Application of the new modeling technique on Grain-128AEAD

No Author Given

No Institute Given

1 Specification of Grain-128AEAD

Grain-128AEAD [4] is a member of the Grain family and also one of the ten finalist candidates of the NIST LWC standardization process. Grain-128AEAD inherits many specifications from Grain-128a, proposed in 2011 [1]. Hereinafter, we follow the assumption in [2,3] that the first bit of the pre-output key stream can be observed, so there is no difference between Grain-128a and Grain-128AEAD under this assumption.

The internal state of Grain-128AEAD is represented by two 128-bit states, $(b_0, b_1, \ldots, b_{127})$ and $(s_0, s_1, \ldots, s_{127})$. The 128-bit key is loaded to the first register $\boldsymbol{b}$, and the 96-bit initialization vector is loaded to the second register $\boldsymbol{s}$. The other state bits are set to 1 except the least one bit in the second register. Namely, the initial state bits are represented as

$$(b_0, b_1, \ldots, b_{127}) \leftarrow (k_0, k_1, \ldots, k_{127}),$$
$$(s_0, s_1, \ldots, s_{127}) \leftarrow (v_0, v_1, \ldots, v_{95}, 1, \ldots, 1, 0).$$

The pseudo-code of the update function in the initialization is given as follows.

$$\begin{aligned}
g &\leftarrow b_0 \oplus b_{26} \oplus b_{56} \oplus b_{91} \oplus b_{96} \oplus b_3 b_{67} \oplus b_{11} b_{13} \oplus b_{17} b_{18} \oplus b_{27} b_{59} \oplus b_{40} b_{48} \\
&\quad \oplus b_{61} b_{65} \oplus b_{68} b_{84} \oplus b_{88} b_{92} b_{93} b_{95} \oplus b_{22} b_{24} b_{25} \oplus b_{70} b_{78} b_{82}, \\
f &\leftarrow s_0 \oplus s_7 \oplus s_{38} \oplus s_{70} \oplus s_{81} \oplus s_{96}, \\
h &\leftarrow b_{12} s_8 \oplus s_{13} s_{20} \oplus b_{95} s_{42} \oplus s_{60} s_{79} \oplus b_{12} b_{95} s_{94}, \\
z &\leftarrow h \oplus s_{93} \oplus b_2 \oplus b_{15} \oplus b_{36} \oplus b_{45} \oplus b_{64} \oplus b_{73} \oplus b_{89}, \\
&\quad (b_0, b_1, \ldots, b_{127}) \leftarrow (b_1, b_2, \ldots, b_{127}, g \oplus s_0 \oplus z), \\
&\quad (s_0, s_1, \ldots, s_{127}) \leftarrow (s_1, s_2, \ldots, s_{127}, f \oplus z).
\end{aligned}$$

In the initialization, the state is updated 256 times without producing an output. After the initialization, the update function is tweaked such that z is not fed to the state, and z is used as a pre-output key stream. Hereinafter, we assume that the first bit of the pre-output key stream can be observed. We study the variant of Grain-128AEAD, whose initialization phase is reduced to r rounds.

2 The propagation rules of 3SDP/u for basic Boolean functions

2.1 The propagation rules for three basic functions.

Proposition 1 (COPY). *Let* $\boldsymbol{x} = (x_0, x_1, ..., x_{m-1})$ *and* $\boldsymbol{y} = (x_0, x_0, x_1, ..., x_{m-1})$ *be the input and output vector of a Copy function. Let* $\mathbb{X}$ *and* $\mathbb{Y}$ *be the input and output multisets, respectively. Assuming that* $\mathbb{X}$ *has* $\mathcal{T}_{\mathbb{L}}^{1^m}$, $\mathbb{Y}$ *has* $\mathcal{T}_{\mathbb{L}'}^{1^{m+1}}$ *where* $\mathbb{L}'$ *is computed as*

$$\mathbb{L}' \leftarrow (l_0', l_0'', l_1, \ldots, l_{m-1}), \text{ if } l_0' \vee l_0'' = l_0,$$

for all $\boldsymbol{l} \in \mathbb{L}$. *Here* $\mathbb{L}' \leftarrow \boldsymbol{l}$ *denotes that* $\boldsymbol{l}$ *is inserted into the multiset* $\mathbb{L}'$.

Proposition 2 (AND). *Let* $\boldsymbol{x} = (x_1, x_2, ..., x_m)$ *and* $\boldsymbol{y} = (x_1 \cdot x_2, x_3, ..., x_m)$ *be the input and output vector of an And function. Let* $\mathbb{X}$ *and* $\mathbb{Y}$ *be the input and output multisets, respectively. Assuming that* $\mathbb{X}$ *has* $\mathcal{T}_{\mathbb{L}}^{1^m}$, $\mathbb{Y}$ *has* $\mathcal{T}_{\mathbb{L}'}^{1^{m-1}}$ *where* $\mathbb{L}'$ *is computed as*

$$\mathbb{L}' \leftarrow (l_0, l_2, \ldots, l_{m-1}), \text{ if } l_0 = l_1,$$

for all $\boldsymbol{l} \in \mathbb{L}$.

Proposition 3 (XOR). *Let* $\boldsymbol{x} = (x_1, x_2, ..., x_m)$ *and* $\boldsymbol{y} = (x_1 \oplus x_2, x_3, ..., x_m)$ *be the input and output vector of a Xor function. Let* $\mathbb{X}$ *and* $\mathbb{Y}$ *be the input and output multisets, respectively. Assuming that* $\mathbb{X}$ *has* $\mathcal{T}_{\mathbb{L}}^{1^m}$, $\mathbb{Y}$ *has* $\mathcal{T}_{\mathbb{L}'}^{1^{m-1}}$ *where* $\mathbb{L}'$ *is computed as*

$$\mathbb{L}' \leftarrow (l_0 + l_1, l_2, \ldots, l_{m-1}), \text{ if } l_0 \cdot l_1 = 0,$$

for all $\boldsymbol{l} \in \mathbb{L}$.

2.2 MILP models for three basic functions.

We introduce the MILP models for the three basic functions COPY, AND, and XOR. Note that these models are first proposed in [2,3], and describe the generalized forms of their propagation rules, which are easily deduced from their original forms. When constructing the MILP model, we usually initialize an empty model $\mathcal{M}$, then generate some MILP variables $v_1, \ldots, v_m$ by $\mathcal{M}.var \leftarrow v_1, \ldots, v_m$, which are used in the inequalities. Next, the inequality is added to the model $\mathcal{M}.con \leftarrow v_2 + \cdots + v_m \geq v_1$. Finally, we can call an off-the-shelf MILP solver to solve the model and obtain the results, including feasible or infeasible. In this paper, the MILP tool we used is the Gurobi optimizer.

