IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 0, No. 0, pp. 0–0. DOI:10.46586/tosc.v0.i0.0-0

Improved Attacks on LowMC with Algebraic Techniques

Yimeng Sun[1,3], Jiamin Cui[1,3] and Meiqin Wang[1,2,3](✉)

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
[2] Quan Cheng Laboratory, Jinan, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
{sunyimeng,cuijiamin}@mail.sdu.edu.cn, mqwang@sdu.edu.cn

Abstract. The LowMC family of SPN block cipher proposed by Albrecht et al. was designed specifically for MPC-/FHE-/ZKP-friendly use cases. It is especially used as the underlying block cipher of PICNIC, one of the alternate third-round candidate digital signature algorithms for NIST post-quantum cryptography standardization. The security of PICNIC is highly related to the difficulty of recovering the secret key of LowMC from a given plaintext/ciphertext pair, which raises new challenges for security evaluation under extremely low data complexity.

In this paper, we improve the attacks on LowMC under low data complexity, i.e. 1 or 2 chosen plaintext/ciphertext pairs. For the difference enumeration attack with 2 chosen plaintexts, we propose new algebraic methods to better exploit the nonlinear relation inside the introduced variables based on the attack framework proposed by Liu et al. at ASIACRYPT 2022. With this technique, we significantly extend the number of attack rounds for LowMC with partial nonlinear layers and improve the success probability from around 0.5 to over 0.9. The security margin of some instances can be reduced to only 3/4 rounds. For the key-recovery attack using a single plaintext, we adopt a different linearization strategy to reduce the huge memory consumption caused by the polynomial methods for solving multivariate equation systems. The memory complexity reduces drastically for all 5-/6-round LowMC instances with full nonlinear layers at the sacrifice of a small factor of time complexity. For 5-round LowMC instances with a block size of 129, the memory complexity decreases from $2^{86.46}$ bits to $2^{48.18}$ bits while the time complexity even slightly reduces. Our results indicate that the security for different instances of LowMC under extremely low data complexity still needs further exploration.

Keywords: LowMC · PICNIC · linearization · algebraic attack · key recovery · crossbred algorithm · polynomial method

1 Introduction

The new progress of advanced cryptographic protocols has motivated a range of symmetric-key primitives dedicated to multi-party computation (MPC), fully homomorphic encryption (FHE), and zero-knowledge proofs (ZKP). As the pioneering work of this direction, LowMC, a family of SPN block ciphers proposed by Albrecht et al. [ARS+15], has attracted lots of attention. It uses partial or complete Sbox layers consisting of m 3-bit S-boxes of degree 2 to minimize the multiplicative complexity. Such a trend has led to several unconventional designs including MiMC [AGR+16], FLIP [MJSC16], Kreyvrium [CCF+18], Rasta [DEG+18], Ciminion [DGGK21] and Poseidon [GKR+21].

Recently, LowMC has been utilized as the underlying block cipher of the digital signature algorithm PICNIC [CDG+17, KZ20], which is one of the third-round alternate candidates for NIST post-quantum cryptography standardization. Let $c = E_K(p)$ be the LowMC encryption of plaintext p under the key K. The plaintext/ciphertext (p, c) is used as the public key of PICNIC, and K is used as the secret key. The security of PICNIC is directly related to the difficulty of recovering K from a single plaintext/ciphertext (p, c).

Previous work. The exploration of reducing the number of multiplications has prompted cryptanalysts to analyze LowMC. The interpolation attack [DLMW15] and higher-order differential attack [DEM16] directly made LowMC move to LowMCv2. As mentioned in PICNIC specification [CDG+17], LowMC instances should be secure with 1 or 2 plaintext/ciphertext pairs, which raises new challenges for security evaluation under extremely low data complexity.

In FSE 2018, Rechberger et al. proposed the difference enumeration attack [RST18] to analyze the security of LowMCv2 with partial nonlinear layers using 3 or slightly more chosen plaintext/ciphertext pairs. Its powerfulness was undoubtedly demonstrated by the break of several instances of LowMCv2 and LowMC was further updated to LowMCv3. For the rest of this paper, we refer to LowMCv3 as LowMC for simplicity.

