Improved Algebraic Attacks on Round-Reduced LowMC with Single-Data Complexity

Xingwei Ren[1,2] Yongqiang Li[1,2] Mingsheng Wang[1,2]

[1]Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

August, 2024

Contents

LowMC

- A family of block ciphers with flexible SPN structures.
- First designed for MPC/FHE/ZK protocols at EUROCRYPT 2015.

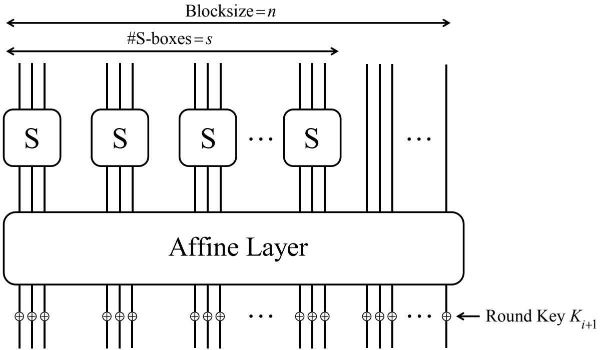

Figure 1: LowMC Round Function

Picnic

- Proposed at CCS 2017.
- A signature scheme in the third round of NIST PQC competition.
- LowMC is as its underlying symmetric primitive.

Security

Picnic is based on the MPC-in-the-head paradigm, its security is equivalent to the difficuly of recovering the secret key K from a single plaintext-ciphertext (P, C).

$$\mathsf{LowMC}_{\mathsf{Enc}}(P, K) = C$$

Picnic3 has introduced new LowMC instances with full S-box layers.

Previous Work

In 2020, the LowMC cryptanalysis competition[a] (with single-data) began...

- Guess-and-determine (GnD) + Meet-in-the-middle (MITM) attack (ToSC 2020, ASIACRYPT 2021, SAC 2022)
- Polynomial method (EUROCRYPT 2021)
- Polynomial method + GnD (ToSC 2022, ePrint 2022, ToSC 2023)

[a] https://lowmcchallenge.github.io/

Linearization Techniques for the LowMC S-box

LowMC employs the 3-bit S-box $S(x_0, x_1, x_2) = (y_0, y_1, y_2)$, where

$$
\begin{aligned}
y_0 &= x_0 \oplus x_1 x_2, \\
y_1 &= x_0 \oplus x_1 \oplus x_0 x_2, \\
y_2 &= x_0 \oplus x_1 \oplus x_2 \oplus x_0 x_1.
\end{aligned}
$$

The First Method: Guess the value of any one output bit

Let $x_0 \oplus x_1 x_2 = c$, the output bits can be rewritten as

$$
\begin{aligned}
y_0 &= c, \\
y_1 &= c \oplus x_1 \oplus c x_2, \\
y_2 &= c \oplus x_1 \oplus x_2 \oplus c x_1.
\end{aligned}
$$

The LowMC S-box is fully linearized. (Similarly for the inverse S-box)

Linearization Techniques for the LowMC S-box

LowMC employs the 3-bit S-box $S(x_0, x_1, x_2) = (y_0, y_1, y_2)$, where

$$y_0 = x_0 \oplus x_1 x_2,$$
$$y_1 = x_0 \oplus x_1 \oplus x_0 x_2,$$
$$y_2 = x_0 \oplus x_1 \oplus x_2 \oplus x_0 x_1.$$

The Second Method: Guess the values of any two input bits

Let $x_0 = c'$ and $x_2 = c''$, the output bits can be rewritten as

$$y_0 = c' \oplus c'' x_1,$$
$$y_1 = c' \oplus x_1 \oplus c' c'',$$
$$y_2 = c' \oplus x_1 \oplus c'' \oplus c' x_1.$$

The LowMC S-box is also fully linearized. (Similarly for the inverse S-box)

Fast Exhaustive Search (FES) Algorithm

How to fastly evaluate a Boolean polynomial of degree d with u variables?

FES Algorithm

1. To evaluate any $f(x)$.
   - An initialization phase $O(u^{2d})$. (negligible when $d \ll u$)
   - Use Gray-codes to enumerate $\forall x \in \mathbb{F}_2^u$, then $f(x)$ can be evaluated within $d \cdot 2^u$ bit operations.
2. To find all zeros of any $\{f_i(x)\}_{i=1}^m$ $(deg(f_i) \leq d)$.
   - Time: $2d \cdot \log_2 u \cdot 2^u$ bit operations.
   - Memory: $m \cdot \binom{u}{\leq d}$ bits, where $\binom{u}{\leq d} = \sum_{i=0}^{d} \binom{u}{i}$.