Proposition 4 (MILP model for COPY). *Let a* $\xrightarrow{COPY}$ $(b_0, b_1, \ldots, b_{m-1})$ *be a three-subset division trail of* ***COPY****. The following inequalities are sufficient to describe the propagation of the modified three-subset division property for* ***COPY****.*

$$\begin{cases} \mathcal{M}.var \leftarrow a, b_0, \ldots, b_{m-1} \text{ as binary;} \\ \mathcal{M}.con \leftarrow a = b_0 \vee b_1 \vee \cdots \vee b_{m-1}. \end{cases}$$

Note that the Gurobi optimizer supports the Or $(\vee)$ *operation.*

Proposition 5 (MILP model for AND). *Let* $(a_0, a_1, \ldots, a_{m-1}) \xrightarrow{AND} b$ *be a three-subset division trail of* ***AND****. The following inequalities are sufficient to describe the propagation of the modified three-subset division property for* ***AND****.*

$$\begin{cases} \mathcal{M}.var \leftarrow a_0, \ldots, a_{m-1}, b \text{ as binary;} \\ \mathcal{M}.con \leftarrow b = a_i, \forall i \in \{0, 1, \ldots, m-1\}. \end{cases}$$

Proposition 6 (MILP model for XOR). *Let* $(a_0, a_1, \ldots, a_{m-1}) \xrightarrow{XOR} b$ *be a three-subset division trail of* ***XOR****. The following inequalities are sufficient to describe the propagation of the modified three-subset division property for* ***XOR****.*

$$\begin{cases} \mathcal{M}.var \leftarrow a_0, \ldots, a_{m-1}, b \text{ as binary;} \\ \mathcal{M}.con \leftarrow b = a_0 + a_1 + \cdots + a_{m-1}. \end{cases}$$

3 The new MILP model for Grain-128AEAD

Similarly, r-round Grain-128AEAD $G(\boldsymbol{k}, \boldsymbol{v})$ can also be decomposed into the expression as follows,

$$G(\boldsymbol{k}, \boldsymbol{v}) = F_o \circ \boldsymbol{F}_r \circ \cdots \circ \boldsymbol{F}_1 \circ \boldsymbol{F}_0(\boldsymbol{k}, \boldsymbol{v}), \tag{1}$$

where $\boldsymbol{k} \in \mathbb{F}_2^{128}$, $\boldsymbol{v} \in \mathbb{F}_2^{96}$ and $\boldsymbol{F}_i : \mathbb{F}_2^{256} \to \mathbb{F}_2^{256}$ for all $i \in \{1, \ldots, r\}$. Let $(\boldsymbol{s}_i, \boldsymbol{b}_i) = \boldsymbol{F}_i(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1})$ for $i \in \{1, \ldots, r\}$. Let $(\boldsymbol{s}_0, \boldsymbol{b}_0) = \boldsymbol{F}_0(\boldsymbol{k}, \boldsymbol{v})$. In any round i, only two state bits, $s_{i,127}$ and $b_{i,127}$, are updated respectively by $f \oplus z$ and $g \oplus s_{i-1,0} \oplus z$, and the rest of the state bits are shifted. The additional variables f, g and $z (\in \mathbb{F}_2)$ can be represented as the polynomials of $\boldsymbol{s}_{i-1}$ and $\boldsymbol{b}_{i-1}$.

3.1 Some skills in the traditional modeling method

The traditional modeling method for Grain-128AEAD is to model round by round according to Eq.(1). In the process of modeling $\boldsymbol{F}_i$, the equations $(\boldsymbol{s}_i, \boldsymbol{b}_i) = \boldsymbol{F}_i(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1})$ is not built into the model directly, because a large number of redundant three-subset division trails will be included in this model. For example, if there is a three-subset division trail satisfying the state bits $s_{i,127}$ and $b_{i,127}$ involved in the same monomial, then we have the following equation,

$$\begin{aligned} s_{i,127}b_{i,127} = \;& (f(\boldsymbol{s}_{i-1}) + z(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1})) \cdot (g(\boldsymbol{b}_{i-1}) + s_{i-1,0} + z(\boldsymbol{s}_{i-1})) \\ = \;& f(\boldsymbol{s}_{i-1}) \cdot (g(\boldsymbol{b}_{i-1}) + s_{i-1,0}) + z^2(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}) \\ + \;& (f(\boldsymbol{s}_{i-1}) + g(\boldsymbol{b}_{i-1}) + s_{i-1,0}) \cdot z(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}) \qquad (\in \bar{\mathbb{Z}}[\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}]) \end{aligned}$$

It is easy to check that the polynomial $z^2(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}) \in \bar{\mathbb{Z}}[\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}]$ include many monomials whose coefficients are even; thus there are many redundant trails included in this model. Therefore, to solve this problem, additional binary variables f, g, and z are first defined, and the algebraic relation between them and $\boldsymbol{s}_{i-1}$ as well as $\boldsymbol{b}_{i-1}$ are added to the model as constraints. Then, the expressions of $\boldsymbol{s}_i$ and $\boldsymbol{b}_i$ with respect to $\boldsymbol{s}_{i-1}$, $\boldsymbol{b}_{i-1}$, f, g and z are added to the model as constraints.

However, it does not completely avoid redundant three-subset division trails, as $z \oplus s_{i-1,0}$ is involved in both two update functions of $s_{i,127}$ and $b_{i,127}$, which makes the following equations hold,

$$\begin{aligned} s_{i,127}b_{i,127} = &\ (f+z)\cdot(g+s_{i-1,0}+z) && (\in \bar{\mathbb{Z}}[\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}, f, g, z]) \\ = &\ f\cdot(g+s_{i-1,0}) + (f+g+s_{i-1,0})\cdot z + z && (\in \bar{\mathbb{Z}}[\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}, f, g, z]) \\ = &\ f(\boldsymbol{s}_{i-1})\cdot(g(\boldsymbol{b}_{i-1}) + s_{i-1,0}) \\ + &\ 2s_{i-1,0}\cdot z(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}) \\ + &\ (f'(\boldsymbol{s}_{i-1}) + g(\boldsymbol{b}_{i-1}) + 1)\cdot z(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}) && (\in \bar{\mathbb{Z}}[\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}]), \end{aligned}$$

where $f(\boldsymbol{s}_{i-1}) = f'(\boldsymbol{s}_{i-1}) \oplus s_{i-1,0}$. Thus, for each round, an additional condition is added to the traditional model: either $s_{i-1,0}$ or z must be 0. See Property 1 in [2,3] for details.

Different from the traditional modeling technique, we solve this problem by treating $z \oplus s_{i-1,0}$ as a whole and denote $z(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}) \oplus s_{i-1,0}$ by z'. Thus, the update functions can be rewritten as $f' \oplus z'$ and $g \oplus z'$, and the following equations can be obtained,

$$\begin{aligned} s_{i,127}b_{i,127} = &\ (f'+z')\cdot(g+z') = f'g + (f'+g+1)\cdot z' && (\in \bar{\mathbb{Z}}[\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}, f', g, z']) \\ = &\ (f'(\boldsymbol{s}_{i-1}) + g(\boldsymbol{b}_{i-1}) + 1)\cdot z'(\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}) \\ + &\ f'(\boldsymbol{s}_{i-1})\cdot g(\boldsymbol{b}_{i-1}) && (\in \bar{\mathbb{Z}}[\boldsymbol{s}_{i-1}, \boldsymbol{b}_{i-1}]). \end{aligned}$$

Based on the abovementioned skills, a traditional model without redundant three-subset division trails has been built. Given the practicality of these skills, we continue to adopt them in new modeling techniques to avoid the appearance of redundant trails.