The difference enumeration attack mainly has two phases: the difference enumeration phase and the key-recovery phase. In the difference enumeration phase, the compact differential trails are recovered, which can be exploited to retrieve the full key in the key-recovery phase. However, the original difference enumeration attack is not quite efficient for the vast memory consumption to store the pre-computed differences and the strong constraints on the number of recovered differential trails. So Liu et al. revisited the difference enumeration attack and developed the algebraic techniques at CRYPTO 2021 [LIM21]. Based on some observations on the 3-bit S-box of degree 2, the problem of recovering the compact differential trails/retrieving the full key can be reduced to solving linear equation systems by introducing intermediate variables. Some important LowMC instances can be successfully broken with only 2 chosen plaintext/ciphertext pairs. Recently, Liu et al. [LSW+22] pointed out that the introduced intermediate variables are not independent and there are nonlinear relations inside them. So they developed an MITM strategy to enumerate the compact differential trails, i.e. utilizing the nonlinear relation of the variables to build tables in the offline phase and then enumerating the remaining variables in the online phase. In the key recovery phase, they introduced quadratic equations and solved them using linearization techniques. By exploiting the above techniques, they further improved the number of attack rounds on LowMC instances with a success probability of around 0.5.

On the other hand, the LowMC team launched a public challenge[1] for LowMC key-recovery in PICNIC use cases under only 1 chosen plaintext/ciphertext pair. The first successful key recovery was proposed by Banik et al. [BBDV20]. The core idea is to linearize the LowMC S-box by guessing a balanced quadratic equation on the input bits of the S-box. Following this work, Banik et al. [BBVY21] proposed a 2-stage MITM method and extended the number of attack rounds for LowMC with partial nonlinear layer to $\lfloor n/m \rfloor$ rounds. In EUROCRYPT 2021, Dinur presented an advanced method of finding roots for a multivariate Boolean equation equation system [Din21]. This attack works quite well for low-degree systems and successfully broke 2-/3-/4- or 5-round LowMC with full nonlinear layer. However, the polynomial method requires vast memory. So Liu et al. [LMSI22] adopted a better time-memory trade-off method and further reduced the memory complexity for 3-/4-round LowMC at a sacrifice of a smaller factor of time complexity. In addition, Banik et al. [BBCV22] also combined the linearization technique [BBDV20, BBVY21] and the polynomial methods in [Din21]. The memory complexity reduces drastically and the

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

number of attacked rounds can be extended to 6 rounds.

Our contributions. This paper is based on the attack against LowMC under extremely low data complexity, i.e. 1 or 2 chosen plaintext/ciphertext pairs. We aim to give a more comprehensive analysis of the security margin of LowMC with state-of-the-art techniques. All the results are summarized in Table 1 and Table 2.

For the difference enumeration attack, we observe that the nonlinear relations inside the free variables can be used in a more fine-grained way to benefit the equation solving in the difference enumeration phase. Specifically, we introduce nonlinear equations of degree 3 for the unknown variables in the online phase instead of enumerating them in [LSW+22] and then solve the equation system effectively utilizing Dinur's algorithm [Din21]. In the key recovery phase, we introduce quadratic equations and use a simple version of the crossbred algorithm [BDT22, LMSI22] to solve an over-defined quadratic equation system, which successfully reduces the time complexity and relaxes the constraints on the number of active S-boxes caused by the linearization technique. In this way, we significantly extend the number of rounds for the difference enumeration attack on LowMC and improve the success probability from 0.5 to over 0.9. The time complexity decreases quickly and the memory complexity also reduces rapidly when $m = 1$. Some LowMC instances have only 3/4 rounds of security margin.

For attacks on LowMC with a single plaintext/ciphertext pair, we perform the guess strategy in [LMSI22] to partition the key space into $2^{2n/3}$ disjoint partial space, i.e. guess 2 input bits instead of 1 quadratic polynomial in [BBDV20]. Then we can find roots by the interpolation and evaluation of the so-called black-box function [Din21, BBCV22] in a partial space of around $n/3$ variables and achieve better time-memory trade-offs. The memory complexity reduces drastically at the sacrifice of a smaller factor of time complexity. In some special cases, the time complexity is even slightly reduced.

