrow$	$A_{383} = (y_{256}, \dots, y_{319})$
$R_{384} = (y_{320}, \dots, y_{383})$	$\rightarrow$	$R_{383} = (*, y_{320}, \dots, y_{382})$

Fig. 2. Inverting the internal state from t=384 to t=383

Using the following equations that compute y_{381} and b_{511}, it is possible to recover the two missing state bits b_{383} and s_{383}.

$$\begin{aligned} y_{381} =& h(b_{393}, s_{389}, s_{394}, s_{401}, b_{476}, s_{423}, s_{441}, s_{460}, s_{475}) + s_{474} + \mathbf{b_{383}} + b_{396} \\ &+ b_{417} + b_{426} + b_{445} + b_{454} + b_{470} \\ b_{511} =& \mathbf{s_{383}} + \mathbf{b_{383}} + b_{409} + b_{439} + b_{474} + b_{479} + b_{386}b_{450} + b_{394}b_{396} + b_{400}b_{401} \\ &+ b_{410}b_{442} + b_{423}b_{431} + b_{444}b_{448} + b_{451}b_{467} + b_{405}b_{407}b_{408} \\ &+ b_{453}b_{461}b_{465} + b_{471}b_{475}b_{476}b_{478} \end{aligned}$$

Next, the key bit k_{127} is calculated as using the following equation (obtained from the equation on line (6) of Algorithm 1, evaluated at t=383:

$$\mathbf{k_{127}} = s_{383} + s_{390} + s_{421} + s_{453} + s_{464} + s_{479} + s_{511}.$$

After recovering the internal state at time $t = 383$, the attacker repeats a similar procedure and recovers k_i using b_{i+384}, s_{i+384}, and y_{i+254}, for $2 \leq i \leq 127$. The attacker guesses the remaining key bits k_0 and k_1, and inverts the cipher to $t = 0$ using the guess, and checks the correctness of the guess by comparing it to b_0 and b_1.

3.2 Using LFSR and NFSR states at $t = 384 + \Delta$

In this case, we assume that the attacker recovers the LFSR and NFSR states at time $t = 384 + \Delta$. Since LFSR and NFSR states are invertible during encryption, the attacker can recover the LFSR and NFSR states at time $t = 384$ (right after initialization). However, the attacker still needs the states of the accumulator and the register to apply the method presented in Section 3.1. In this case, the attacker uses a known message M, and the corresponding ciphertext C and the tag T pair. Let $m_0, m_1, \dots, m_{L-1}$ be the padded message.

First, the attacker guesses the state of the 64-bit register at time $t = 384$, namely $[y_{320}, y_{321}, \dots, y_{383}]$. Given the LFSR, NFSR and the register states at $t = 384$, the attacker can now clock the cipher forward and obtain the LFSR, NFSR and register states at any time $384 < t \leq 384 + 2L - 1$. Since the tag T provides the state of the accumulator at time $t = 384 + 2L - 1$ (after the message processing), it is possible to recover the state of the accumulator at time $t = 384$, using the following equation:

$$A_{384} = T + \sum_{i=0}^{L-1} m_i \cdot R_{2i+384}. \tag{1}$$

For each guess of the register state, the attacker can recover a key candidate (using the methods from Section 3.1), and check the correctness of the key by comparing it to the initial state of the NFSR. The time complexity of this step is 2^{64}.

In this note, we did not attempt to optimize the complexity of guessing the register state $[y_{320}, y_{321}, \ldots, y_{383}]$, however, it can be improved. For example, the attacker can determine the following two bits of the register y_{383} and y_{382}, by using the available LFSR and the NFSR bits, i.e.,

$$\begin{aligned} y_{383} =& h(b_{395}, s_{391}, s_{396}, s_{403}, b_{478}, s_{425}, s_{443}, s_{462}, s_{477}) \\ &+ s_{476} + b_{385} + b_{398} + b_{419} + b_{428} + b_{447} + b_{456} + b_{472}, \\ y_{382} =& h(b_{394}, s_{390}, s_{395}, s_{402}, b_{477}, s_{424}, s_{442}, s_{461}, s_{476}) \\ &+ s_{475} + b_{384} + b_{397} + b_{418} + b_{427} + b_{446} + b_{455} + b_{471}. \end{aligned}$$

Hence, the time complexity of guessing the register state is limited by 2^{62}.

3.3 Using LFSR and NFSR states at $t = 384 + \Delta$ in Nonce-misuse

In this section, we present an efficient key recovery attack from the internal state in a nonce-misuse setting. The attack requires two queries and has negligible time complexity.

In Grain-128AEAD, the associated data ad and message M is padded as

$$pad(ad, M) = Encode(adlen)||ad||M||\texttt{0x80},$$

where $Encode() = y$ is defined as follows if the first byte in y starts with a 0, the remaining 7 bits is an encoding of the number of bytes in the associated data (up to 127 bytes). If the first byte in y starts with a 1, the remaining 7 bits are instead an encoding of the number of forthcoming bytes that are used to describe the length (in bytes) of the associated data.