Dinur's Algorithm

Consider a system $E(x) := \{f_i(x) = 0\}_{i=1}^{m}$, where $x \in \mathbb{F}_2^u$ and $deg(f_i) \leq d$.

Dinur's Algorithm

1. The core idea:
   - Choose a parameter u_1 and split x into $y \in \mathbb{F}_2^{u-u_1}$ and $z \in \mathbb{F}_2^{u_1}$.
   - Randomly select four different choices for the system $\widetilde{E}(y, z)$, each containing $u_1 + 1$ equations from $E(y, z)$.
   - Efficiently enumerate all solutions to each $\widetilde{E}$ and then verify them by E.
     - Based on a polynomial $\widetilde{F}(x) = \prod_{i=1}^{u_1+1}(\widetilde{f}_i(x) \oplus 1)$.
2. Time: $n^2 \cdot 2^{(1-1/2.7d)n}$ bit operations. / Memory: $n^2 \cdot 2^{(1-1/1.35d)n}$ bits.

The GnD Attack Framework for 2-round LowMC

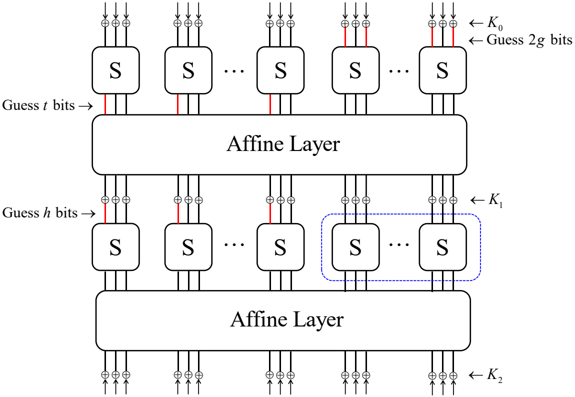

Figure 2: GnD Attack on 2-round LowMC

Preliminaries:

- The key schedule is linear.
- Both the whitened key K_0 and all round keys K_{i+1} are generated by multiplying the master key K with a full-rank binary matrix M_j.
- $\mathsf{Subkey}(i) = \mathsf{Lin}_i(K)$.

The GnD Attack Framework for 2-round LowMC

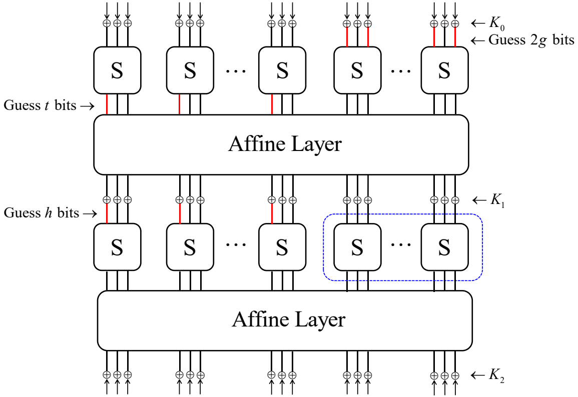

Figure 2: GnD Attack on 2-round LowMC

In the 1st round:

1. Linearize the last g S-boxes by the second method.
2. Obtain $2g$ linear equations about K.
3. Perform Gaussian elimination to yield $n - 2g$ free variables v.
4. Linearize the first $t = s - g$ S-boxes by the first method.

The GnD Attack Framework for 2-round LowMC

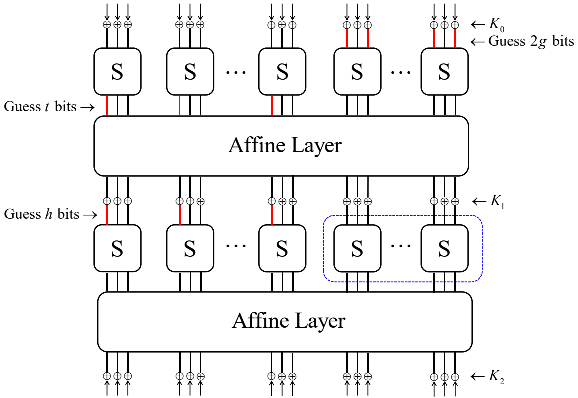

Figure 2: GnD Attack on 2-round LowMC

In the 2nd round:

1. Linearize the first h inverse S-boxes by the first method.
2. Obtain $3h$ linear equations about v.
3. Perform Gaussian elimination to yield $n - 2g - 3h$ free variables β.
4. Construct the target system of $n - 3h$ quadratic equations in terms of β.