Outline. The rest of the paper is organized as follows. In Section 2, we introduce some background knowledge needed in this paper. In Section 3, we show our improvements on difference enumeration attacks against LowMC with 2 plaintext-ciphertext pairs. Then, we present our low-memory attacks on LowMC with a single plaintext/ciphertext pair in Section 4, The experimental results are described in Section 5. Finally, the paper is concluded in Section 6.

2 Preliminaries

2.1 Notation

Due to the various parameters of LowMC [ARS+15], we use n, k, m, and R to represent block size in bits, key size in bits, the number of S-boxes per round, and the total number of rounds, respectively. The following notations will be used throughout this paper.

1. $M_0||M_1$ represents the composition of two matrices M_0 and M_1 of the same number of rows.

2. $V = (a_0, a_1, \cdots, a_{i-1})^T$ represents an i-bit vector. To number the elements in V, we start the index from 0, i.e. $V[j]$ represents the j-th bit in V, $0 \leq j \leq i-1$. For example, $V[0]$ represents the bit a_0.

3. $V[a:b]$ represents the a-th bit to the b-th bit of V, $0 \leq a < b \leq i-1$. For example, $V[0:1]$ represents the two bits $V[0]$ and $V[1]$ of V.

4. $V_0|V_1$ represents the composition of two vectors V_0 and V_1.

Table 1: Summary of the attacks on LowMC with two chosen plaintexts in Section 3. Where T, M, Pro. and $R-r$ represent the $\log_2$ time/memory complexity, success probability and security margin, respectively. D represents the data complexity. T and M are given in the number of LowMC encryptions and bits respectively.

n	k	m	D	R	r_0	r_1	r_2	r	T	M	Pro.	$R-r$	Ref.
128	128	1	2	182	42	68	67	177	125.38	122.76	0.56	5	[LSW+22]
					42	67	68	177	123.21	117.18	0.90	5	Ours
					42	69	68	179	126.91	120.90	0.90	3	Ours
128	128	10	2	20	4	7	6	17	125.2	98.58	0.56	3	[LSW+22]
					4	7	6	17	117.59	106.02	0.90	3	Ours
192	192	1	2	273	64	101	102	267	189.72	182.28	0.51	6	[LSW+22]
					64	101	102	267	186.12	178.56	0.95	6	Ours
					64	104	102	270	191.68	184.14	0.95	3	Ours
192	192	10	2	30	6	9	10	25	189.72	124.62	0.51	5	[LSW+22]
					6	10	9	25	165.38	159.96	0.95	5	Ours
					6	11	9	26	183.93	178.56	0.95	4	Ours
256	256	1	2	363	85	136	136	357	253.34	247.38	0.54	6	[LSW+22]
					85	135	137	357	250.46	241.8	0.97	6	Ours
					85	138	137	360	255.44	249.24	0.97	3	Ours
256	256	10	2	38	8	13	13	34	253.82	187.86	0.54	4	[LSW+22]
					8	13	13	34	231.08	217.62	0.97	4	Ours
					8	14	13	35	249.63	236.22	0.97	3	Ours

5. $\binom{u}{\leq d}=\sum_{i=0}^{d}\binom{u}{i}$, where $\binom{u}{i}$ represents the binomial coefficient.

2.2 Description of LowMC

LowMC [ARS+15] is a family of SPN block ciphers proposed in EUROCRYPT 2015. A notable characteristic of LowMC is that it has multiple parameters to choose from for different use cases. Denote the input and output state of the $(i+1)$-th round by $A^{(i)}$ and $A^{(i+1)}$, respectively. The $(i+1)$-th $(0 \leq i \leq R-1)$ round function of LowMC consists of the following operations:

- SboxLayer (SB): An 3-bit S-box has Algebraic Normal Form (ANF): $S(x_0, x_1, x_2) = (z_0, z_1, z_2)$, where z_0, z_1, z_2 defined as

$$z_0 = x_0 \oplus x_1 x_2,\ z_1 = x_0 \oplus x_1 \oplus x_0 x_2,\ z_2 = x_0 \oplus x_1 \oplus x_2 \oplus x_0 x_1, \tag{1}$$

  The same 3-bit S-box is applied to the first $3m$ bits of the internal state in parallel, while the remaining $n-3m$ bits undergo identity mapping.

