Supplementary Materials

No Author Given

No Institute Given

Abstract. We give details of the targeted ciphers including the specifications and the MILP models for describing the division property propagation. All the algorithms have been realized with our C++ source codes.

1 Details of Grain-128

1.1 Specification of Grain-128

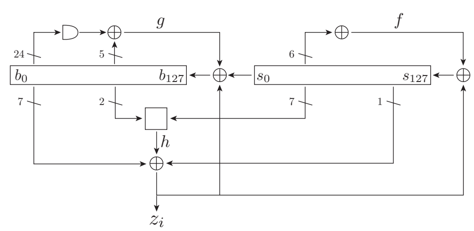

Fig. 1. Structure of Grain-128

Grain-128 [1] is a NLFSR-based stream cipher. It takes as input 128 key bits $x_{[128)}$ and 96 IV bits $v_{[96)}$. Its internal state consists of an LFSR and an NFSR, both of length 128 bits. The NFSR and LFSR, denoted as $\boldsymbol{b}$, $\boldsymbol{s}$, are initialized by the key and IV bits respectively as follows:

$$\boldsymbol{b}^0 = (b_0, b_1, \ldots, b_{127}) = (x_0, \ldots, x_{127}), \boldsymbol{s}^0 = (s_0, s_1, \ldots, s_{127}) = (v_0, \ldots, v_{95}, 1, \ldots, 1).$$

After that, for $r = 0, \ldots, 255$, the NLFSRs are updated by calling `Upd` as follows:

$$
\begin{aligned}
&h^r \leftarrow b_{r+12}s_{r+8} + s_{r+13}s_{r+20} + b_{r+95}s_{r+42} + s_{r+60}s_{r+79} + b_{r+12}b_{r+95}s_{95} \\
&z^r \leftarrow h^r + s_{r+93} + \sum_{j\in A} b_{r+j} \text{ where } A = \{2, 15, 36, 45, 64, 73, 89\} && (1) \\
&g^r \leftarrow b_r + b_{r+26} + b_{r+56} + b_{r+91} + b_{r+96} + b_{r+3}b_{r+67} + b_{r+11}b_{r+13} + b_{r+17}b_{r+18} \\
&\qquad + b_{r+27}b_{r+59} + b_{r+40}b_{r+48} + b_{r+61}b_{r+65} + b_{r+68}b_{r+84} \\
&b_{r+128} \leftarrow g^r + z^r + s_r && (2) \\
&f^r \leftarrow s_r + s_{r+7} + s_{r+38} + s_{r+70} + s_{r+81} + s_{r+96} \\
&s_{r+128} \leftarrow f^r + z^r && (3) \\
&\boldsymbol{b}^r \leftarrow (b_{r+1}, \ldots b_{r+128}) && (4) \\
&\boldsymbol{s}^r \leftarrow (s_{r+1}, \ldots s_{r+128}) && (5)
\end{aligned}
$$

Finally, the first keystream bit z^{256} is output, computed as (1) with parameter $r = 256$. Such a procedure can be reflected by Fig. 1.

1.2 MILP Models of Grain-128

The division property propagation corresponding to the whole procedure of updating function can be constructed by calling `UpdDiv` Algorithm 2. The subroutines `funcZ`, `funcG`, `funcF` are defined in Algorithm 3, where `funcZ` can also describe the DP structure of the output bit. The initialization `IniDP` is defined in Algorithm 1.

Algorithm 1 The initial DP structures for Grain-128/128a.

1: **procedure** `IniDP`(MILP model $\mathcal{M}$, cube index set I, non-cube IV assignment $\boldsymbol{IV}$, initialization round number R, split set Λ.)
2: Declare the DP structures $\boldsymbol{k}_x, \boldsymbol{k}_v$ both of length 128.
3: For $i \in I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 1$ and $\boldsymbol{k}_v[i].F = \delta$.
4: For $i \notin I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 0$ and assign the flag value according to $\boldsymbol{IV}[i]$: $\boldsymbol{k}_v[i].F = 1_c$ if $\boldsymbol{IV}[i] = 1$ and $\boldsymbol{k}_v[i].F = 0_c$ if $\boldsymbol{IV}[i] = 0$.
5: Set $\mathcal{M}.con \leftarrow \boldsymbol{k}_x[\Lambda].val = 0$ and $\boldsymbol{k}_x[\Lambda].F = 0_c$. Set $\boldsymbol{k}_x[j].F = \delta$ for $j \notin \Lambda$.
6: **return** $(\mathcal{M}, \boldsymbol{k}_x, \boldsymbol{k}_v)$.
7: **end procedure**

2 Details of Grain-128a

2.1 Specification of Grain-128a

Grain-128a [2] share the same structure with Grain-128. They both consists of a NFSR and a LFSR, both of length 128. They both have 128 key bits $x_{[128)}$ and 96 IV bits $v_{[96)}$. They both requires $R = 256$ initialization rounds. The differences can be seen in many details as well. For Grain-128a, the initialization of the NFSR and LFSR are as follows:

$$\boldsymbol{b}^0 = (b_0, b_1, \ldots, b_{127}) = (x_0, \ldots, x_{127}), \boldsymbol{s}^0 = (s_0, s_1, \ldots, s_{127}) = (v_0, \ldots, v_{95}, 1, \ldots, 1, 0).$$

Algorithm 2 MILP model for Grain-128/128a updating function.

procedure `UpdDiv`(the current MILP model $\mathcal{M}$, the DP structure $\boldsymbol{k}^{r-1} = (\boldsymbol{k}_b^{r-1}, \boldsymbol{k}_s^{r-1})$ where $\boldsymbol{k}_b$ and $\boldsymbol{k}_s$ are of length 128 corresponding to the DP structures of $\boldsymbol{b}^{r-1}, \boldsymbol{s}^{r-1}$ respectively, the round number r $(r = 1, 2 \ldots)$.)
 $(\mathcal{M}, \boldsymbol{k}'_b, \boldsymbol{k}'_s, z) = \texttt{funcZ}(\mathcal{M}, \boldsymbol{k}_b^{r-1}, \boldsymbol{k}_s^{r-1})$
 $(\mathcal{M}, z_g, z_f) \leftarrow \texttt{copyf}(\mathcal{M}, z)$
 $(\mathcal{M}, \boldsymbol{k}''_b, g) \leftarrow \texttt{funcG}(\mathcal{M}, \boldsymbol{k}'_b)$
 $(\mathcal{M}, \boldsymbol{k}''_s, f) \leftarrow \texttt{funcF}(\mathcal{M}, \boldsymbol{k}'_s)$
 $(\mathcal{M}, s_0^{\star}, s_0^{\star\star}) \leftarrow \texttt{copyf}(\mathcal{M}, \boldsymbol{k}''_s[0], 2)$
 $(\mathcal{M}, b_{r+127}) \leftarrow \texttt{xorf}(\mathcal{M}, g, s_0^{\star}, z_g)$
 $(\mathcal{M}, s_{r+127}) \leftarrow \texttt{xorf}(\mathcal{M}, f, s_0^{\star\star}, z_f)$
 Assign $\boldsymbol{k}_b^r \leftarrow \boldsymbol{k}''_b[1, \ldots, 127] \| b_{r+127}$ and $\boldsymbol{k}_s^r \leftarrow \boldsymbol{k}''_s[1, \ldots, 127] \| s_{r+127}$
 Assign $\boldsymbol{k}^r = (\boldsymbol{k}_b^r, \boldsymbol{k}_s^r)$.
end procedure