The GnD Attack Framework for 2-round LowMC

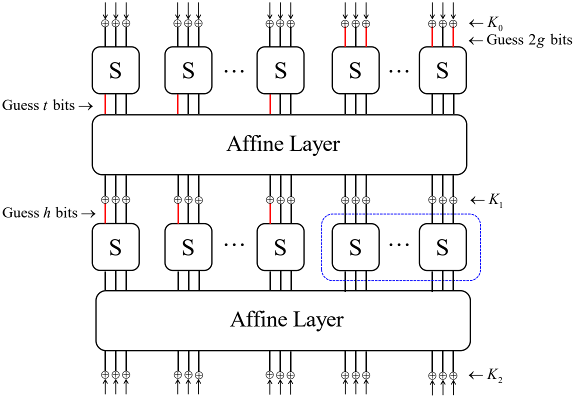

Figure 2: GnD Attack on 2-round LowMC

Solve the target system by using FES or Dinur's algorithm.

- Vs. naive FES, the acceleration factor is $\mathbf{2^{31.9}}/\mathbf{2^{51.7}}/\mathbf{2^{71.8}}$ for the 129/192/255-bit key.
- Vs. Dinur's results, the acceleration factor is $\mathbf{2^{9.8}}/\mathbf{2^{19.8}}/\mathbf{2^{29.8}}$ for the 129/192/255-bit key.
- The required memory is negligible.

2-stage MITM Attack Framework

Due to the linear key schedule of LowMC, the whitened key can be regarded as the secret key $K = [k_1, k_2, \cdots, k_n]$ for cryptanalysis.

1st MITM Stage:

- Split K into three parts $U_0 = [k_1, k_2, \cdots, k_{3h}]$, $U_1 = [k_{3h+1}, k_{3h+2}, \cdots, k_{3h+t}]$ and $U_2 = [k_{3h+t+1}, k_{3h+t+2}, \cdots, k_n]$, where $t = \lfloor (n - 3h)/6 \rfloor \cdot 3$.
- Based on the first method, linearize the inverse of the 2nd round and the first h S-boxes in the 1st round.
- Denote $X = (a_1, a_2, \cdots, a_{3h}, x_1, x_2, \cdots, x_{n-3h})$ to be the output state of the 1st S-box layer.

2-stage MITM Attack Framework

- To reach the state $(a_1, a_2, \cdots, a_{3h})$ from the plaintext and ciphertext, a system of $3h$ linear equations can be constructed, rewritten as

$$A \cdot U_0 = A \cdot [k_1, k_2, \cdots, k_{3h}]^T = B, \quad (1)$$

  where A is an $3h \times 3h$ matrix over $\mathbb{F}_2$, and B is a vector whose elements are affine functions in terms of U_1, U_2.

- Perform Gaussian elimination on Equ. (1), then each bit of U_0 can be an affine function over U_1, U_2.

2-stage MITM Attack Framework

2nd MITM Stage:

- To reach the state x_b ($b \in [1, n-3h]$), each of them can be expressed as

$$
\begin{aligned}
x_b &= f_i(U_1) + c_i = A_i(U_1) + B_i(U_2) \ \text{ for } \ \forall b = i \in [1, t], \\
x_b &= g_j(U_2) + d_j = C_j(U_1) + D_j(U_2) \ \text{ for } \ \forall b = j \in [t+1, n-3h],
\end{aligned} \tag{2}
$$

  where f_i, g_j are quadratic functions and A_i, B_i, C_j, D_j are affine functions, and c_i, d_j are single bit constants.

- Rearrange Equ. (2) to obtain the following collision equations:

$$
\begin{aligned}
f_i(U_1) + A_i(U_1) + c_i &= B_i(U_2), \\
C_j(U_1) &= g_j(U_2) + D_j(U_2) + d_j.
\end{aligned}
$$

2-stage MITM Attack Framework

- Use Gray-codes to enumerate $\forall U_1 \in \{0,1\}^t$, create hash table L_1 indexed by the $(n-3h)$-bit vector $[f_i(U_1)+A_i(U_1)+c_i, \cdots, C_j(U_1)]$.
- Enumerate $\forall U_2 \in \{0,1\}^{n-3h-t}$ in Gray-codes order, create hash table L_2 indexed by the $(n-3h)$-bit vector $[B_i(U_2), \cdots, g_j(U_2)+D_j(U_2)+d_j]$.
- Find possible collisions between L_1 and L_2, the expected number is about $2^{t+n-3h-t} \cdot 2^{3h-n} = 1$.
- When a collision is found, verify the correctness of $K = (U_0, U_1, U_2)$.