3.2 The new MILP model $\mathcal{M}_{32}$

First, we set d to 32 by the heuristic method. Thus, for $0 \le i < 32$, $f'(\boldsymbol{s}_i)$, $g(\boldsymbol{b}_i)$ and $z'(\boldsymbol{s}_i, \boldsymbol{b}_i)$ can be directly represented as the polynomials of $\boldsymbol{k}$ and $\boldsymbol{v}$, denoted by $f'_i(\boldsymbol{k}, \boldsymbol{v})$, $g_i(\boldsymbol{k}, \boldsymbol{v})$ and $z'(\boldsymbol{k}, \boldsymbol{v})$ respectively. Further, the composite expression derived from Eq.(**??**) can be obtained as follow,

$$G(\boldsymbol{k}, \boldsymbol{v}) = F_o \circ \boldsymbol{F}_r \circ \cdots \circ \boldsymbol{F}_{33} \circ \boldsymbol{E}_{32}(\boldsymbol{k}, \boldsymbol{v}), \tag{2}$$

where $\boldsymbol{E}_{32} = \boldsymbol{F}_{32} \circ \cdots \circ \boldsymbol{F}_1 \circ \boldsymbol{F}_0$, which can be expressed as $\boldsymbol{E}_{32} = (s_{0,32}, \ldots, s_{0,127}, f'_0 \oplus z'_0, \ldots, f'_{31} \oplus z'_{31}, b_{0,32}, \ldots, b_{0,127}, g_0 \oplus z'_0, \ldots, g_{31} \oplus z'_{31})$.

To achieve the goal of reducing the number of feasible solutions in the new model, we replace the specific types of expressions involved in $\boldsymbol{E}_{32}(\boldsymbol{k}, \boldsymbol{v})$ by new variables. For $i \in \{0, \ldots, 31\}$, the functions $f'_i(\boldsymbol{k}, \boldsymbol{v})$, $g_i(\boldsymbol{k}, \boldsymbol{v})$, and $z'_i(\boldsymbol{k}, \boldsymbol{v})$ can be uniquely decomposed into the form $\varphi(\boldsymbol{k}, \boldsymbol{v}) \oplus \psi(\boldsymbol{k})$, where $\varphi(\boldsymbol{k}, \boldsymbol{v})$ and $\psi(\boldsymbol{k})$ are allowed to be 0-constant polynomial, and for any $\boldsymbol{\mu} \in \mathbb{F}_2^{128}$, the monomial $\pi_{\boldsymbol{\mu}}(\boldsymbol{k})$ is not involved in $\varphi(\boldsymbol{k}, \boldsymbol{v})$. Then, we replace the different expressions $\psi(\boldsymbol{k})$ by the new secret key variables $k_{128}, \ldots, k_{\alpha_1 - 1}$. Let $\boldsymbol{k}^{(1)} = (k_0, \ldots, k_{127}, k_{128}, \ldots, k_{\alpha_1 - 1}) \in$

$\mathbb{F}_2^{\alpha_1}$. Thus, the functions can be represented as the polynomials of $\boldsymbol{k}^{(1)}$ and $\boldsymbol{v}$, and according to their ANFs with respect to $\boldsymbol{k}^{(1)}$ and $\boldsymbol{v}$, they are mainly divided into two cases as follows,

Case 1: k_i or v_i or c,
Case 2: $\varphi(\boldsymbol{k}^{(1)}, \boldsymbol{v})$ or $\varphi(\boldsymbol{k}^{(1)}, \boldsymbol{v}) \oplus k_j$,

where c means a constant value. We further process the expressions belonging to Case 2 by replacing them with new IV variables $\hat{\boldsymbol{v}} = (\hat{v}_0, \ldots, \hat{v}_{\beta_1-1}) \in \mathbb{F}_2^{\beta_1-1}$, which are regarded as the intermediate temporary variables. Consequently, $(\boldsymbol{s}_{32}, \boldsymbol{b}_{32})$ can be regarded as the polynomials of $\boldsymbol{k}^{(1)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}$, denoted by $(\boldsymbol{s}_{32}, \boldsymbol{b}_{32}) = \boldsymbol{E}'(\boldsymbol{k}, \boldsymbol{v}, \hat{\boldsymbol{v}})$. Meanwhile, $G(\boldsymbol{k}, \boldsymbol{v})$ can be rewritten as

$$G(\boldsymbol{k}, \boldsymbol{v}) = F_o \circ \boldsymbol{F}_r \circ \cdots \circ \boldsymbol{F}_{33} \circ \boldsymbol{E}'_{32}(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}}). \tag{3}$$

According to Eq.(3), a new MILP model $\mathcal{M}_{32}$ of Grain-128AEAD can be built. When building the model $\mathcal{M}_{32}$, we first declare the variables $\boldsymbol{k}^{(1)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}$, and then establish the equations for the algebraic relations between $\hat{\boldsymbol{v}}$ and $\boldsymbol{k}^{(1)}$ as well as $\boldsymbol{v}$, which need to be added to the model $\mathcal{M}_{32}$. Next, the algebraic relations $(\boldsymbol{s}_{32}, \boldsymbol{b}_{32}) = \boldsymbol{E}'_{32}(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}})$ are added to the model as restrictions. Finally, a complete model $\mathcal{M}_{32}$ is established by iterative method according to Eq.(3). The details of the new MILP model are present in Algorithms 1, 2, 3, and 4.

Expression Construction. GRAIN in Algorithm 1 is used to compute the ANFs of the state bits with respect to $\boldsymbol{k}$ and $\boldsymbol{v}$. It is worth noting that we use the modified update function to calculate the ANFs of the intermediate state bits in Grain-128AEAD, and the pseudo-code of the modified update function in the initialization is given as follows.

$$\begin{aligned}
g &\leftarrow b_0 \oplus b_{26} \oplus b_{56} \oplus b_{91} \oplus b_{96} \oplus b_3 b_{67} \oplus b_{11} b_{13} \oplus b_{17} b_{18} \oplus b_{27} b_{59} \oplus b_{40} b_{48} \\
&\quad \oplus b_{61} b_{65} \oplus b_{68} b_{84} \oplus b_{88} b_{92} b_{93} b_{95} \oplus b_{22} b_{24} b_{25} \oplus b_{70} b_{78} b_{82}, && (4)\\
f' &\leftarrow s_7 \oplus s_{38} \oplus s_{70} \oplus s_{81} \oplus s_{96}, && (5)\\
h &\leftarrow b_{12} s_8 \oplus s_{13} s_{20} \oplus b_{95} s_{42} \oplus s_{60} s_{79} \oplus b_{12} b_{95} s_{94}, && (6)\\
z' &\leftarrow s_0 \oplus h \oplus s_{93} \oplus b_2 \oplus b_{15} \oplus b_{36} \oplus b_{45} \oplus b_{64} \oplus b_{73} \oplus b_{89}, && (7)\\
&\quad (b_0, b_1, \ldots, b_{127}) \leftarrow (b_1, b_2, \ldots, b_{127}, g \oplus z'),\\
&\quad (s_0, s_1, \ldots, s_{127}) \leftarrow (s_1, s_2, \ldots, s_{127}, f' \oplus z').
\end{aligned}$$

Constructing the expression Eq.(3) derived by r-round Grain-128AEAD $f(\boldsymbol{k}, \boldsymbol{v})$ is illustrated as PARAMSELECTOR32 in Algorithm 1.