Algorithm 3 MILP model for NLFSRs in Grain-128

procedure `funcZ`$(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s})$
 $(\mathcal{M}, \boldsymbol{b}_1, \boldsymbol{s}_1, a_1) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s}, \{12\}, \{8\})$
 $(\mathcal{M}, \boldsymbol{b}_2, \boldsymbol{s}_2, a_2) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_1, \boldsymbol{s}_1, \phi, \{13, 20\})$
 $(\mathcal{M}, \boldsymbol{b}_3, \boldsymbol{s}_3, a_3) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_2, \boldsymbol{s}_2, \{95\}, \{42\})$
 $(\mathcal{M}, \boldsymbol{b}_4, \boldsymbol{s}_4, a_4) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_3, \boldsymbol{s}_3, \phi, \{60, 79\})$
 $(\mathcal{M}, \boldsymbol{b}_5, \boldsymbol{s}_5, a_5) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_4, \boldsymbol{s}_4, \{12, 95\}, \{95\})$
 $(\mathcal{M}, \boldsymbol{b}_6, \boldsymbol{s}_6, x) \leftarrow \texttt{CXOR}(\mathcal{M}, \boldsymbol{b}_5, \boldsymbol{s}_5, \{2, 15, 36, 45, 64, 73, 89\}, \{93\})$
 $(\mathcal{M}, z) \leftarrow \texttt{xorf}(\mathcal{M}, x, a_1, \ldots, a_5)$
 return $(\mathcal{M}, \boldsymbol{b}_6, \boldsymbol{s}_6, z)$
end procedure

procedure `funcF`$(\mathcal{M}, \boldsymbol{s})$
 $(\mathcal{M}, \phi, \boldsymbol{s}_1, f) \leftarrow \texttt{CXOR}(\mathcal{M}, \phi, \boldsymbol{s}, \phi, \{7, 38, 70, 81, 96\})$
 return $(\mathcal{M}, \boldsymbol{s}_1, f)$
end procedure

procedure `funcG`$(\mathcal{M}, \boldsymbol{b})$
 $(\mathcal{M}, \boldsymbol{b}_1, \phi, a_1) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}, \phi, \{3, 67\}, \phi)$
 $(\mathcal{M}, \boldsymbol{b}_2, \phi, a_2) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_1, \phi, \{11, 13\}, \phi)$
 $(\mathcal{M}, \boldsymbol{b}_3, \phi, a_3) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_2, \phi, \{17, 18\}, \phi)$
 $(\mathcal{M}, \boldsymbol{b}_4, \phi, a_4) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_3, \phi, \{27, 59\}, \phi)$
 $(\mathcal{M}, \boldsymbol{b}_5, \phi, a_5) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_4, \phi, \{40, 48\}, \phi)$
 $(\mathcal{M}, \boldsymbol{b}_6, \phi, a_6) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_5, \phi, \{61, 65\}, \phi)$
 $(\mathcal{M}, \boldsymbol{b}_7, \phi, a_7) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_6, \phi, \{68, 84\}, \phi)$
 $(\mathcal{M}, \boldsymbol{b}_{11}, \phi, x) \leftarrow \texttt{CXOR}(\mathcal{M}, \boldsymbol{b}_{10}, \phi, \{0, 26, 56, 91, 96\}, \phi)$
 $(\mathcal{M}, g) \leftarrow \texttt{xorf}(\mathcal{M}, x, a_1, \ldots, a_{10})$
 return $(\mathcal{M}, \boldsymbol{b}_{11}, g)$
end procedure

Algorithm 4 MILP model for COPY+XOR and COPY+AND in Grain-like stream ciphers

procedure $\texttt{CAND}(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s}, I, J)$
    $(\mathcal{M}, b'_i, x_i) \leftarrow \texttt{copyf}(\mathcal{M}, b_i)$ for all $i \in I$
    $(\mathcal{M}, s'_j, y_j) \leftarrow \texttt{copyf}(\mathcal{M}, s_j)$ for all $j \in J$
    for all $i \in \{0, 1, \ldots, 127\} - I$ **do**
        $b'_i = b_i$
    end for
    for all $j \in \{0, 1, \ldots, 127\} - J$ **do**
        $s'_i = s_i$
    end for
    $(\mathcal{M}, z) \leftarrow \texttt{andf}(\mathcal{M}, {b'}_{i,i \in I}, {s'}_{j,j \in J})$
    return $(\mathcal{M}, \boldsymbol{b}', \boldsymbol{s}', z)$
end procedure

procedure $\texttt{CXOR}(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s}, I, J)$
    $(\mathcal{M}, b'_i, x_i) \leftarrow \texttt{copyf}(\mathcal{M}, b_i)$ for all $i \in I$
    $(\mathcal{M}, s'_j, y_j) \leftarrow \texttt{copyf}(\mathcal{M}, s_j)$ for all $j \in J$
    for all $i \in \{0, 1, \ldots, 127\} - I$ **do**
        $b'_i = b_i$
    end for
    for all $j \in \{0, 1, \ldots, 127\} - J$ **do**
        $s'_i = s_i$
    end for
    $(\mathcal{M}, z) \leftarrow \texttt{xorf}(\mathcal{M}, {b'}_{i,i \in I}, {s'}_{j,j \in J})$
    return $(\mathcal{M}, \boldsymbol{b}', \boldsymbol{s}', z)$
end procedure

For $r = 0, \ldots, R-1$, the update function in the initialization round r is as follows.

$$
\begin{aligned}
& h^r \leftarrow b_{r+12}s_{r+8} + s_{r+13}s_{r+20} + b_{r+95}s_{r+42} + s_{r+60}s_{r+79} + b_{r+12}b_{r+95}s_{94} \\
& z^r \leftarrow h^r + s_{r+93} + \sum_{j \in A} b_{r+j} \text{ where } A = \{2, 15, 36, 45, 64, 73, 89\} && (6) \\
& g^r \leftarrow b_r + b_{r+26} + b_{r+56} + b_{r+91} + b_{r+96} + b_{r+3}b_{r+67} + b_{r+11}b_{r+13} + b_{r+17}b_{r+18} \\
& \qquad + b_{r+27}b_{r+59} + b_{r+40}b_{r+48} + b_{r+61}b_{r+65} + b_{r+68}b_{r+84} \\
& \qquad + +b_{r+88}b_{r+92}b_{r+93}b_{r+95} + b_{r+22}b_{r+24}b_{r+25} + b_{r+70}b_{r+78}b_{r+82} \\
& b_{r+128} \leftarrow g^r + z^r + s_r && (7) \\
& f^r \leftarrow s_r + s_{r+7} + s_{r+38} + s_{r+70} + s_{r+81} + s_{r+96} \\
& s_{r+128} \leftarrow f^r + z^r && (8) \\
& \boldsymbol{b}^r \leftarrow (b_{r+1}, \ldots b_{r+128}) && (9) \\
& \boldsymbol{s}^r \leftarrow (s_{r+1}, \ldots s_{r+128}) && (10)
\end{aligned}
$$

After R initialization rounds, the output function (6) is called for $r = R$ so the 1st output bit is z^R. Of course, full Grain-128a has $R = 256$ but our secure bounds consider $R < 256$.

2.2 MILP Models of Grain-128a

The updating function can also be represented as Algorithm 2 but the subroutines are substituted as Algorithm 5. These algorithms are first defined in [3].

Algorithm 5 MILP model for NLFSR and LFSR in Grain-128a

procedure $\texttt{funcZ}(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s})$
  $(\mathcal{M}, \boldsymbol{b}_1, \boldsymbol{s}_1, a_1) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s}, \{12\}, \{8\})$
  $(\mathcal{M}, \boldsymbol{b}_2, \boldsymbol{s}_2, a_2) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_1, \boldsymbol{s}_1, \phi, \{13, 20\})$
  $(\mathcal{M}, \boldsymbol{b}_3, \boldsymbol{s}_3, a_3) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_2, \boldsymbol{s}_2, \{95\}, \{42\})$
  $(\mathcal{M}, \boldsymbol{b}_4, \boldsymbol{s}_4, a_4) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_3, \boldsymbol{s}_3, \phi, \{60, 79\})$
  $(\mathcal{M}, \boldsymbol{b}_5, \boldsymbol{s}_5, a_5) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_4, \boldsymbol{s}_4, \{12, 95\}, \{94\})$
  $(\mathcal{M}, \boldsymbol{b}_6, \boldsymbol{s}_6, x) \leftarrow \texttt{CXOR}(\mathcal{M}, \boldsymbol{b}_5, \boldsymbol{s}_5, \{2, 15, 36, 45, 64, 73, 89\}, \{93\})$
  $(\mathcal{M}, z) \leftarrow \texttt{xorf}(\mathcal{M}, x, a_1, \ldots, a_5)$
  return $(\mathcal{M}, \boldsymbol{b}_6, \boldsymbol{s}_6, z)$
end procedure

procedure $\texttt{funcF}(\mathcal{M}, \boldsymbol{s})$
  $(\mathcal{M}, \phi, \boldsymbol{s}_1, f) \leftarrow \texttt{CXOR}(\mathcal{M}, \phi, \boldsymbol{s}, \phi, \{0, 7, 38, 70, 81, 96\})$
  return $(\mathcal{M}, \boldsymbol{s}_1, f)$
end procedure

procedure $\texttt{funcG}(\mathcal{M}, \boldsymbol{b})$
  $(\mathcal{M}, \boldsymbol{b}_1, \phi, a_1) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}, \phi, \{3, 67\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_2, \phi, a_2) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_1, \phi, \{11, 13\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_3, \phi, a_3) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_2, \phi, \{17, 18\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_4, \phi, a_4) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_3, \phi, \{27, 59\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_5, \phi, a_5) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_4, \phi, \{40, 48\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_6, \phi, a_6) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_5, \phi, \{61, 65\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_7, \phi, a_7) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_6, \phi, \{68, 84\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_9, \phi, a_9) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_8, \phi, \{22, 24, 25\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_{10}, \phi, a_{10}) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_9, \phi, \{70, 78, 82\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_{10}, \phi, a_{11}) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_9, \phi, \{88, 92, 93, 95\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_{11}, \phi, x) \leftarrow \texttt{CXOR}(\mathcal{M}, \boldsymbol{b}_{10}, \phi, \{0, 26, 56, 91, 96\}, \phi)$
  $(\mathcal{M}, g) \leftarrow \texttt{xorf}(\mathcal{M}, x, a_1, \ldots, a_{11})$
  return $(\mathcal{M}, \boldsymbol{b}_{11}, g)$
end procedure