2-stage MITM Attack Framework

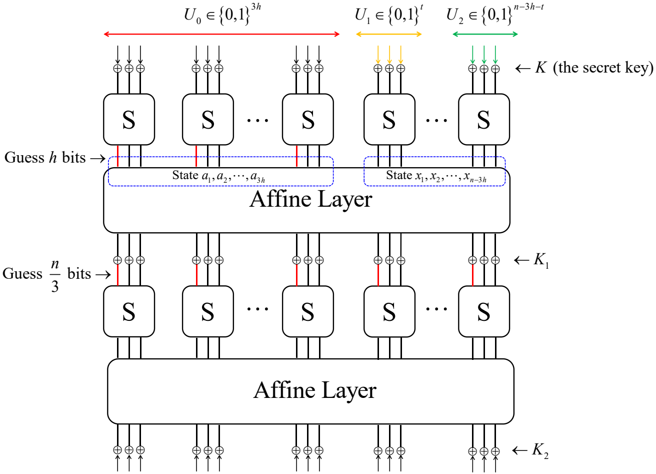

Figure 3: 2-stage MITM Attack Framework for 2-round LowMC

3-stage MITM Attack Framework

The time complexity of attacks can be further reduced…

1st MITM Stage:

- Split K into three parts $V_0 = [k_1, k_2, \cdots, k_{3h}]$, $V_1 = [k_{3h+1}, k_{3h+2}, \cdots, k_{3h+t}]$ and $V_2 = [k_{3h+t+1}, k_{3h+t+2}, \cdots, k_n]$, note that $t = \lfloor (n - 3h)/9 \rfloor \cdot 3$ here.

2nd MITM Stage:

- After 1st MITM stage, the original collision equations can be written as

$$p_i(V_1) + E_i(V_1) + w_i = F_i(V_2) \text{ for } \forall i \in [1, t], \quad (3)$$

$$G_j(V_1) + s_j = q_j(V_2) + H_j(V_2) \text{ for } \forall j \in [t+1, n-3h]. \quad (4)$$

p_i, q_j are quadratic and E_i, F_i, G_j, H_j are affine, and w_i, s_j are constants.

3-stage MITM Attack Framework

- Let $k'_i = k_{3h+i}$ for $\forall i \in [1, t]$ and define

$$\overline{V}_1 = [k'_1, k'_2, k'_3, k'_1k'_2, k'_2k'_3, k'_1k'_3, \cdots, k'_{t-2}, k'_{t-1}, k'_t, k'_{t-2}k'_{t-1}, k'_{t-1}k'_t, k'_{t-2}k'_t].$$

- There exist affine functions $\overline{p}_i$, $\overline{E}_i$, $\overline{G}_j$ over $\overline{V}_1$, so that

$$\overline{p}_i(\overline{V}_1) = p_i(V_1),\ \ \overline{E}_i(\overline{V}_1) = E_i(V_1),\ \ \overline{G}_j(\overline{V}_1) = G_j(V_1).$$

- Equ. (3) and Equ. (4) can be rewritten as

$$\overline{p}_i(\overline{V}_1) + \overline{E}_i(\overline{V}_1) + w_i = F_i(V_2), \tag{5}$$

$$\overline{G}_j(\overline{V}_1) + s_j = q_j(V_2) + H_j(V_2). \tag{6}$$

3-stage MITM Attack Framework

- Define a map ϕ:

$$\overline{V}_1 \rightarrow [\overline{p}_i(\overline{V}_1) + \overline{E}_i(\overline{V}_1), \cdots, \overline{G}_j(\overline{V}_1)]^T.$$

  which can be seen as a linear code of length $n - 3h$ and dimension $2t$.

- Find the $(n - 3h) \times 2t$ generator matrix **G** and the $(n - 3h - 2t) \times (n - 3h)$ check matrix **H** of ϕ.