MILP Model for Grain-128AEAD. GRAIN-128AEADMODEL in Algorithm 2 is used to build a complete MILP model for Grain-128AEAD, and this model can evaluate all three-subset division trails for Grain-128AEAD whose initialization rounds are reduced to r. GRAIN-128AEADINIT32 in Algorithm 3 is used to model the algebraic relation $(\boldsymbol{s}_{32}, \boldsymbol{b}_{32}) = \boldsymbol{E}'_{32}(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}})$ in Eq.(3). FUNCG generates MILP variables and constraints for Eq.(4), FUNF generates MILP variables and

Algorithm 1: The Auxiliary Functions for Building an MILP Model

procedure GRAIN-128AEAD(state $(\boldsymbol{s}, \boldsymbol{b})$, cube indices I, round R)
  $\boldsymbol{c} = (c_0, \ldots, c_{95}) \in \mathbb{F}_2^{96}$, if $i \in I$, $c_i = 1$, otherwise $c_i = 0$
  $(s_0, s_1, \ldots, s_{127}) \leftarrow (c_0 \cdot v_0, \ldots, c_{95} \cdot v_{95}, 1, \ldots, 1, 0)$
  $(b_0, b_1, \ldots, b_{127}) \leftarrow (k_0, \ldots, k_{127})$
  for i **from** 0 **to** $R-1$ **do**
    $g_i \leftarrow b_{0+i} \oplus b_{26+i} \oplus b_{56+i} \oplus b_{91+i} \oplus b_{96+i} \oplus b_{3+i}b_{67+i} \oplus b_{11+i}b_{13+i} \oplus b_{17+i}b_{18+i}$
      $\oplus b_{27+i}b_{59+i} \oplus b_{40+i}b_{48+i} \oplus b_{61+i}b_{65+i} \oplus b_{68+i}b_{84+i} \oplus b_{88+i}b_{92+i}b_{93+i}b_{95+i}$
      $\oplus b_{22+i}b_{24+i}b_{25+i} \oplus b_{70+i}b_{78+i}b_{82+i}$
    $f'_i \leftarrow s_{7+i} \oplus s_{38+i} \oplus s_{70+i} \oplus s_{81+i} \oplus s_{96+i}$
    $z'_i \leftarrow b_{12+i}s_{8+i} \oplus s_{13+i}s_{20+i} \oplus b_{95+i}s_{42+i} \oplus s_{60+i}s_{79+i} \oplus b_{12+i}b_{95+i}s_{94+i}$
      $\oplus s_{93+i} \oplus b_{2+i} \oplus b_{15+i} \oplus b_{36+i} \oplus b_{45+i} \oplus b_{64+i} \oplus b_{73+i} \oplus b_{89+i} \oplus s_{0+i}$
  $\boldsymbol{f'} \leftarrow (f'_0, \ldots, f'_{31})$
  $\boldsymbol{g} \leftarrow (g_0, \ldots, g_{31}$
  $\boldsymbol{z'} \leftarrow (z'_0, \ldots, z'_{31})$
  Return $(\boldsymbol{s}, \boldsymbol{b})$, $\boldsymbol{f'}, \boldsymbol{g}, \boldsymbol{z'}$
end procedure

procedure PARAMSELECTOR32(cube indices I)
  $s_i, b_i \in \mathbb{F}_2[\boldsymbol{k}, \boldsymbol{v}]$ for $i \in \{0, \ldots, 127\}$ ▷ $\boldsymbol{k} \in \mathbb{F}_2^{128}$, $\boldsymbol{v} \in \mathbb{F}_2^{96}$
  $(\boldsymbol{s}, \boldsymbol{b}), \boldsymbol{f'}, \boldsymbol{g}, \boldsymbol{z'} \leftarrow$ GRAIN-128AEAD$((\boldsymbol{s}, \boldsymbol{b}), I, 32)$
  for i **from** 0 **to** 95 **do**
    $s_i \leftarrow s_{i+32}$
    $b_i \leftarrow b_{i+32}$
  Prepare two empty polynomial sequence $\boldsymbol{h}^{(k)}, \boldsymbol{h}^{(v)}$
  for i **from** 0 **to** 127 **do**
    $h_i^{(k)} \leftarrow k_i$
  $loc1, loc2 \leftarrow 128, 0$
  for i **from** 0 **to** 31 **do**
    for p **in** $\{f'_i, g_i, z'_i\}$ **do**
      Decompose $p = \varphi_{i,1}(\boldsymbol{k}, \boldsymbol{v}) \oplus \varphi_{i,2}(\boldsymbol{k})$
        ▷ there is no monomial $\boldsymbol{k}^{\boldsymbol{\mu}}$ in $\varphi_{i,1}(\boldsymbol{k}, \boldsymbol{v})$ for any $\boldsymbol{\mu} \in \mathbb{F}_2^{128}$
      if there are at least two monomials in $\varphi_{i,2}(\boldsymbol{k})$ **then**
        if $\varphi_{i,2}(\boldsymbol{k})$ is included in $\boldsymbol{h}^{(k)}$ **and** $h_l^{(k)} = \varphi_{i,2}(\boldsymbol{k})$ **then**
          $p \leftarrow \varphi_{i,1}(\boldsymbol{k}, \boldsymbol{v}) \oplus k_j$
        else
          $h_{loc1}^{(k)} \leftarrow \varphi_{i,2}(\boldsymbol{k})$
          Define the variable $k_{loc1} \in \mathbb{F}_2$
          $p \leftarrow \varphi_{i,1}(\boldsymbol{k}, \boldsymbol{v}) \oplus k_{loc1}$
          $loc1 \leftarrow loc1 + 1$
      if there are at least two monomials in p **then**
        $h_{loc2}^{(v)} \leftarrow p$
        Define the variable $v_{loc2} \in \mathbb{F}_2$
        $p \leftarrow \hat{v}_{loc2}$
      $s_{96+i} \leftarrow f' \oplus z'$
      $b_{96+i} \leftarrow g \oplus z'$
  Return $(\boldsymbol{s}, \boldsymbol{b})$, $\boldsymbol{h}^{(k)}$, $\boldsymbol{h}^{(v)}$.
end procedure

Algorithm 2: The New MILP Model for Grain-128AEAD (1)