3 Details of Grain-V1

3.1 Specification of Grain-V1

Grain-V1 [4] also has a NFSR and a LFSR but the sizes are 80 rather than 128. It has 80 key bits $x_{[80)}$ and 64 IV bits $v_{[64)}$. Full Grin-V1 requires $R = 160$ initialization rounds. The initialization of the NFSR and LFSR are assinged as:

$$\boldsymbol{b}^0 = (b_0, b_1, \ldots, b_{79}) = (x_0, \ldots, x_{79}), \boldsymbol{s}^0 = (s_0, s_1, \ldots, s_{79}) = (v_0, \ldots, v_{63}, 1, \ldots, 1).$$

For $r = 0, \ldots, R-1$, the update function in the initialization round r is as follows.

$$
\begin{aligned}
h^r &\leftarrow s_{r+25} + b_{r+63} + s_{r+3}s_{r+64} + s_{r+46}s_{r+64} + b_{r+63}s_{r+64} + s_{r+3}s_{r+25}s_{r+46} \\
&\quad + s_{r+3}s_{r+46}s_{r+64} + s_{r+3}s_{r+46}b_{r+63} + s_{r+25}s_{r+46}b_{r+63} + s_{r+46}s_{r+64}b_{r+63} \\
z^r &\leftarrow h^r + \sum_{j \in A} b_{r+j} \text{ where } A = \{1, 2, 4, 10, 31, 43, 56\} &&(11) \\
g^r &\leftarrow b_r + b_{r+9} + b_{r+14} + b_{r+21} + b_{r+28} + b_{r+33} + b_{r+37} + b_{r+45} + b_{r+52} + b_{r+60} + b_{r+62} \\
&\quad + b_{r+63}b_{r+60} + b_{r+37}b_{r+33} + b_{r+15}b_{r+9} \\
&\quad + b_{r+60}b_{r+52}b_{r+45} + b_{r+33}b_{r+28}b_{r+21} + b_{r+63}b_{r+45}b_{r+28}b_{r+9} \\
&\quad + b_{r+60}b_{r+52}b_{r+37}b_{r+33} + b_{r+63}b_{r+60}b_{r+21}b_{r+15} \\
&\quad + b_{r+63}b_{r+60}b_{r+52}b_{r+45}b_{r+37} + b_{r+33}b_{r+28}b_{r+21}b_{r+15}b_{r+9} \\
&\quad + b_{r+52}b_{r+45}b_{r+37}b_{r+33}b_{r+28}b_{r+21} \\
b_{r+80} &\leftarrow g^r + z^r + s_r &&(12) \\
f^r &\leftarrow s_r + s_{r+13} + s_{r+23} + s_{r+38} + s_{r+51} + s_{r+62} \\
s_{r+80} &\leftarrow f^r + z^r &&(13) \\
\boldsymbol{b}^r &\leftarrow (b_{r+1}, \ldots b_{r+80}) &&(14) \\
\boldsymbol{s}^r &\leftarrow (s_{r+1}, \ldots s_{r+80}) &&(15)
\end{aligned}
$$

After R initialization rounds, the output function (11) is called for $r = R$ so the 1st output bit is z^R. Of course, full Grain-V1 has $R = 160$ but our secure bounds consider $R < 160$.

3.2 MILP Models of Grain-V1

The initialization DP structures are assigned by calling Algorithm 6. The updating function is defined as Algorithm 7 and its subroutines defined in Algorithm 8.

Algorithm 6 The initial DP structures for Grain-V1.

procedure `IniDP`(MILP model $\mathcal{M}$, cube index set I, non-cube IV assignment $\boldsymbol{IV}$, initialization round number R, split set Λ.)
  Declare the DP structures $\boldsymbol{k}_x, \boldsymbol{k}_v$ both of length 80.
  For $i \in I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 1$ and $\boldsymbol{k}_v[i].F = \delta$.
  For $i \notin I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 0$ and assign the flag value according to $\boldsymbol{IV}[i]$: $\boldsymbol{k}_v[i].F = 1_c$ if $\boldsymbol{IV}[i] = 1$ and $\boldsymbol{k}_v[i].F = 0_c$ if $\boldsymbol{IV}[i] = 0$.
  Set $\mathcal{M}.con \leftarrow \boldsymbol{k}_x[\Lambda].val = 0$ and $\boldsymbol{k}_x[\Lambda].F = 0_c$. Set $\boldsymbol{k}_x[j].F = \delta$ for $j \notin \Lambda$.
  return $(\mathcal{M}, \boldsymbol{k}_x, \boldsymbol{k}_v)$.
end procedure

Algorithm 7 MILP model for Grain-V1 updating function.

procedure `UpdDiv`(the current MILP model $\mathcal{M}$, the DP structure $\boldsymbol{k}^{r-1} = (\boldsymbol{k}_b^{r-1}, \boldsymbol{k}_s^{r-1})$ where $\boldsymbol{k}_b$ and $\boldsymbol{k}_s$ are of length 80 corresponding to the DP structures of $\boldsymbol{b}^{r-1}, \boldsymbol{s}^{r-1}$ respectively, the round number r $(r = 1, 2 \ldots)$.)

$(\mathcal{M}, \boldsymbol{k}'_b, \boldsymbol{k}'_s, z) \leftarrow \texttt{funcZ}(\mathcal{M}, \boldsymbol{k}_b^{r-1}, \boldsymbol{k}_s^{r-1})$

$(\mathcal{M}, z_g, z_f) \leftarrow \texttt{copyf}(\mathcal{M}, z)$

$(\mathcal{M}, \boldsymbol{k}''_b, g) \leftarrow \texttt{funcG}(\mathcal{M}, \boldsymbol{k}'_b)$

$(\mathcal{M}, \boldsymbol{k}''_s, f) \leftarrow \texttt{funcF}(\mathcal{M}, \boldsymbol{k}'_s)$

$(\mathcal{M}, s_0^\star, s_0^{\star\star}) \leftarrow \texttt{copyf}(\mathcal{M}, \boldsymbol{k}''_s[0], 2)$

$(\mathcal{M}, b_{r+79}) \leftarrow \texttt{xorf}(\mathcal{M}, g, s_0^\star, z_g)$

$(\mathcal{M}, s_{r+79}) \leftarrow \texttt{xorf}(\mathcal{M}, f, s_0^{\star\star}, z_f)$

Assign $\boldsymbol{k}_b^r \leftarrow \boldsymbol{k}''_b[1, \ldots, 79] \| b_{r+79}$ and $\boldsymbol{k}_s^r \leftarrow \boldsymbol{k}''_s[1, \ldots, 79] \| s_{r+79}$

Assign $\boldsymbol{k}^r = (\boldsymbol{k}_b^r, \boldsymbol{k}_s^r)$.

end procedure

Algorithm 8 MILP model for NLFSRs in Grain-V1

procedure `funcZ`$(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s})$

$(\mathcal{M}, \boldsymbol{b}_1, \boldsymbol{s}_1, a_1) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s}, \phi, \{3, 64\})$

$(\mathcal{M}, \boldsymbol{b}_2, \boldsymbol{s}_2, a_2) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_1, \boldsymbol{s}_1, \phi, \{46, 64\})$

$(\mathcal{M}, \boldsymbol{b}_3, \boldsymbol{s}_3, a_3) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_2, \boldsymbol{s}_2, \{63\}, \{64\})$

$(\mathcal{M}, \boldsymbol{b}_4, \boldsymbol{s}_4, a_4) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_3, \boldsymbol{s}_3, \phi, \{3, 25, 46\})$

$(\mathcal{M}, \boldsymbol{b}_5, \boldsymbol{s}_5, a_5) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_4, \boldsymbol{s}_4, \phi, \{3, 46, 64\})$

$(\mathcal{M}, \boldsymbol{b}_6, \boldsymbol{s}_6, a_6) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_5, \boldsymbol{s}_5, \{63\}, \{3, 46\})$

$(\mathcal{M}, \boldsymbol{b}_7, \boldsymbol{s}_7, a_7) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_6, \boldsymbol{s}_6, \{63\}, \{25, 46\})$