Let $\overline{x}$:= be the bitwise complement of x, $x[i]$ be the i-th byte of x, and $x[i-j] := x[i]|| \ldots ||x[j]$, and $x[i-] := x[i]|| \ldots ||last\ byte\ of\ x$.

Firstly, the attacker chooses the two pairs of associate data and message, (ad, M) and (ad', M'), where

- $ad = ad_1||ad_2$, where ad_1 is any 6-byte string and ad_2 is any 122-byte string,
- M is any byte string,
- $ad' = ad'_1||ad'_2$, where ad'_1 is 7-byte string and ad'_2 is 119-byte string such that the first byte of ad'_1, i.e. $ad'_1[1]$, is '01111111' and $ad'_1[2-7] = \overline{ad_1}$, $ad'_2 = ad_2[1-119]$.

– $M' = M'_1 || M'_2$, where $M'_1 = ad_2[120-122]$ and $M'_2 = M$.

Note that the padded inputs satisfy the following relation:

$$\begin{gathered} pad(ad, M)[1-8] = \overline{pad(ad', M')[1-8]} \\ pad(ad, M)[9-] = pad(ad', M')[9-]. \end{gathered}$$

Let us assume that the attacker obtains the corresponding ciphertext-tag pairs (C, T) and (C', T') with a same nonce IV in the nonce-misuse setting and recovers LFSR and NFSR states at time $t = 384 + \Delta$. Note that A_{384} and R_{384} are the same for both cases when the nonce is repeated.

The attacker constructs the following two equations (2) and (3), where L is the bit-length of $pad(ad, M)$ and $pad(ad', M')$. Let $pad(ad, M) = (m_0, m_1, \ldots, m_{L-1})$ and $pad(ad', M') = (m'_0, m'_1, \ldots, m'_{L-1})$.

$$T = A_{384} + \sum_{i=0}^{L-1} m_i \cdot R_{2i+384}. \tag{2}$$

$$T' = A_{384} + \sum_{i=0}^{L-1} m'_i \cdot R_{2i+384}. \tag{3}$$

Due to the relations on the two padded inputs, by adding the equations (2) and (3), the attacker can obtain the following equation (4):

$$T + T' = \sum_{i=0}^{63} R_{2i+384}. \tag{4}$$

Since

$$R_{384} = \begin{bmatrix} y_{320} \\ y_{321} \\ y_{322} \\ \vdots \\ y_{383} \end{bmatrix}, R_{386} = \begin{bmatrix} y_{321} \\ y_{322} \\ \vdots \\ y_{383} \\ y_{385} \end{bmatrix}, \ldots, R_{510} = \begin{bmatrix} y_{383} \\ y_{385} \\ y_{387} \\ \vdots \\ y_{509} \end{bmatrix}, \tag{5}$$

we have

$$\begin{bmatrix} 1\ 1\ 1 \ldots 1 \\ 0\ 1\ 1 \ldots 1 \\ 0\ 0\ 1 \ldots 1 \\ \vdots\ \vdots\ \vdots \ddots \vdots \\ 0\ 0\ 0 \ldots 1 \end{bmatrix} \begin{bmatrix} y_{320} \\ y_{321} \\ y_{322} \\ \vdots \\ y_{383} \end{bmatrix} = T + T' + \begin{bmatrix} 0 \\ y_{385} \\ y_{385} + y_{387} \\ \vdots \\ \sum_{i=0}^{62} y_{385+2i} \end{bmatrix}. \tag{6}$$

It is easy to check that the first matrix of the left side is invertible. Since the attacker is assumed to recover LFSR and NFSR states at time $t = 384 + \Delta$, as explained in Section 3.2, the attacker can also recover the LFSR and NFSR states at time $t = 384$. Then $(y_{385}, y_{387}, \ldots, y_{509})$ is determined. Therefore, $R_{384} = [y_{320}, y_{321}, \ldots, y_{383}]$ is recovered from the equation (6). Then, from the equation (2), $A_{384} = [y_{256}, y_{257}, \ldots, y_{319}]$ is also recovered. Finally, the attacker performs the key recovery attack described in Section 3.1.

4 Conclusion

In this paper, we analyzed the effectiveness of reintroducing the key to the internal state to avoid key recovery from the knowledge of the state. We observed that the secrecy of the key is not fully protected by reintroducing the key in the three different scenarios considered in this paper. In these scenarios, adding more rounds before or after reintroducing the key does not invalidate the presented techniques, as these parts are invertible. In general, updating the state by inserting a single key bit in each clock may not be a successful strategy. Alternatively, the 128-bit key can be added back to the NFSR state right after initialization to avoid key recovery when the state is disclosed.

Note that this paper assumes the knowledge of the internal state, and state recovery techniques are not within the scope of the paper.

Acknowledgments

The authors thank the Grain-128AEAD designers for comments and suggestions; and Kerry McKay and Jeffrey Marron from NIST for editorial comments.

References

1. M. Hell, T. Johansson, W. Meier, J. Sönnerup, and H. Yoshida. Grain-128AEAD - a lightweight AEAD stream cipher. Submission to the NIST Lightweight Cryptography Standardization Process, 2019. https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.