procedure Grain-128aeadModel(Round R, I, Monomial $\boldsymbol{\omega}$)
  $(\boldsymbol{Ps}, \boldsymbol{Pb})$, $\boldsymbol{h}^{(k)}, \boldsymbol{h}^{(v)} \leftarrow$ ParamSelector32(I)
  $n \leftarrow \boldsymbol{h}^{(k)}.size()$ ▷ Assign the number of the secret key variables to n
  $m \leftarrow \boldsymbol{h}^{(v)}.size()$ ▷ Assign the number of the variables $\hat{\boldsymbol{v}}$ to m
  Prepare an empty MILP Model $\mathcal{M}_{32}$
  $\mathcal{M}_{32}.var \leftarrow \boldsymbol{s} = (s_0, \ldots, s_{128})$ as binary variables
  $\mathcal{M}_{32}.var \leftarrow \boldsymbol{b} = (b_0, \ldots, b_{128})$ as binary variables
  $\mathcal{M}_{32}.var \leftarrow \boldsymbol{k} = (k_0, \ldots, k_{n-1})$ as binary variables
  $\mathcal{M}_{32}.var \leftarrow \boldsymbol{v} = (v_0, \ldots, v_{95})$ as binary variables
  $\mathcal{M}_{32}.con \leftarrow v_i = 1$ for $i \in I$
  $\mathcal{M}_{32}.con \leftarrow v_i = 0$ for $i \in \{0, \ldots, 95\} \backslash I$
  $(\boldsymbol{s}, \boldsymbol{b}) \leftarrow$ Grain-128aeadInit32($\mathcal{M}_{32}$, $\boldsymbol{Polys}$, $\boldsymbol{Polyb}$, $\boldsymbol{s}$, $\boldsymbol{b}$, $\boldsymbol{k}$, $\boldsymbol{v}$, $\boldsymbol{h}^{(v)}$, n, m)
  for r **from** 33 **to** R **do**
    $\boldsymbol{s}^{(1)}, \boldsymbol{b}^{(1)}, g \leftarrow$ funcG($\mathcal{M}_{32}$, $\boldsymbol{s}$, $\boldsymbol{b}$)
    $\boldsymbol{s}^{(2)}, \boldsymbol{b}^{(2)}, f' \leftarrow$ funcF($\mathcal{M}_{32}$, $\boldsymbol{s}^{(1)}$, $\boldsymbol{b}^{(1)}$)
    $\boldsymbol{s}^{(3)}, \boldsymbol{b}^{(3)}, z' \leftarrow$ funcZ($\mathcal{M}_{32}$, $\boldsymbol{s}^{(2)}$, $\boldsymbol{b}^{(2)}$, 1)
    $\mathcal{M}_{32}.var \leftarrow zg, zf, news, newb$ as binary variables
    $\mathcal{M}_{32}.con \leftarrow z' = zg \vee zf$
    $\mathcal{M}_{32}.con \leftarrow news = zf + f'$
    $\mathcal{M}_{32}.con \leftarrow newb = zg + g$
    $\boldsymbol{s} \leftarrow (s_1^{(3)}, \ldots, s_{126}^{(3)}, news)$
    $\boldsymbol{b} \leftarrow (b_1^{(3)}, \ldots, b_{126}^{(3)}, newb)$
  if $\boldsymbol{\omega} = (0, \ldots, 0)$ **then**
    $\boldsymbol{s}', \boldsymbol{b}', z \leftarrow$ funcZ($\mathcal{M}_{32}$, $\boldsymbol{s}$, $\boldsymbol{b}$, 0)
    for i **from** 0 **to** 127 **do**
      $\mathcal{M}_{32}.con \leftarrow s'_i = 0$
      $\mathcal{M}_{32}.con \leftarrow b'_i = 0$
    $\mathcal{M}_{32}.con \leftarrow z = 1$
  else
    $\mathcal{M}_{32}.con \leftarrow s_i = \omega_i,\ i \in \{0, \ldots, 127\}$
    $\mathcal{M}_{32}.con \leftarrow b_i = \omega_{128+i},\ i \in \{0, \ldots, 127\}$
  return $\mathcal{M}_{32}, \boldsymbol{g}^{(k)}$
end procedure

procedure ModelPoly($\mathcal{M}$, p, f, $\boldsymbol{k}'$, $\boldsymbol{v}'$, $\hat{\boldsymbol{v}}'$, $\boldsymbol{c}^k$, $\boldsymbol{c}^v$, $\boldsymbol{c}^{\hat{v}}$)
  $\mathcal{M}.LinExpr \leftarrow g = 0$
  $\mathcal{M}.var \leftarrow t$ as binary variable
  for each monomial $\boldsymbol{k}^{\boldsymbol{\mu}}\boldsymbol{v}^{\boldsymbol{\nu}}\hat{\boldsymbol{v}}^{\boldsymbol{\omega}}$ **in** f **do**
    for l **from** 0 **to** $n-1$ **do**
      if $\mu_l = 1$ **then**
        $\mathcal{M}.con \leftarrow t = k'_{l,c_l^k}$
        $c_l^k \leftarrow c_l^k + 1$
    for l **from** 0 **to** 95 **do**
      if $\nu_l = 1$ **then**
        $\mathcal{M}.con \leftarrow t = v'_{l,c_l^v}$
        $c_l^v \leftarrow c_l^v + 1$
    for l **from** 0 **to** $m-1$ **do**
      if $\omega_l = 1$ **then**
        $\mathcal{M}.con \leftarrow t = \hat{v}'_{l,c_l^{\hat{v}}}$
        $c_l^{\hat{v}} \leftarrow c_l^{\hat{v}} + 1$
    $g \leftarrow g + t$
  $\mathcal{M}.con \leftarrow p = g$
end procedure

Algorithm 3: The New MILP Model for Grain-128AEAD (2)