$(\mathcal{M}, \boldsymbol{b}_8, \boldsymbol{s}_8, a_8) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_7, \boldsymbol{s}_7, \{63\}, \{46, 64\})$

$(\mathcal{M}, \boldsymbol{b}_9, \boldsymbol{s}_9, x) \leftarrow \texttt{CXOR}(\mathcal{M}, \boldsymbol{b}_8, \boldsymbol{s}_8, \{1, 2, 4, 10, 31, 43, 56, 63\}, \{25\})$

$(\mathcal{M}, z) \leftarrow \texttt{xorf}(\mathcal{M}, x, a_1, \ldots, a_8)$

return $(\mathcal{M}, \boldsymbol{b}_9, \boldsymbol{s}_9, z)$

end procedure

---

procedure `funcF`$(\mathcal{M}, \boldsymbol{s})$

$(\mathcal{M}, \phi, \boldsymbol{s}_1, f) \leftarrow \texttt{CXOR}(\mathcal{M}, \phi, \boldsymbol{s}, \phi, \{13, 23, 38, 51, 62\})$

return $(\mathcal{M}, \boldsymbol{s}_1, f)$

end procedure

---

procedure `funcG`$(\mathcal{M}, \boldsymbol{b})$

$(\mathcal{M}, \boldsymbol{b}_1, \phi, a_1) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}, \phi, \{63, 60\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_2, \phi, a_2) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_1, \phi, \{33, 37\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_3, \phi, a_3) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_2, \phi, \{15, 9\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_4, \phi, a_4) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_3, \phi, \{60, 52, 45\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_5, \phi, a_5) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_4, \phi, \{33, 28, 21\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_6, \phi, a_6) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_5, \phi, \{63, 45, 28, 9\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_7, \phi, a_7) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_6, \phi, \{60, 52, 37, 33\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_8, \phi, a_8) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_7, \phi, \{63, 60, 21, 15\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_9, \phi, a_9) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_8, \phi, \{63, 60, 52, 45, 37\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_{10}, \phi, a_{10}) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_9, \phi, \{33, 28, 21, 15, 9\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_{11}, \phi, a_{11}) \leftarrow \texttt{CAND}(\mathcal{M}, \boldsymbol{b}_{10}, \phi, \{52, 45, 37, 33, 28, 21\}, \phi)$

$(\mathcal{M}, \boldsymbol{b}_{12}, \phi, x) \leftarrow \texttt{CXOR}(\mathcal{M}, \boldsymbol{b}_{11}, \phi, \{0, 9, 14, 21, 28, 33, 37, 45, 52, 60, 62\}, \phi)$

$(\mathcal{M}, g) \leftarrow \texttt{xorf}(\mathcal{M}, x, a_1, \ldots, a_{11})$

return $(\mathcal{M}, \boldsymbol{b}_{12}, g)$

end procedure

4 Details of Plantlet

4.1 Specification of Plantlet

Plantlet [5] has 80 key bits $x_{[80)}$ and 90 IV bits $v_{[90)}$. It is composed of a 40-bit NFSR and a LFSR of length 60[1] The NFSR and LFSR are initialized by the key and IV bits respectively as follows:

$$\boldsymbol{b}^0 = (b_0, b_1, \ldots, b_{39}) = (v_0, \ldots, v_{39}), \boldsymbol{s}^0 = (s_0, s_1, \ldots, s_{59}) = (v_{40}, \ldots, v_{89}, 1, \ldots, 1).$$

For initialization round number $r = 0, \ldots, R-1$, the following updating function is called:

$$h^r \leftarrow b_{r+4l} s_{r+6} + s_{r+8l} s_{r+10} + s_{t+17} s_{r+32} + s_{r+19} s_{r+23} + b_{r+4} s_{r+32} b_{r+38}$$

$$z^r \leftarrow h^r + s_{r+30} + \sum_{j \in A} b_{r+j} \text{ where } A = \{1, 6, 15, 17, 23, 28, 34\} \tag{16}$$

$$\begin{aligned} g^r \leftarrow\ & b_r + b_{r+13} + b_{r+19} + b_{r+35} + b_{r+39} + b_{r+2} b_{r+25} + b_{r+3} b_{r+5} + b_{r+7} b_{r+8} \\ & + b_{r+14} b_{r+21} + b_{r+16} b_{r+18} + b_{r+22} b_{r+24} + b_{r+26} b_{r+32} + b_{r+33} b_{r+36} b_{r+37} b_{r+38} \\ & + b_{r+10} b_{r+11} b_{r+12} + b_{r+27} b_{r+30} b_{r+31} \end{aligned}$$

$$b_{r+40} \leftarrow g^r + z^r + s_r + \boldsymbol{K}[r] + c_4^r \tag{17}$$

$$f^r \leftarrow s_r + s_{r+14} + s_{r+20} + s_{r+34} + s_{r+43} + s_{r+54}$$

$$s_{r+60} \leftarrow f^r + z^r \tag{18}$$

$$\boldsymbol{b}^r \leftarrow (b_{r+1}, \ldots b_{r+40}) \tag{19}$$

$$\boldsymbol{s}^r \leftarrow (s_{r+1}, \ldots s_{r+60}) \tag{20}$$

Note that in (17), we denote $\boldsymbol{K}$ as a vector of length R and its ith entry is defined as $\boldsymbol{K}[i] \leftarrow x_{i \mod 80}$. The number c_4^r is the 4th bit of a counter and it can be determined purely by r as

$$c_4^r = \lfloor \frac{r \mod 80}{2^4} \rfloor \tag{21}$$

4.2 MILP Models for Plantlet

It is noticeable that the updating function of Plantlet requires $\boldsymbol{K}$. Therefore, in `IniDP`, we should determine the DP structures of both IV bits and $\boldsymbol{K}$. Such a process is described as Algorithm 9. The updating function can be defined as Algorithm 10 and its subroutines are in Algorithm 11.

5 Details of Kreyvium

5.1 Specification of Kreyvium

Kreyvium is designed for the use of fully Homomorphic encryption [6]. It claims 128-bit security and accepts 128-bit IV. Kreyvium consists of 5 registers. Two

[1] Only during the initialiation phase. Afterwards, the length of LFSR becomes 61 [5].

Algorithm 9 The initial DP structures for Kreyvium.

procedure `IniDP`(MILP model $\mathcal{M}$, cube index set I, non-cube IV assignment $\boldsymbol{IV}$, initialization round number R ($R \geq 80$), split set Λ.)
    Declare the DP structures $\boldsymbol{k}_x, \boldsymbol{k}_v$ both of lengths 80 and 90 respectively.
    For $i \in I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 1$ and $\boldsymbol{k}_v[i].F = \delta$.
    For $i \notin I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 0$ and assign the flag value according to $\boldsymbol{IV}[i]$: $\boldsymbol{k}_v[i].F = 1_c$ if $\boldsymbol{IV}[i] = 1$ and $\boldsymbol{k}_v[i].F = 0_c$ if $\boldsymbol{IV}[i] = 0$.
    Set $\mathcal{M}.con \leftarrow \boldsymbol{k}_x[\Lambda].val = 0$ and $\boldsymbol{k}_x[\Lambda].F = 0_c$. Set $\boldsymbol{k}_x[j].F = \delta$ for $j \notin \Lambda$.
    Initialize $\boldsymbol{k}_K \leftarrow k_x$.
    for $r = 80, \ldots, R-1$ **do**
        $(\mathcal{M}, t_1, t_2) \leftarrow$ `copyf`$(\mathcal{M}, \boldsymbol{k}_K[r \mod 80], 2)$.
        Update $\boldsymbol{k}_K[r \mod 80] \leftarrow t_1$ and $\boldsymbol{k}_K \leftarrow \boldsymbol{k}_K \| t_2$.
    end for
    return $(\mathcal{M}, \boldsymbol{k}_v, \boldsymbol{k}_K)$.
end procedure

Algorithm 10 MILP model for Plantlet updating function.