- LinearLayer (L): The n-bit internal state is multiplied with an invertible matrix $L_i \in \mathbb{F}_2^{n\times n}$. The matrix L_i is chosen independently and uniformly at random from all invertible binary matrices.

- ConstantAddition (AC): An n-bit round constant C_i is XORed with the n-bit internal state. C_i is randomly generated.

- KeyAddition (AK): An n-bit round key K_{i+1} is XORed with the n-bit internal state, where $K_{i+1} = U_{i+1} \cdot K$. K denotes the k-bit master key and U_{i+1} denotes a full-rank matrix of size $n \times k$. U_{i+1} is randomly generated.

Table 2: Summary of the attacks on LowMC with a single plaintext/ciphertext pair in Section 4. T and M represent the $\log_2$ time/memory complexity respectively, and are given in bits. n_1, l are the parameters used in the attack with $N = 4$ as explained in Section 4. The complexity of exhaustive search is given in the number of bit operations. We further compare our results with the Gray code assisted exhaustive search technique proposed in [BCC+10].

r	n	k	m	N	n_1	l	T	M	Gray Code	Exh.Search	Ref.
5	129	129	43	4	9 3	7 2	134.43 133.87	86.46 45.00	136	146	[BBCV22] Ours
5	192	192	64	4	/ 9 9	/ 7 4	192 196.39 196.78	173 125.38 65.00	199	210	[Din21] [BBCV22] Ours
5	255	255	85	4	/ 15 15	/ 11 6	251 256.74 258.92	228 165.52 84.59	262	274	[Din21] [BBCV22] Ours
6	192	192	64	4	6 6	5 3	197.96 197.52	128.4 65.59	199	211	[BBCV22] Ours
6	255	255	85	4	12 9	9 4	259.62 260.54	167.28 86.00	262	275	[BBCV22] Ours

Notice that there is a whitening key (WK) K_0 before the first round function, where $K_0 = U_0 \cdot K$ and U_0 is also a full rank matrix of size $n \times k$.

Following the notation of [LIM21], the difference of the $(i+1)$-th input state of SB is denoted by Δ_i and the difference of the corresponding output state is denoted by Δ_i^S. The difference of plaintexts is denoted by Δ_p, i.e. $\Delta_p = \Delta_0$. In our attacks, Δ_i and Δ_i^S will be viewed as n-bit vectors.

$$\Delta_p \xrightarrow{WK} \Delta_0 \xrightarrow{SB} \Delta_0^S \xrightarrow{L} \xrightarrow{AC} \xrightarrow{AK} \Delta_1 \longrightarrow \cdots \longrightarrow \Delta_{R-1} \xrightarrow{SB} \Delta_{R-1}^S \xrightarrow{L} \xrightarrow{AC} \xrightarrow{AK} \Delta_R.$$

The **compact differential trail** is defined as below:

Definition 1. [LIM21] A differential trail $\Delta_0 \longrightarrow \Delta_1 \longrightarrow \cdots \longrightarrow \Delta_r$ is called a r-round **compact differential trail** when all (Δ_j, Δ_j^S) $(0 \le j \le r-1)$ and Δ_r are known.

2.3 The Difference Enumeration Attack

The difference enumeration attack was proposed by Rechberger et al. in FSE 2018 [RST18] to analyze the security of LowMCv2 under extremely low data complexity. The difference enumeration attack consists of two phases. The first phase is to enumerate all the possible differential trails to recover the compact differential trails, called the **difference enumeration phase**. The second phase is to retrieve the full key from the recovered differential trails, called the **key-recovery phase**. However, the original difference enumeration attack is not quite efficient for the vast memory consumption to store the pre-computed differences and the strong constraint on the number of recovered differential trails. So Liu et al. [LIM21] revisited the difference enumeration attack and developed the algebraic techniques at CRYPTO 2021. The core idea is to convert the problem of recovering compact differential trails/retrieving the key into solving a linear equation system by introducing intermediate variables. Some important LowMC instances can be successfully broken with only 2 chosen plaintext/ciphertext pairs.