procedure Grain-128aeadInit32($\mathcal{M}_{64}, \boldsymbol{Ps}, \boldsymbol{Pb}, \boldsymbol{s}, \boldsymbol{b}, \boldsymbol{k}, \boldsymbol{v}, \boldsymbol{h}^{(v)}, n, m$)
  $\mathcal{M}_{32}.var \leftarrow \hat{\boldsymbol{v}} = (\hat{v}_0, \ldots, \hat{v}_{m-1})$ as binary variables
  $\boldsymbol{n}^k, \boldsymbol{n}^v \leftarrow$ **CountNum**($\boldsymbol{h}^{(v)}$) ▷ Count the number of $\boldsymbol{k}$ and $\boldsymbol{v}$ in the polynomials $\boldsymbol{h}^{(v)}$
  for i **from** 0 **to** $n-1$ **do**
    if $n_i^{(k)} > 0$ **then**
      $\mathcal{M}_{32}.var \leftarrow k'_i = (k'_{i,0}, k'_{i,1}, \ldots, k'_{i,n_i^{(k)}})$ as binary variables
      $\mathcal{M}_{32}.con \leftarrow k_i = k'_{i,0} \vee k'_{i,1} \vee \cdots \vee k'_{i,n_i^{(k)}}$
      $k_i \leftarrow k'_{i,0}$
  for i **from** 0 **to** 95 **do**
    if $n_i^{(v)} > 0$ **then**
      $\mathcal{M}_{32}.var \leftarrow v'_i = (v'_{i,0}, v'_{i,1}, v'_{i,2}, \ldots, v'_{i,n_i^{(v)}})$ as binary variables
      $\mathcal{M}_{32}.con \leftarrow v_i = v'_{i,0} \vee v'_{i,1} \vee \cdots \vee v'_{i,n_i^{(v)}}$
      $v_i \leftarrow v'_{i,0}$
  $\boldsymbol{c}^k \leftarrow (1, \ldots, 1) \in \mathbb{Z}^n$
  $\boldsymbol{c}^v \leftarrow (1, \ldots, 1) \in \mathbb{Z}^{96}$
  $\boldsymbol{k'} \leftarrow (k'_0, \ldots, k'_{n-1})$
  $\boldsymbol{v'} \leftarrow (v'_0, \ldots, v'_{95})$
  for i **from** 0 **to** $m-1$ **do**
    ModelPoly($\mathcal{M}_{32}, \hat{v}_i, h_i^{(v)}, \boldsymbol{k'}, \boldsymbol{v'}, \boldsymbol{c}^k, \boldsymbol{c}^v$)
  $\boldsymbol{n}^k, \boldsymbol{n}^v, \boldsymbol{n}^{\hat{v}} \leftarrow$ **CountNum**($\boldsymbol{Ps}, \boldsymbol{Pb}$)
  for i **from** 0 **to** $n-1$ **do**
    if $n_i^{(k)} = 0$ **then**
      $\mathcal{M}_{32}.con \leftarrow k_i = 0$
    else
      $\mathcal{M}_{32}.var \leftarrow k'_i = (k'_{i,1}, k'_{i,2}, \ldots, k'_{i,n_i^{(k)}})$ as binary variables
      $\mathcal{M}_{32}.con \leftarrow k_i = k'_{i,1} \vee \cdots \vee k'_{i,n_i^{(k)}}$
  for i **from** 0 **to** 95 **do**
    if $n_i^{(v)} = 0$ **then**
      $\mathcal{M}_{32}.con \leftarrow v_i = 0$
    else
      $\mathcal{M}_{32}.var \leftarrow v'_i = (v'_{i,1}, v'_{i,2}, \ldots, v'_{i,n_i^{(v)}})$ as binary variables
      $\mathcal{M}_{32}.con \leftarrow v_i = v'_{i,1} \vee \cdots \vee v'_{i,n_i^{(v)}}$
  for i **from** 0 **to** $m-1$ **do**
    if $n_i^{(\hat{v})} = 0$ **then**
      $\mathcal{M}_{32}.con \leftarrow \hat{v}_i = 0$
    else
      $\mathcal{M}_{32}.var \leftarrow \hat{v}'_i = (\hat{v}'_{i,1}, \hat{v}'_{i,2}, \ldots, \hat{v}'_{i,n_i^{(\hat{v})}})$ as binary variables
      $\mathcal{M}_{32}.con \leftarrow \hat{v}_i = \hat{v}'_{i,1} \vee \cdots \vee \hat{v}'_{i,n_i^{(\hat{v})}}$
  $\boldsymbol{c}^k \leftarrow (1, \ldots, 1) \in \mathbb{Z}^n$
  $\boldsymbol{c}^v \leftarrow (1, \ldots, 1) \in \mathbb{Z}^{96}$
  $\boldsymbol{c}^{\hat{v}} \leftarrow (1, \ldots, 1) \in \mathbb{Z}^m$
  $\boldsymbol{k'} \leftarrow (k'_0, \ldots, k'_{n-1})$
  $\boldsymbol{v'} \leftarrow (v'_0, \ldots, v'_{95})$
  $\hat{\boldsymbol{v}}' \leftarrow (\hat{v}'_0, \ldots, \hat{v}'_{m-1})$
  for i **from** 0 **to** 287 **do**
    ModelPoly($\mathcal{M}_{32}, s_i, Ps[i], \boldsymbol{k'}, \boldsymbol{v'}, \hat{\boldsymbol{v}}', \boldsymbol{c}^k, \boldsymbol{c}^v, \boldsymbol{c}^{\hat{v}}$)
    ModelPoly($\mathcal{M}_{32}, b_i, Pb[i], \boldsymbol{k'}, \boldsymbol{v'}, \hat{\boldsymbol{v}}', \boldsymbol{c}^k, \boldsymbol{c}^v, \boldsymbol{c}^{\hat{v}}$)
  Return $\boldsymbol{s}, \boldsymbol{b}$
end procedure

constraints for Eq.(5), and FUNZ generates MILP variables and constraints for Eq.(6) and Eq.(7). MILP models for these three functions are represented in Algorithm 4.

3.3 Further improvement for Grain-128AEAD

The idea of improving the modeling technique is consistent with the previous one; that is, by increasing d, more expressions with respect to the secret key can be replaced by the new variables, thereby further reducing the number of feasible solutions in the model and improving the solution speed of the model. Similarly, to select a larger d, we still adopt the divide-and-conquer strategy to avoid the extra three-subset division trails. We choose $d = 96$ based on many experiments. Since with the increase of d, the solving ability of the improved model does not increase significantly. Next, we will introduce how to build the improved MILP model for Grain-128AEAD.

Firstly, we represent $G(\boldsymbol{k}, \boldsymbol{v})$ as Eq.(3), where $(\boldsymbol{s}_{32}, \boldsymbol{b}_{32}) = \boldsymbol{E}'_{32}(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}})$. According to the ANF of each component Boolean function in the vectorial Boolean function $\boldsymbol{E}'_{32}$, they are mainly divided into two cases as follows,

Case 1: k_i or v_i or $\hat{v}_i$ or c,
Case 2: $\hat{v}_i \oplus \hat{v}_j$,

where c means a constant value. To avoid existing extra three-subset division trails in the improved model, we replace the expressions of Case 2 with new variables $\hat{v}_{\beta_1}, \ldots, \hat{v}_{\beta_2-1}$. Let $\hat{\boldsymbol{v}}^{(1)} = (\hat{v}_0, \ldots, \hat{v}_{\beta_1-1}, \hat{v}_{\beta_1}, \ldots, \hat{v}_{\beta_2-1}) \in \mathbb{F}_2^{\beta_2}$. Thus, the state bits $(\boldsymbol{s}_{32}, \boldsymbol{b}_{32})$ can further represented as the polynomials of $\boldsymbol{k}^{(1)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}^{(1)}$, denoted by $(\boldsymbol{s}_{32}, \boldsymbol{b}_{32}) = \boldsymbol{E}''_{32}(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(1)})$.

Similarly, for $32 \leq i < 64$, the functions $f'(\boldsymbol{s}_i)$, $g(\boldsymbol{b}_i)$ and $z'(\boldsymbol{s}_i, \boldsymbol{b}_i)$ can be represented as the polynomials of $\boldsymbol{s}_{32}$ and $\boldsymbol{b}_{32}$. Further, due to $(\boldsymbol{s}_{32}, \boldsymbol{b}_{32}) = \boldsymbol{E}''_{32}(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(1)})$, the functions $f'(\boldsymbol{s}_i)$, $g(\boldsymbol{b}_i)$ and $z'(\boldsymbol{s}_i, \boldsymbol{b}_i)$ can also be represented as the polynomials of $\boldsymbol{k}^{(1)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}^{(1)}$, denoted by $f'_i(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(1)})$, $g_i(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(1)})$ and $z'_i(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(1)})$ respectively. Thus, $G(\boldsymbol{k}, \boldsymbol{v})$ can be represented as follows,

$$G(\boldsymbol{k}, \boldsymbol{v}) = F_o \circ \boldsymbol{F}_r \circ \cdots \circ \boldsymbol{F}_{65} \circ \boldsymbol{E}_{64}(\boldsymbol{k}^{(1)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(1)}),$$

where $\boldsymbol{E}_{64} = (s_{32,32}, \ldots, s_{32,127}, f'_{32} \oplus z'_{32}, \ldots, f'_{63} \oplus z'_{63}, b_{32,32}, \ldots, b_{32,127}, g_{32} \oplus z'_{32}, \ldots, g_{63} \oplus z'_{63})$. By using the same approach as before, we replace the different expressions of $\boldsymbol{k}^{(1)}$ involved in f'_i,g_i and z'_i by the new secret key variables $k_{\alpha_1}, \ldots, k_{\alpha_2-1}$. Let $\boldsymbol{k}^{(2)} = (k_0, \ldots, k_{\alpha_1-1}, k_{\alpha_1}, \ldots, k_{\alpha_2-1}) \in \mathbb{F}_2^{\alpha_2}$. Thus, these functions can be represented as the polynomials of $\boldsymbol{k}^{(2)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}^{(1)}$, and according to their ANFs, they are mainly divided into two cases as follows,