procedure `UpdDiv`(the current MILP model $\mathcal{M}$, the DP structure $\boldsymbol{k}^{r-1} = (\boldsymbol{k}_b^{r-1}, \boldsymbol{k}_s^{r-1})$ where $\boldsymbol{k}_b$ and $\boldsymbol{k}_s$ are of lengths 40 and 60 corresponding to the DP structures of $\boldsymbol{b}^{r-1}, \boldsymbol{s}^{r-1}$ respectively, the round number r ($r = 1, 2 \ldots$), the DP structure $\boldsymbol{k}_K$ corresponding to the division property of $\boldsymbol{K}$.)
    $(\mathcal{M}, \boldsymbol{k}_b', \boldsymbol{k}_s', z) \leftarrow$ `funcZ`$(\mathcal{M}, \boldsymbol{k}_b^{r-1}, \boldsymbol{k}_s^{r-1})$
    $(\mathcal{M}, z_g, z_f) \leftarrow$ `copyf`$(\mathcal{M}, z)$
    $(\mathcal{M}, \boldsymbol{k}_b'', g) \leftarrow$ `funcG`$(\mathcal{M}, \boldsymbol{k}_b')$
    $(\mathcal{M}, \boldsymbol{k}_s'', f) \leftarrow$ `funcF`$(\mathcal{M}, \boldsymbol{k}_s')$
    $(\mathcal{M}, s_0^\star, s_0^{\star\star}) \leftarrow$ `copyf`$(\mathcal{M}, \boldsymbol{k}_s''[0], 2)$
    Compute c_4^{r-1} according to (21) and declare a DP structure o with constraint $\mathcal{M}.con \leftarrow o.val = 0$.
    If $c_4^{r-1} = 0$, set the flag value $o.F = 0_c$; otherwise, $o.F = 1_c$.
    $(\mathcal{M}, b_{r+39}) \leftarrow$ `xorf`$(\mathcal{M}, g, s_0^\star, z_g, \boldsymbol{k}_K[r-1], c_4^{r-1})$
    $(\mathcal{M}, s_{r+59}) \leftarrow$ `xorf`$(\mathcal{M}, f, s_0^{\star\star}, z_f)$
    Assign $\boldsymbol{k}_b^r \leftarrow \boldsymbol{k}_b''[1, \ldots, 39] \| b_{r+39}$ and $\boldsymbol{k}_s^r \leftarrow \boldsymbol{k}_s''[1, \ldots, 59] \| s_{r+59}$
    Assign $\boldsymbol{k}^r = (\boldsymbol{k}_b^r, \boldsymbol{k}_s^r)$.
end procedure

of them are LFSRs denoted as $\boldsymbol{K}$ and $\boldsymbol{V}$ respectively. The remaining is three concatenated NFSRs making up a 288-bit state $\boldsymbol{s}$ denoted as

$$(s_0, \ldots, s_{92}) \| (s_{93}, \ldots, s_{176}) \| (s_{177}, \ldots, s_{287})$$

The registers are initialized with the 128 key bits, $x_{[127)}$, and 128 IV bits, $v_{[127)}$ as follows:

$$\begin{aligned}
\boldsymbol{s}^0[0, \ldots, 92] &= (s_0^0, \ldots, s_{93}^0) \leftarrow (x_0, \ldots, x_{92}), \\
\boldsymbol{s}^0[93, \ldots, 176] &= (s_{93}^0, \ldots, s_{176}^0) \leftarrow (v_0, \ldots, v_{83}), \\
\boldsymbol{s}^0[177, \ldots, 287] &= (s_{177}^0, \ldots, s_{287}^0) \leftarrow (v_{84}, \ldots, v_{127}, 1, 1, \ldots, 1, 0), \\
\boldsymbol{V}^0 &= (V_{127}^0, V_{126}^0, \ldots, V_0^0) \leftarrow (v_0, \ldots, v_{127}), \\
\boldsymbol{K}^0 &= (K_{127}^0, K_{126}^0, \ldots, K_0^0) \leftarrow (x_0, \ldots, x_{127}),
\end{aligned}$$

For initialization round R, the updating function is called as $(\boldsymbol{s}^r, \boldsymbol{V}^r, \boldsymbol{K}^r) \leftarrow$ `Upd`$(\boldsymbol{s}^{r-1}, \boldsymbol{V}^{r-1}, \boldsymbol{K}^{r-1})$ for $r = 1, \ldots R$. The procedure of `Upd` can be depicted

Algorithm 11 MILP model for NLFSRs in Plantlet

procedure funcZ($\mathcal{M}, \boldsymbol{b}, \boldsymbol{s}$)
  $(\mathcal{M}, \boldsymbol{b}_1, \boldsymbol{s}_1, a_1) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}, \boldsymbol{s}, \{4\}, \{6\})$
  $(\mathcal{M}, \boldsymbol{b}_2, \boldsymbol{s}_2, a_2) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_1, \boldsymbol{s}_1, \phi, \{8, 10\})$
  $(\mathcal{M}, \boldsymbol{b}_3, \boldsymbol{s}_3, a_3) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_2, \boldsymbol{s}_2, \phi, \{17, 32\})$
  $(\mathcal{M}, \boldsymbol{b}_4, \boldsymbol{s}_4, a_4) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_3, \boldsymbol{s}_3, \phi, \{19, 23\})$
  $(\mathcal{M}, \boldsymbol{b}_5, \boldsymbol{s}_5, a_5) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_4, \boldsymbol{s}_4, \{4, 38\}, \{32\})$
  $(\mathcal{M}, \boldsymbol{b}_6, \boldsymbol{s}_6, x) \leftarrow$ CXOR$(\mathcal{M}, \boldsymbol{b}_5, \boldsymbol{s}_5, \{1, 6, 15, 17, 23, 28, 34\}, \{30\})$
  $(\mathcal{M}, z) \leftarrow$ xorf$(\mathcal{M}, x, a_1, \ldots, a_5)$
  return $(\mathcal{M}, \boldsymbol{b}_6, \boldsymbol{s}_6, z)$
end procedure

procedure funcF($\mathcal{M}, \boldsymbol{s}$)
  $(\mathcal{M}, \phi, \boldsymbol{s}_1, f) \leftarrow$ CXOR$(\mathcal{M}, \phi, \boldsymbol{s}, \phi, \{14, 20, 34, 43, 54\})$
  return $(\mathcal{M}, \boldsymbol{s}_1, f)$
end procedure

procedure funcG($\mathcal{M}, \boldsymbol{b}$)
  $(\mathcal{M}, \boldsymbol{b}_1, \phi, a_1) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}, \phi, \{2, 25\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_2, \phi, a_2) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_1, \phi, \{3, 5\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_3, \phi, a_3) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_2, \phi, \{7, 8\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_4, \phi, a_4) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_3, \phi, \{14, 21\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_5, \phi, a_5) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_4, \phi, \{16, 18\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_6, \phi, a_6) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_5, \phi, \{22, 24\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_7, \phi, a_7) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_6, \phi, \{26, 32\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_8, \phi, a_8) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_7, \phi, \{33, 36, 37, 38\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_9, \phi, a_9) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_8, \phi, \{10, 11, 12\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_{10}, \phi, a_{10}) \leftarrow$ CAND$(\mathcal{M}, \boldsymbol{b}_9, \phi, \{27, 30, 31\}, \phi)$
  $(\mathcal{M}, \boldsymbol{b}_{11}, \phi, x) \leftarrow$ CXOR$(\mathcal{M}, \boldsymbol{b}_{10}, \phi, \{0, 13, 19, 35, 39\}, \phi)$
  $(\mathcal{M}, g) \leftarrow$ xorf$(\mathcal{M}, x, a_1, \ldots, a_{10})$
  return $(\mathcal{M}, \boldsymbol{b}_{11}, g)$
end procedure

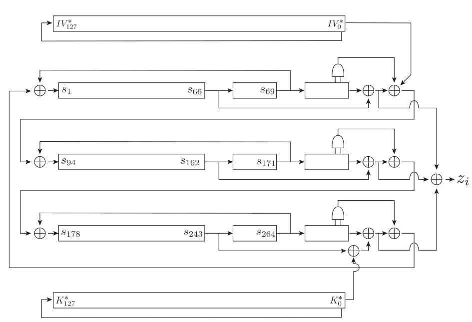

Fig. 2. Structure of Kreyvium

as Fig. 2 defined as follows:

$$
\begin{aligned}
&t_1^{r-1} \leftarrow s_{65}^{r-1} \oplus s_{92}^{r-1}\\
&t_2^{r-1} \leftarrow s_{161}^{r-1} \oplus s_{176}^{r-1}\\
&t_3^{r-1} \leftarrow s_{242}^{r-1} \oplus s_{287}^{r-1} \oplus K_0^{r-1}\\
&z^{r-1} \leftarrow t_1^{r-1} \oplus t_2^{r-1} \oplus t_3^{r-1}\\
&t_1^{r-1} \leftarrow t_1^{r-1} \oplus s_{90}^{r-1} \cdot s_{91}^{r-1} \oplus s_{170}^{r-1} \oplus IV_0^{r-1}\\
&t_2^{r-1} \leftarrow t_2^{r-1} \oplus s_{174}^{r-1} \cdot s_{175}^{r-1} \oplus s_{263}^{r-1}\\
&t_3^{r-1} \leftarrow t_3^{r-1} \oplus s_{285}^{r-1} \cdot s_{286}^{r-1} \oplus s_{68}^{r-1}\\
&\boldsymbol{s}^r[0,\dots,92] = (s_0^r,\dots,s_{92}^r) \leftarrow (t_3^{r-1}, s_0^{r-1},\dots,s_{91}^{r-1})\\
&\boldsymbol{s}^r[93,\dots,176] = (s_{93}^r,\dots,s_{176}^r) \leftarrow (t_1^{r-1}, s_{93}^{r-1},\dots,s_{175}^{r-1})\\
&\boldsymbol{s}^r[177,\dots,287] = (s_{177}^r,\dots,s_{287}^r) \leftarrow (t_2^{r-1}, s_{177}^{r-1},\dots,s_{286}^{r-1})\\
&\boldsymbol{K}^r = (K_{127}^r, K_{126}^r,\dots,K_0^r) \leftarrow (K_0^{r-1}, K_{127}^{r-1}, K_{126}^{r-1},\dots,K_1^{r-1})\\
&\boldsymbol{V}^r = (V_{127}^r, V_{126}^r,\dots,V_0^r) \leftarrow (V_0^{r-1}, V_{127}^{r-1}, V_{126}^{r-1},\dots,V_1^{r-1})
\end{aligned}
$$