Recently, at ASIACRTPT 2022, Liu et al. [LSW+22] further improved the attack framework in [LIM21] by exploiting the nonlinear relations inside these variables to construct and solve the equation system. In the following, we will briefly introduce the attack framework in [LSW+22]. Throughout the paper, we will use T_d and T_k to represent the complexity of the difference enumeration phase and the complexity of the key-recovery phase, respectively. N_d represents the number of potentially correct compact differential trails after the difference enumeration phase.

2.3.1 Difference Enumeration Phase

As shown in Figure 1, in the difference enumeration phase, we can split the r rounds into three parts: the first r_0 rounds, the middle r_1 rounds and the last r_2 rounds, i.e. $r = r_0 + r_1 + r_2$. Then, the linear equation system can be constructed as follows:

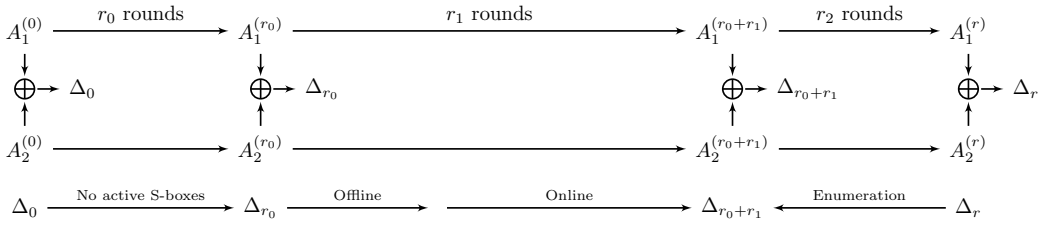

Figure 1: The algebraic MITM attack framework.

1. Choose an input difference Δ_0 such that there will be no active S-boxes in the first r_0 rounds. In this way, Δ_{r_0} is uniquely determined. Therefore, we have $r_0 = \lfloor n/3m \rfloor$.

2. Encrypt a plaintext pair whose XOR difference is Δ_0 for r rounds and obtain the corresponding XOR difference Δ_r of the ciphertexts.

3. Enumerate the state differences backwards from Δ_r for r_2 rounds and obtain the state difference $\Delta_{r_0+r_1}$. For each $\Delta_{r_0+r_1}$, $\Delta^S_{r_0+r_1-1}$ is obtained by applying the inverse of a linear transformation to $\Delta_{r_0+r_1}$. We have $\gamma = \Delta_{r_0+r_1-1}[0:3e-1] | \Delta_{r_0+r_1-1}[3m : n-1]$, where $\Delta_{r_0+r_1-1}[0:3e-1]$ is derived by enumerating the input differences of the first e S-boxes of $\Delta^S_{r_0+r_1-1}$ and $\Delta_{r_0+r_1-1}[3m:n-1] = \Delta^S_{r_0+r_1-1}[3m:n-1]$ due to the partial nonlinear layer.

4. Introduce $3m\ell$ variables $U = (u_0, u_1, \cdots, u_{3m\ell-1})$ to represent the output difference of the $m\ell$ S-boxes in the first $r_1 - 1$ round of the middle r_1 round (see Figure 2), $\ell = r_1 - 1$. Then γ can be written into linear equations in these $3m\ell$ variables as

$$\gamma = M \cdot (u_0, u_1, \cdots, u_{3m\ell-1})^T \oplus \alpha, \tag{2}$$

   where $M \in \mathbb{F}_2^{(n-3m+3e)\times 3m\ell}$ and $\alpha \in \mathbb{F}_2^{n-3m+3e}$ are fixed.

Hence, enumerating the solutions of Equation 2 is equivalent to solving $n - 3m + 3e$ linear equations in $3m\ell$ variables. When the equation system is over-determined, i.e. $3m\ell \leq n - 3m + 3e$, we can expect at most one solution of the equation system. Then all the $3m\ell$ variables are known and we can easily verify the correctness of the differential trails by the differential distribution table (DDT).

When r_1 increases, the equation system becomes under-determined, i.e. $3m\ell > n - 3m + 3e$. If we still assume that these variables are independent, the time complexity to enumerate these variables grows exponentially due to the increased number of variables.