Case 1: k_i or v_i or $\hat{v}_i$ or c,
Case 2: $\varphi(\boldsymbol{k}^{(2)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(1)})$ or $\varphi(\boldsymbol{k}^{(2)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(1)}) \oplus k_j$,

where c means a constant value. We further process the expressions belonging to Case 2 by replacing them with new IV variables $\hat{v}_{\beta_2}, \ldots, \hat{v}_{\beta_3-1}$. Therefore, the state bits $s_{64,96}, \ldots, s_{64,127}, b_{64,96}, \ldots, b_{64,127}$ can be represented as the form $\hat{v}_i \oplus \hat{v}_j$, where $i, j \in \{\beta_2, \ldots, \beta_3 - 1\}$.

Next, we repeat the above process. We first replace these expressions with new variables $\hat{v}_{\beta_3}, \ldots, \hat{v}_{\beta_4-1}$. Let $\hat{\boldsymbol{v}}^{(2)} = (\hat{v}_0, \ldots, \hat{v}_{\beta_2-1}, \hat{v}_{\beta_2}, \ldots, \hat{v}_{\beta_3-1}, \hat{v}_{\beta_3}, \ldots, \hat{v}_{\beta_4-1})$. Thus, the state bits $(\boldsymbol{s}_{64}, \boldsymbol{v}_{64})$ can be represented as the polynomials of $\boldsymbol{k}^{(2)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}^{(2)}$. Then, we deal with the functions f', g and z' for each round $r \in \{64, \ldots, 95\}$ as before, and add new variables $k_{\alpha_2}, \ldots, k_{\alpha_3-1}$ and $\hat{v}_{\beta_4}, \ldots, \hat{v}_{\beta_5-1}$ to replace the special expressions involved in the ANFs of $(\boldsymbol{s}_{64}, \boldsymbol{v}_{64})$ with respect to $\boldsymbol{k}^{(2)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}^{(2)}$. Let $\boldsymbol{k}^{(3)} = (k_0, \ldots, k_{\alpha_2-1}, k_{\alpha_2}, \ldots, k_{\alpha_3-1})$ and $\hat{\boldsymbol{v}}^{(3)} = (\hat{v}_0, \ldots, \hat{v}_{\beta_4-1}, \hat{v}_{\beta_4}, \ldots, \hat{v}_{\beta_5-1})$. Therefore, the state bits $(\boldsymbol{s}_{96}, \boldsymbol{b}_{96})$ can be represented as the polynomials of $\boldsymbol{k}^{(3)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}^{(3)}$, denoted by $(\boldsymbol{s}_{96}, \boldsymbol{b}_{96}) = \boldsymbol{E}'_{96}(\boldsymbol{k}^{(3)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(3)})$. Accordingly, the function $G(\boldsymbol{k}, \boldsymbol{v})$ can be further represented as the following expression, i.e.,

$$G(\boldsymbol{k}, \boldsymbol{v}) = F_o \circ \boldsymbol{F}_r \circ \cdots \circ \boldsymbol{F}_{97} \circ \boldsymbol{E}'_{96}(\boldsymbol{k}^{(3)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(3)}). \tag{8}$$

According to Eq.(8), an improved MILP model $\mathcal{M}_{96}$ of Grain-128AEAD can be built. Similar to $\mathcal{M}_{32}$, when building the model $\mathcal{M}_{96}$, we first declare the variables $\boldsymbol{k}^{(3)}$, $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}^{(3)}$, and then establish the equations for the algebraic relations between $\hat{\boldsymbol{v}}^{(3)}$ and $\boldsymbol{k}^{(3)}$, $\boldsymbol{v}$ as well as $\hat{\boldsymbol{v}}^{(3)}$, which need to be added to the model $\mathcal{M}_{96}$. Next, the equations for the algebraic relations $(\boldsymbol{s}_{96}, \boldsymbol{b}_{96}) = \boldsymbol{E}'_{96}(\boldsymbol{k}^{(3)}, \boldsymbol{v}, \hat{\boldsymbol{v}}^{(3)})$ are established, and added to the model as the constraints. Finally, a complete model is built by an iterative method. Compared with the model $\mathcal{M}_{32}$, the improved model $\mathcal{M}_{96}$ does not include extra feasible solutions, which is guaranteed by the propagation properties of the 3SDP/u.

Algorithm 4: MILP model for NFSR and LFSR in Grain-128AEAD

procedure FUNCG($\mathcal{M}, \boldsymbol{s}, \boldsymbol{b}$)
  $\mathcal{M}.var \leftarrow (y_0, y_1, \ldots, y_{27}), (z_0, z_1, \ldots, z_{27}), g$ as binary variables
  $\mathcal{B} \leftarrow \{3, 67, 11, 13, 17, 18, 27, 59, 40, 48, 61, 65, 68, 84, 88, 92, 93, 95,$
    $22, 24, 25, 70, 78, 82, 26, 56, 91, 96\}$
  $j \leftarrow 0$
  for $i \in \mathcal{B}$ **do**
    $\mathcal{M}_{32}.con \leftarrow b_i = y_j \vee z_j$
    $b_i \leftarrow y_j$
    $j \leftarrow j + 1$
  $\mathcal{M}.con \leftarrow z_0 = z_1$ ▷ $b_3 b_{67}$
  $\mathcal{M}.con \leftarrow z_2 = z_3$ ▷ $b_{11} b_{13}$
  $\mathcal{M}.con \leftarrow z_4 = z_5$ ▷ $b_{17} b_{18}$
  $\mathcal{M}.con \leftarrow z_6 = z_7$ ▷ $b_{27} b_{59}$
  $\mathcal{M}.con \leftarrow z_8 = z_9$ ▷ $b_{40} b_{48}$
  $\mathcal{M}.con \leftarrow z_{10} = z_{11}$ ▷ $b_{61} b_{65}$
  $\mathcal{M}.con \leftarrow z_{12} = z_{13}$ ▷ $b_{68} b_{84}$
  $\mathcal{M}.con \leftarrow z_{14} = z_{15} = z16 = z17$ ▷ $b_{88} b_{92} b_{93} b_{95}$
  $\mathcal{M}.con \leftarrow z_{18} = z19 = z20$ ▷ $b_{22} b_{24} b_{25}$
  $\mathcal{M}.con \leftarrow z_{21} = z22 = z23$ ▷ $b_{70} b_{78} b_{82}$
  $\mathcal{I} \leftarrow \{0, 2, 4, 6, 8, 10, 12, 14, 18, 21\}$
  $\mathcal{M}.con \leftarrow g = \sum_{i \in \mathcal{I}} z_i + \sum_{j=24}^{27} z_j + b_0$
  Return $\mathcal{M}_{32}, \boldsymbol{s}, \boldsymbol{b}, g$
end procedure