- Define V_c to be the vector $[w_1, w_2, \cdots, w_t, s_{t+1}, \cdots, s_{n-3h}]^T$. The left side of Equ. (5) and Equ. (6) can be written as $\phi(\overline{V}_1) + V_c$. Note that

$$\mathbf{H} \cdot [\phi(\overline{V}_1) + V_c] = \mathbf{H} \cdot [\mathbf{G} \cdot \overline{V}_1 + V_c] = \mathbf{H} \cdot V_c.$$

3-stage MITM Attack Framework

- Now, split V_2 into two parts $V_2' \in \{0,1\}^t$, $V_2'' \in \{0,1\}^{n-3h-2t}$ and rewrite

$$F_i(V_2) = F_i^{(1)}(V_2') + F_i^{(2)}(V_2''),$$
$$q_j(V_2) = q_j^{(1)}(V_2') + q_j^{(2)}(V_2''),$$
$$H_j(V_2) = H_j^{(1)}(V_2') + H_j^{(2)}(V_2'').$$

- Then define

$$N_1 = [F_i^{(1)}(V_2'), \cdots, q_j^{(1)}(V_2') + H_j^{(1)}(V_2')]^T,$$
$$N_2 = [F_i^{(2)}(V_2''), \cdots, q_j^{(2)}(V_2'') + H_j^{(2)}(V_2'')]^T.$$

- The right side of Equ. (5) and Equ. (6) can be written as $N_1 + N_2$.

3-stage MITM Attack Framework

- Let us make

$$\mathbf{H} \cdot (N_1 + N_2) = \mathbf{H} \cdot V_c \Leftrightarrow \mathbf{H} \cdot N_1 = \mathbf{H} \cdot N_2 + \mathbf{H} \cdot V_c,$$

  which is an additional collision equation.

- Use Gray-codes to enumerate $\forall V_2' \in \{0,1\}^t$, create hash table I_1 indexed by the $(n-3h-2t)$-bit vector $\mathbf{H} \cdot N_1$.

- Use Gray-codes to enumerate $\forall V_2'' \in \{0,1\}^{n-3h-2t}$, create hash table I_2 indexed by the $(n-3h-2t)$-bit vector $\mathbf{H} \cdot N_2 + \mathbf{H} \cdot V_c$.

- Find possible collisions between I_1 and I_2, the expected number is about $2^{t+n-3h-2t} \cdot 2^{3h+2t-n} = 2^t$, which can be stored in table I_0.

3-stage MITM Attack Framework

3nd MITM Stage:

- Enumerate $\forall V_1 \in \{0,1\}^t$ in Gray-codes order, create hash table I_3 indexed by the $(n-3h)$-bit vector $[p_i(V_1)+E_i(V_1)+w_i, \cdots, G_j(V_1)+s_j]$.
- For all values of $V_2 \in I_0$, create hash table I_4 indexed by the $(n-3h)$-bit vector $[F_i(V_2), \cdots, q_j(V_2)+H_j(V_2)]$.
- Find possible collisions between I_3 and I_4, the expected number is about $2^{2t} \cdot 2^{3h-n} \approx 2^{-(n-3h)/3} < 1$.
- When a collision is found, verify the correctness of $K = (V_0, V_1, V_2)$.

Results

n	k	s	r	(h, t)	$\log_2(T)$	$\log_2(M)$	Exh.Search	References
129	129	43	2	/	97	53	145	Asiacrypt 2021
					118	92		Eurocrypt 2021
					125.43	77.4		ePrint 2022
					128.4*	40.2*		ToSC 2023
				$(28, 15)$	94.4	23.3		Ours
192	192	64	2	/	139	75	209	Asiacrypt 2021
					170	126		Eurocrypt 2021
					181.91	112.58		ePrint 2022
					186.6*	55.9*		ToSC 2023
				$(46, 18)$	136.6	26.6		Ours
255	255	85	2	/	182	97	273	Asiacrypt 2021
					222	173		Eurocrypt 2021
					243.03	152.67		ePrint 2022
					244.5*	71.4*		ToSC 2023
				$(67, 18)$	178.7	26.6		Ours

* The optimal complexity was recalculated using the formula in ToSC 2023 paper.

Summary

- 3-stage MITM attacks outperform the best previous 2-round attacks, with memory drastically reduced by a factor of $\mathbf{2^{29.7} \sim 2^{70.4}}$.

- Attacks can be extended to 3-round LowMC by linearizing the 3rd S-box layer, resulting in a factor of 2^s increase in time complexity.

- The security evaluation of LowMC instances with full S-box layers under extremely low-data complexity (≤ 2) remains our future work.

Thanks!

renxingwei@iie.ac.cn