After R initialization rounds, the output keystream is output as $z^R, z^{R+1}, \dots$. According to [6], full Kreyvium requires $R = 1152$ initialization rounds.

5.2 MILP Model of Kreyvium

The initialization division property assignments for key and IV bits can be assigned by calling Algorithm 12. The division property propagation corresponding to Kreyvium updating function can be described by the MILP model generated with Algorithm 13. Both algorithms have already been given in [3].

Algorithm 12 The initial DP structures for Kreyvium.

1: **procedure** `IniDP`(MILP model $\mathcal{M}$, cube index set I, non-cube IV assignment $\boldsymbol{IV}$, initialization round number R, split set Λ.)
2: Declare the DP structures $\boldsymbol{k}_x, \boldsymbol{k}_v$ both of length 128.
3: For $i \in I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 1$ and $\boldsymbol{k}_v[i].F = \delta$.
4: For $i \notin I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 0$ and assign the flag value according to $\boldsymbol{IV}[i]$: $\boldsymbol{k}_v[i].F = 1_c$ if $\boldsymbol{IV}[i] = 1$ and $\boldsymbol{k}_v[i].F = 0_c$ if $\boldsymbol{IV}[i] = 0$.
5: Set $\mathcal{M}.con \leftarrow \boldsymbol{k}_x[\Lambda].val = 0$ and $\boldsymbol{k}_x[\Lambda].F = 0_c$. Set $\boldsymbol{k}_x[j].F = \delta$ for $j \notin \Lambda$.
6: For $j = 0,\dots,92$, $(\mathcal{M}, \boldsymbol{k}_K^0[127-j], \boldsymbol{k}_s^0[j]) \leftarrow \texttt{copyf}(\mathcal{M}, \boldsymbol{k}_x[j], 2)$.
7: For $j = 93,\dots,127$, $\boldsymbol{k}_K^0[127-j] \leftarrow \boldsymbol{k}_x[j]$.
8: For $j = 0,\dots,127$, $(\mathcal{M}, \boldsymbol{k}_V^0[127-j], \boldsymbol{k}_s^0[93+j]) \leftarrow \texttt{copyf}(\mathcal{M}, \boldsymbol{k}_v[j], 2)$.
9: For $j = 221,\dots,287$, $\mathcal{M}.con \leftarrow \boldsymbol{k}_s^0[j].val = 0$.
10: For $j = 221,\dots,286$, $\boldsymbol{k}_s^0[j].F = 1_c$ and $\boldsymbol{k}_s^0[287].F = 0_c$.
11: **return** $(\mathcal{M}, \boldsymbol{k}^0)$ where $\boldsymbol{k}^0 = (\boldsymbol{k}_s^0, \boldsymbol{k}_V^0, \boldsymbol{k}_K^0)$.
12: **end procedure**

Algorithm 13 MILP model description to the division property propagation for Kreyvium updating function.

procedure $\texttt{UpdDiv}$(the current MILP model $\mathcal{M}$, the DP structure $\boldsymbol{k}^{r-1} = (\boldsymbol{k}_s^{r-1}, \boldsymbol{k}_V^{r-1}, \boldsymbol{k}_K^{r-1})$ where $\boldsymbol{k}_s$, $\boldsymbol{k}_K$ and $\boldsymbol{k}_V$ are of lengths 288, 128, 128 corresponding to the DP structures of $\boldsymbol{s}^{r-1}$, $\boldsymbol{K}^{r-1}$ and $\boldsymbol{V}^{r-1}$ respectively, the round number r $(r = 1, 2, \ldots)$)
    $(\mathcal{M}, \boldsymbol{x}) \leftarrow \texttt{Core}(\mathcal{M}, \boldsymbol{k}_s^{r-1}, 90, 91, 65, 170, 92)$
    $(\mathcal{M}, \boldsymbol{k}_V^r, v^*) \leftarrow \texttt{LFSR}(\mathcal{M}, \boldsymbol{k}_V^{r-1})$
    $(\mathcal{M}, t_1) \leftarrow \texttt{xorf}(\mathcal{M}, v^*, x_{92})$
    $x_{92} \leftarrow t_1$ ▷ Update the 93rd entry of $\boldsymbol{x}$
    $(\mathcal{M}, \boldsymbol{y}) \leftarrow \texttt{Core}(\mathcal{M}, \boldsymbol{x}, 174, 175, 161, 263, 176)$
    $(\mathcal{M}, \boldsymbol{z}) \leftarrow \texttt{Core}(\mathcal{M}, \boldsymbol{y}, 285, 286, 242, 68, 287)$
    $(\mathcal{M}, \boldsymbol{k}_K^r, k^*) \leftarrow \texttt{LFSR}(\mathcal{M}, \boldsymbol{k}_K^{r-1})$
    $(\mathcal{M}, t_3) \leftarrow \texttt{xorf}(\mathcal{M}, k^*, z_{287})$
    $z_{288} \leftarrow t_3$ ▷ Update the 288th entry of $\boldsymbol{z}$
    $\boldsymbol{k}_s^r = \boldsymbol{z} \ggg 1$
    return $\mathcal{M}$ and $\boldsymbol{k}^r = (\boldsymbol{k}_s^r, \boldsymbol{k}_V^r, \boldsymbol{k}_K^r)$.
end procedure

procedure $\texttt{LFSR}(\mathcal{M}, \boldsymbol{x})$
    $(\mathcal{M}, a, o) \leftarrow \texttt{copyf}(\mathcal{M}, x_0, 2)$
    for all $i \in \{0, 1, \ldots, 126\}$ **do**
        $y_i = x_{i+1}$
    end for
    $y_{127} = a$
    return $(\mathcal{M}, \boldsymbol{y}, o)$
end procedure

procedure $\texttt{Core}(\mathcal{M}, \boldsymbol{x}, i_1, i_2, i_3, i_4, i_5)$
    $(\mathcal{M}, y_{i_1}, y'_{i_1}) \leftarrow \texttt{copyf}(\mathcal{M}, x_{i_1}, 2)$.
    $(\mathcal{M}, y_{i_2}, y'_{i_2}) \leftarrow \texttt{copyf}(\mathcal{M}, x_{i_2}, 2)$.
    $(\mathcal{M}, z_1) \leftarrow \texttt{andf}(\mathcal{M}, y'_{i_1}, y'_{i_2})$.
    $(\mathcal{M}, y_{i_3}, y'_{i_3}) \leftarrow \texttt{copyf}(\mathcal{M}, x_{i_3}, 2)$.
    $(\mathcal{M}, y_{i_4}, y'_{i_4}) \leftarrow \texttt{copyf}(\mathcal{M}, x_{i_4}, 2)$.
    $(\mathcal{M}, y_{i_5}) \leftarrow \texttt{xorf}(\mathcal{M}, z_1, y'_{i_3}, y'_{i_4}, x_{i_5})$.
    for all $i \in [288] \backslash \{i_1, i_2, i_3, i_4, i_5\}$ **do**
        $y_i = x_i$
    end for
    return $(\mathcal{M}, \boldsymbol{y})$
end procedure