procedure FUNCF($\mathcal{M}, \boldsymbol{s}, \boldsymbol{b}$)
  $\mathcal{M}.var \leftarrow (y_0, y_1, \ldots, y_4), (z_0, z_1, \ldots, z_4), f'$ as binary variables
  $\mathcal{S} \leftarrow \{7, 38, 70, 81, 96\}$
  $j \leftarrow 0$
  for $i \in \mathcal{S}$ **do**
    $\mathcal{M}_{32}.con \leftarrow s_i = y_j \vee z_j$
    $s_i \leftarrow y_j$, $j \leftarrow j + 1$
  $\mathcal{M}.con \leftarrow f' = z_0 + z_1 + z_2 + z_3 + z_4$
  Return $\mathcal{M}_{32}, \boldsymbol{s}, \boldsymbol{b}, f'$
end procedure

procedure FUNCZ($\mathcal{M}, \boldsymbol{s}, \boldsymbol{b}, flag$)
  $\mathcal{M}.var \leftarrow (ys_0, \ldots, ys_6), (zs_0, \ldots, zs_6), z$ as binary variables
  $\mathcal{M}.var \leftarrow (yb_0, \ldots, yb_{10}), (zb_0, \ldots, zb_{10})$ as binary variables
  $\mathcal{S} \leftarrow \{8, 13, 20, 42, 60, 79, 94, 93\}$
  $\mathcal{B} \leftarrow \{12, 95, 12, 95, 2, 15, 36, 45, 64, 73, 89\}$
  $j \leftarrow 0$
  for $i \in \mathcal{S}$ **do**
    $\mathcal{M}_{32}.con \leftarrow s_i = ys_j \vee zs_j$
    $s_i \leftarrow ys_j$, $j \leftarrow j + 1$
  $j \leftarrow 0$
  for $i \in \mathcal{B}$ **do**
    $\mathcal{M}_{32}.con \leftarrow b_i = yb_j \vee zb_j$
    $b_i \leftarrow yb_j$, $j \leftarrow j + 1$
  $\mathcal{M}.con \leftarrow zb_0 = zs_0$ ▷ $b_{12} s_8$
  $\mathcal{M}.con \leftarrow zs_1 = zs_2$ ▷ $s_{13} s_{20}$
  $\mathcal{M}.con \leftarrow zb_1 = zs_3$ ▷ $b_{95} s_{42}$
  $\mathcal{M}.con \leftarrow zs_4 = zs_5$ ▷ $s_{60} s_{79}$
  $\mathcal{M}.con \leftarrow zb_2 = zb_3 = zs_6$ ▷ $b_{12} b_{95} s_{94}$
  if $flag = 1$ **do**
    $\mathcal{M}.con \leftarrow z = zb_0 + zs_1 + zb_1 + zs_4 + zb_2 + zs_7 + \sum_{i=4}^{10} zb_i + s_0$
  if $flag = 0$ **do**
    $\mathcal{M}.con \leftarrow z = zb_0 + zs_1 + zb_1 + zs_4 + zb_2 + zs_7 + \sum_{i=4}^{10} zb_i$
  Return $\mathcal{M}_{32}, \boldsymbol{s}, \boldsymbol{b}, z$
end procedure

4 The results of verification experiments on Grain-128AEAD

For Grain-128AEAD, the method of superpolies recovery we used in the experiments is the divide-and-conquer algorithm, and we recovered the superpolies of 190-round. The cube indices sets we used come from [2,3]. Table 1 shows the statistics of the results and provides experimental proof of the effectiveness of the new modeling technique for Grain-128AEAD. Similar to Trivium, the improved MILP model $\mathcal{M}_{96}$ has the fewest feasible solutions and the fastest solving speed among the three different models; the traditional model is the worst performer. Different from Trivium, although the superpoly recovered by the improved model $\mathcal{M}_{96}$ still has the fewest monomials, its algebraic degree does not decrease significantly compared with the corresponding superpolies recovered by the other models.

Table 1: statistics of the results for different MILP models of 190-round Grain-128AEAD

I	# of trails			time (s)			# of monomials in $p_I(\cdot)$			degree of $p_I(\cdot)$		
	$\mathcal{M}_T$	$\mathcal{M}_{32}$	$\mathcal{M}_{96}$	$\mathcal{M}_T$	$\mathcal{M}_{32}$	$\mathcal{M}_{96}$	$\boldsymbol{k}$	$\boldsymbol{k}^{(1)}$	$\boldsymbol{k}^{(3)}$	$\boldsymbol{k}$	$\boldsymbol{k}^{(1)}$	$\boldsymbol{k}^{(3)}$
$\{0,\ldots,95\}\backslash\{26\}$	74263	50039	36856	12440	3778	591	11295	7695	6110	24	23	23
$\{0,\ldots,95\}\backslash\{29\}$	238076	117440	75164	13247	4107	671	18956	11188	8150	22	22	22
$\{0,\ldots,95\}\backslash\{30\}$	65785	20591	16615	8019	1742	412	1701	1205	1111	20	20	20
$\{0,\ldots,95\}\backslash\{31\}$	87518	45072	34926	11766	2598	546	11362	4600	3490	23	22	21
$\{0,\ldots,95\}\backslash\{33\}$	106427	65395	43298	13359	2928	597	18245	7995	6132	22	22	22
$\{0,\ldots,95\}\backslash\{40\}$	4780	3390	3015	6528	1382	207	1238	752	605	20	20	20
$\{0,\ldots,95\}\backslash\{43\}$	3188	2056	1743	3866	894	160	1358	852	641	20	20	20
$\{0,\ldots,95\}\backslash\{44\}$	790	704	639	1968	1014	182	176	156	127	19	19	19
$\{0,\ldots,95\}\backslash\{45\}$	71307	45809	30737	9172	2397	522	10273	5659	3935	24	23	21
$\{0,\ldots,95\}\backslash\{47\}$	35525	24445	19845	7975	1912	539	4235	3509	2939	23	23	23
$\{0,\ldots,95\}\backslash\{57\}$	39510	24810	20893	13281	3631	605	11746	7004	5989	23	22	22
$\{0,\ldots,95\}\backslash\{58\}$	78029	50703	41789	16277	4769	602	15149	10619	8973	24	23	23
$\{0,\ldots,95\}\backslash\{63\}$	159445	84691	64065	18094	4810	661	19925	8719	7557	24	24	24
$\{0,\ldots,95\}\backslash\{69\}$	201185	98449	63327	12669	3768	599	13109	7949	5841	22	22	22
$\{0,\ldots,95\}\backslash\{71\}$	45541	24559	21056	10978	3120	607	7055	4029	3401	24	23	23

References

1. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of Grain-128 with optional authentication. Int. J. Wirel. Mob. Comput. **5**, 48–59 (2011)
2. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128AEAD. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020,

Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 466–495. Springer, Cham (2020), https://doi.org/10.1007/978-3-030-45721-1_17
3. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset. J. Cryptol. **34**(3), 22 (2021). https://doi.org/10.1007/s00145-021-09383-2, https://doi.org/10.1007/s00145-021-09383-2
4. Hell, M., Johansson, T., Meier, W., Sönnerup, J., Yoshida, H.: An AEAD variant of the Grain stream cipher. In: International Conference on Codes, Cryptology, and Information Security (2019)