6 Details of Acorn

6.1 Specification of Acorn

Acorn is an authenticated encryption algorithm and is one of the finalists in CAESAR competition [7]. The structure is based on NLFSR, and the internal state is represented by a 293-bit state $\boldsymbol{s} = (s_0, \dots, s_{292})$. There are two component functions, $ks = KSG128(\boldsymbol{s})$ and $f = FBK128(\boldsymbol{s})$, in the update function, and each is defined as

$$
\begin{aligned}
ks &= s_{12} \oplus s_{154} \oplus maj(s_{235}, s_{61}, s_{193}) \oplus ch(s_{230}, s_{111}, s_{66}),\\
f &= s_0 \oplus (s_{107} \oplus 1) \oplus maj(s_{244}, s_{23}, s_{160}) \oplus (ca \wedge s_{196}) \oplus (cb \wedge ks),
\end{aligned}
$$

where ks is used as the key stream, and maj and ch are defined as

$$
\begin{aligned}
maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z),\\
ch(x, y, z) &= (x \wedge y) \oplus ((x \oplus 1) \wedge z).
\end{aligned}
$$

Initialized as $\boldsymbol{s}^0 = \boldsymbol{0}$, the following updating function is called for round number $r = 1, \dots, R$:

$$s^{r-1}_{289} \leftarrow s^{r-1}_{289} \oplus s^{r-1}_{235} \oplus s^{r-1}_{230} \tag{22}$$
$$s^{r-1}_{230} \leftarrow s^{r-1}_{230} \oplus s^{r-1}_{196} \oplus s^{r-1}_{193} \tag{23}$$
$$s^{r-1}_{193} \leftarrow s^{r-1}_{193} \oplus s^{r-1}_{160} \oplus s^{r-1}_{154} \tag{24}$$
$$s^{r-1}_{154} \leftarrow s^{r-1}_{154} \oplus s^{r-1}_{111} \oplus s^{r-1}_{107} \tag{25}$$
$$s^{r-1}_{107} \leftarrow s^{r-1}_{107} \oplus s^{r-1}_{66} \oplus s^{r-1}_{61} \tag{26}$$
$$s^{r-1}_{61} \leftarrow s^{r-1}_{61} \oplus s^{r-1}_{23} \oplus s^{r-1}_{0} \tag{27}$$
$$ks^{r-1} = KSG128(\boldsymbol{s}^{r-1}) \tag{28}$$
$$f^{r-1} = FBK128(\boldsymbol{s}^{r-1}, ca, cb) \tag{29}$$
$$\boldsymbol{s}^r = (s^r_0, s^r_1, \dots, s^r_{292}) \leftarrow (s^{r-1}_1, s^{r-1}_2, \dots, S^{r-1}_{292}, f^{r-1} \oplus \boldsymbol{m}[r-1]) \tag{30}$$

The full Acorn requires $R = 1792$. The vector $\boldsymbol{m}$ is of length R:

- The first 256 entries are assigned as $\boldsymbol{m}[j] = x_j$ and $\boldsymbol{m}[128 + j] = v_j$ for $j \in [128)$.
- For $r \geq 256$: if $128|r$, $\boldsymbol{m}[r] = x_{r \mod 128} + 1$; otherwise, $\boldsymbol{m}[r] = x_{r \mod 128}$.

After the R initialization rounds, the 1st output keystream bit is simply ks^r generated by the process from (22) to (28). Of course, the associated data is always loaded before the output of the key stream. But in our attack, the initialization round number R is smaller than 1792 so we do not consider the associate data. This is the same setting with [8,3]. Fig. 3 shows the structure of Acorn and more detailed specification of Acorn can be found in [9].

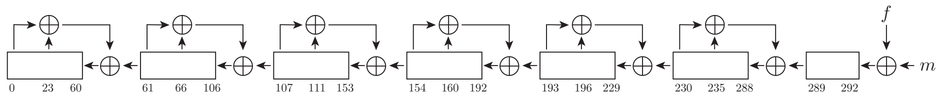

Fig. 3. Structure of ACORN

6.2 MILP Model of ACORN

According to the description in Section 6.1, the MILP model describing the division property propagation of ACORN updating function can be constructed as Algorithm 15. The subroutines are described in detail: as Algorithm 16, 17 and 18. `xorFB` is called for the LFSR updating (22) (Algorithm 17); `ksg128` and `fbk128` corresponds to $KSG128$ and $FBK128$ respectively (Algorithm 18); maj and ch are also handled (Algorithm 16). Since $\boldsymbol{s}^0 = \boldsymbol{0}$, $\boldsymbol{k}_s^0$ can be defined without the DP structures of key and IV bits. But, in addition to the division property of the internal state $\boldsymbol{k}_s^{r-1}$, Algorithm 15 also requires $\boldsymbol{k}_m$, the division property of $\boldsymbol{m}$. Such $\boldsymbol{k}_m$ is determined at the very beginning of model construction as `IniDP` in Algorithm 14 This is identical to the MILP descriptions in [3].

Algorithm 14 The initial DP structures for ACORN.

procedure `IniDP`(MILP model $\mathcal{M}$, cube index set I, non-cube IV assignment $\boldsymbol{IV}$, initialization round number R, split set Λ.)
  Declare the DP structures $\boldsymbol{k}_x, \boldsymbol{k}_v$ both of length 128.
  For $i \in I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 1$ and $\boldsymbol{k}_v[i].F = \delta$.
  For $i \notin I$, set $\mathcal{M}.con \leftarrow \boldsymbol{k}_v[i].val = 0$ and assign the flag value according to $\boldsymbol{IV}[i]$: $\boldsymbol{k}_v[i].F = 1_c$ if $\boldsymbol{IV}[i] = 1$ and $\boldsymbol{k}_v[i].F = 0_c$ if $\boldsymbol{IV}[i] = 0$.
  Set $\mathcal{M}.con \leftarrow \boldsymbol{k}_x[\Lambda].val = 0$ and $\boldsymbol{k}_x[\Lambda].F = 0_c$. Set $\boldsymbol{k}_x[j].F = \delta$ for $j \notin \Lambda$.
  For $j \in [128)$, set $\boldsymbol{k}_m[j] \leftarrow k_x[j]$ and $\boldsymbol{k}_m[128 + j] \leftarrow k_v[j]$.
  for $r = 256, \ldots, R - 1$ **do**
    $(\mathcal{M}, t_1, t_2) \leftarrow$ `copyf`$(\mathcal{M}, \boldsymbol{k}_m[r \mod 128], 2)$.
    if $r \mod 128 = 0$ **then**
      Declare DP structure o as $\mathcal{M}.con \leftarrow o.val = 0$ and $o.F = 1_c$.
      Compute $(\mathcal{M}, t_3) \leftarrow$ `xorf`$(\mathcal{M}, o, t_2)$.
      Update $\boldsymbol{k}_m[r \mod 128] \leftarrow t_1$ and $\boldsymbol{k}_m \leftarrow \boldsymbol{k}_m \| t_3$.
    else
      Update $\boldsymbol{k}_m[r \mod 128] \leftarrow t_1$ and $\boldsymbol{k}_m \leftarrow \boldsymbol{k}_m \| t_2$.
    end if
  end for
  return $(\mathcal{M}, \boldsymbol{k}_m)$.
end procedure

References

1. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: IEEE International Symposium on Information Theory. (2006)
2. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of grain-128 with optional authentication. IJWMC **5**(1) (2011) 48–59

Algorithm 15 MILP model for ACORN updating function

procedure `UpdDiv`(The current MILP model $\mathcal{M}$, the DP structures for $\boldsymbol{s}^{r-1}$ and $\boldsymbol{m}$ denoted as $\boldsymbol{k}_s^{r-1}$ and $\boldsymbol{k}_m$ respectively, round number r ($r = 1, \ldots, R$))
    $(\mathcal{M}, \boldsymbol{T}) \leftarrow \texttt{xorFB}(\mathcal{M}, \boldsymbol{k}_s^{r-1}, 289, 235, 230)$
    $(\mathcal{M}, \boldsymbol{U}) \leftarrow \texttt{xorFB}(\mathcal{M}, \boldsymbol{T}, 230, 196, 193)$
    $(\mathcal{M}, \boldsymbol{V}) \leftarrow \texttt{xorFB}(\mathcal{M}, \boldsymbol{U}, 193, 160, 154)$
    $(\mathcal{M}, \boldsymbol{W}) \leftarrow \texttt{xorFB}(\mathcal{M}, \boldsymbol{V}, 154, 111, 107)$
    $(\mathcal{M}, \boldsymbol{X}) \leftarrow \texttt{xorFB}(\mathcal{M}, \boldsymbol{W}, 107, 66, 61)$
    $(\mathcal{M}, \boldsymbol{Y}) \leftarrow \texttt{xorFB}(\mathcal{M}, \boldsymbol{X}, 61, 23, 0)$
    $(\mathcal{M}, \boldsymbol{Z}, ks) \leftarrow \texttt{ksg128}(\mathcal{M}, \boldsymbol{Y})$
    $(\mathcal{M}, \boldsymbol{A}, f) \leftarrow \texttt{fbk128}(\mathcal{M}, \boldsymbol{Z}, ks)$
    for $i = 0$ to 291 **do**
        $\boldsymbol{k}_s^r[i] = A_{i+1}$
    end for
    $(\mathcal{M}, \boldsymbol{k}_s^r[292]) \leftarrow \texttt{xorf}(\mathcal{M}, ks, f, \boldsymbol{m}[r-1])$.
    return $(\mathcal{M}, \boldsymbol{k}_s^r)$.
end procedure

3. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. In Shacham, H., Boldyreva, A., eds.: CRYPTO 2018, Part I. Volume 10991 of LNCS. (2018) 275–305
4. Hell, M., Johansson, T., Meier, W.: Grain: A stream cipher for constrained environments. IJWMC **2** (01 2007) 86–93
5. Mikhalev, V., Armknecht, F., Müller, C.: On ciphers that continuously access the non-volatile key. IACR Trans. Symmetric Cryptol. **2016** (2016) 52–79
6. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In Peyrin, T., ed.: FSE. Volume 9783 of LNCS., Springer (2016) 313–333
7. : CAESAR: Competition for authenticated encryption: Security, applicability, and robustness (2014)
8. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In Katz, J., Shacham, H., eds.: CRYPTO Part III. Volume 10403 of LNCS., Springer (2017) 250–279
9. Wu, H.: Acorn v3 (2016) Submission to CAESAR competition.

Algorithm 16 MILP model for *maj* and *ch* in ACORN

procedure maj($\mathcal{M}, \boldsymbol{X}, i, j, k$)
    if $X_i.F \oplus X_j.F = 0_c$ **then**
        $(\mathcal{M}, Y_i, a) \leftarrow \texttt{copyf}(\mathcal{M}, X_i)$
        $(\mathcal{M}, Y_j, b) \leftarrow \texttt{copyf}(\mathcal{M}, X_j)$
        $(\mathcal{M}, o) \leftarrow \texttt{andf}(\mathcal{M}, a, b)$
        $Y_s = X_s$ for all $s \in \{0, \dots, 292\} - \{i, j\}$
    else if $X_i.F \oplus X_k.F = 0_c$ **then**
        $(\mathcal{M}, Y_i, a) \leftarrow \texttt{copyf}(\mathcal{M}, X_i)$
        $(\mathcal{M}, Y_k, c) \leftarrow \texttt{copyf}(\mathcal{M}, X_k)$
        $(\mathcal{M}, o) \leftarrow \texttt{andf}(\mathcal{M}, a, c)$
        $Y_s = X_s$ for all $s \in \{0, \dots, 292\} - \{i, k\}$
    else if $X_j.F \oplus X_k.F = 0_c$ **then**
        $(\mathcal{M}, Y_j, b) \leftarrow \texttt{copyf}(\mathcal{M}, X_j)$
        $(\mathcal{M}, Y_k, c) \leftarrow \texttt{copyf}(\mathcal{M}, X_k)$
        $(\mathcal{M}, o) \leftarrow \texttt{andf}(\mathcal{M}, b, c)$
        $Y_s = X_s$ for all $s \in \{0, \dots, 292\} - \{j, k\}$
    else
        $(\mathcal{M}, Y_j, a_1, a_2) \leftarrow \texttt{copyf}(\mathcal{M}, X_i)$
        $(\mathcal{M}, Y_j, b_1, b_2) \leftarrow \texttt{copyf}(\mathcal{M}, X_j)$
        $(\mathcal{M}, Y_k, c_1, c_2) \leftarrow \texttt{copyf}(\mathcal{M}, X_k)$
        $(\mathcal{M}, a) \leftarrow \texttt{andf}(\mathcal{M}, a_1, b_1)$
        $(\mathcal{M}, b) \leftarrow \texttt{andf}(\mathcal{M}, a_2, c_1)$
        $(\mathcal{M}, c) \leftarrow \texttt{andf}(\mathcal{M}, b_2, c_2)$
        $(\mathcal{M}, o) \leftarrow \texttt{xorf}(\mathcal{M}, a, b, c)$
        $Y_s = X_s$ for all $s \in \{0, \dots, 292\} - \{i, j, k\}$
    end if
    return $(\mathcal{M}, \boldsymbol{Y}, o)$
end procedure

procedure ch($\mathcal{M}, \boldsymbol{X}, i, j, k$)
    if $X_i.F = 0_c$ or $X_j.F \oplus X_k.F = 0_c$ **then**
        $(\mathcal{M}, Y_k, o) \leftarrow \texttt{copyf}(\mathcal{M}, X_k)$
        $Y_s = X_s$ for all $s \in \{0, \dots, 292\} - \{k\}$
    else if $X_i.F = 1_c$ **then**
        $(\mathcal{M}, Y_j, o) \leftarrow \texttt{copyf}(\mathcal{M}, X_j)$
        $Y_s = X_s$ for all $s \in \{0, \dots, 292\} - \{j\}$
    else
        $(\mathcal{M}, Y_j, a_1, a_2) \leftarrow \texttt{copyf}(\mathcal{M}, X_i)$
        $(\mathcal{M}, Y_j, b_1) \leftarrow \texttt{copyf}(\mathcal{M}, X_j)$
        $(\mathcal{M}, Y_k, c, c_1) \leftarrow \texttt{copyf}(\mathcal{M}, X_k)$
        $(\mathcal{M}, a) \leftarrow \texttt{andf}(\mathcal{M}, a_1, b_1)$
        $(\mathcal{M}, b) \leftarrow \texttt{andf}(\mathcal{M}, a_2, c_1)$
        $(\mathcal{M}, o) \leftarrow \texttt{xorf}(\mathcal{M}, a, b, c)$
        $Y_s = X_s$ for all $s \in \{0, \dots, 292\} - \{i, j, k\}$
    end if
    return $(\mathcal{M}, \boldsymbol{Y}, o)$
end procedure

Algorithm 17 MILP model for LFSR in ACORN

procedure xorFB($\mathcal{M}, \boldsymbol{X}, i, j, k$)
    $(\mathcal{M}, Y_j, a) \leftarrow \texttt{copyf}(\mathcal{M}, X_j)$
    $(\mathcal{M}, Y_k, b) \leftarrow \texttt{copyf}(\mathcal{M}, X_k)$
    $(\mathcal{M}, Y_i) \leftarrow \texttt{xorf}(\mathcal{M}, a, b, X_i)$
    $Y_s = X_s$ for all $s \in \{1, \dots, 293\} - \{i, j, k\}$
    return $(\mathcal{M}, \boldsymbol{Y})$
end procedure

Algorithm 18 MILP model for `ksg128` and `fbk128` in ACORN

procedure `ksg128`$(\mathcal{M}, \boldsymbol{X})$
    $(\mathcal{M}, A_{12}, x_1) \leftarrow \texttt{copyf}(\mathcal{M}, X_{12})$
    $(\mathcal{M}, A_{154}, x_2) \leftarrow \texttt{copyf}(\mathcal{M}, X_{154})$
    $A_t = X_t$ for all $t \in \{0, \ldots, 292\} - \{12, 154\}$
    $(\mathcal{M}, \boldsymbol{B}, x_3) \leftarrow \texttt{maj}(\mathcal{M}, \boldsymbol{A}, 235, 61, 193)$
    $(\mathcal{M}, \boldsymbol{Y}, x_4) \leftarrow \texttt{ch}(\mathcal{M}, \boldsymbol{B}, 230, 111, 66)$
    $(\mathcal{M}, z) \leftarrow \texttt{xorf}(\mathcal{M}, x_1, x_2, x_3, x_4)$
    return $(\mathcal{M}, \boldsymbol{Y}, z)$
end procedure

procedure `fbk128`$(\mathcal{M}, \boldsymbol{X}, ks)$
    $(\mathcal{M}, A_0, x_1) \leftarrow \texttt{copyf}(\mathcal{M}, X_0)$
    $(\mathcal{M}, A_{107}, x_2) \leftarrow \texttt{copyf}(\mathcal{M}, X_{107})$
    $(\mathcal{M}, A_{196}, x_3) \leftarrow \texttt{copyf}(\mathcal{M}, X_{196})$
    $A_t = X_t$ for all $t \in \{0, \ldots, 292\} - \{0, 107, 196\}$
    $(\mathcal{M}, \boldsymbol{Y}, x_4) \leftarrow \texttt{maj}(\mathcal{M}, \boldsymbol{A}, 244, 23, 160)$.
    Declare a variable o s.t. $o.val = NULL$, $o.F = 1_c$.   ▷ The complementary of A_{107}.
    $(\mathcal{M}, z) \leftarrow \texttt{xorf}(\mathcal{M}, x_1, x_2, x_3, x_4, ks, o)$
    return $(\mathcal{M}, \boldsymbol{Y}, z)$
end procedure