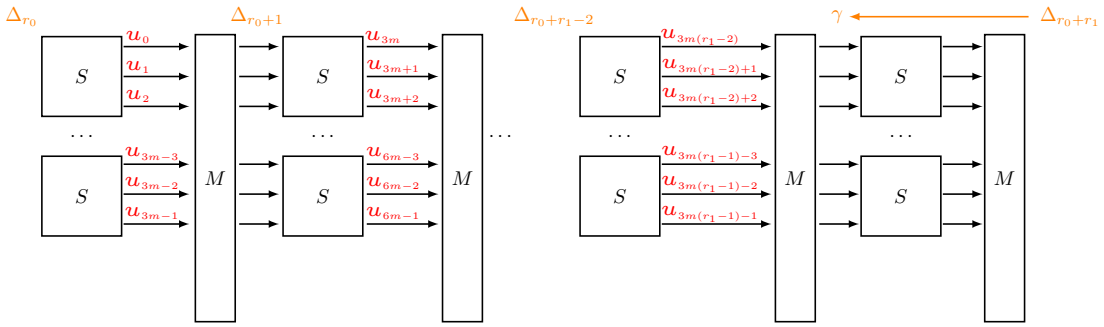

Figure 2: Introduce variables to represent the output differences of the S-boxes.

Notice that since $3m\ell$ variables denote the output difference of the $m\ell$ S-boxes, they only take values from a constrained space[2]. To handle this under-determined linear equation system, we first perform Gaussian elimination on Equation 2 to reduce the linear equation system and the $3m\ell$ variables are split into two parts[3] as follows:

$$\gamma^{'} = M_1^{'} \cdot (u_0, u_1, \cdots, u_{3t-1})^T \oplus M_2^{'}(u_{3t}, u_{3t+1}, \cdots, u_{3m\ell-1})^T \oplus \alpha^{'},$$

where M_2' is in reduced row echelon form and the last $\omega = n - 3m + 3e - rank(M_2')$ rows are all zeros.

So we can easily obtain ω linear equations only involving $(u_0, u_1, \cdots, u_{3t-1})$ as:

$$\gamma^{''} = M_1^{''} \cdot (u_0, u_1, \cdots, u_{3t-1})^T \oplus \alpha^{''}, \tag{3}$$

where $M_1^{''}$ is the submatrix of $M_1^{'}$ representing the last ω rows of $M_1^{'}$ and

$$\gamma^{''} = \gamma^{'}[n - 3m + 3e - \omega : n - 3m + 3e - 1],$$
$$\alpha^{''} = \alpha^{'}[n - 3m + 3e - \omega : n - 3m + 3e - 1].$$

Then [LSW+22] use a time-memory trade-off method to efficiently solve the equation system. The method consists of two phases, i.e. the offline phase and the online phase.

The offline phase. Enumerate the state difference Δ_{r_0} forwards and then we can obtain all the $2^{1.86t}$ possible solutions of $(u_0, u_1, \cdots, u_{3t-1})$. For each solution, compute $\gamma^{''}$ via Equation 3 and insert the tuple $(u_0, u_1, \cdots, u_{3t-1}, \gamma^{''})$ into a table denoted by D_u. Hence, for each $\gamma^{''}$, there are around $2^{1.86t}/2^{\omega}$ corresponding solutions of $(u_0, u_1, \cdots, u_{3t-1})$. After all the solutions are traversed, sort the table D_u according to $\gamma^{''}$.

The online phase. Based on each γ computed backward, we can obtain $\gamma^{''}$ and sort D_u to collect the corresponding values of the tuple $(u_0, u_1, \cdots, u_{3t-1})$. For each $(u_0, u_1, \cdots, u_{3t-1})$, we can perform Gaussian elimination on Equation 2 and solve for $(u_{3t}, u_{3t+1}, \cdots, u_{3m\ell-1})$ by enumerating all the free variables. The correctness can be checked via DDT.

Complexity evaluation. For the offline phase, the time and memory complexity are both $2^{1.86t}$. For the online phase, there are in total $2^{(3m\ell-3t)-(n-3m+3e-\omega)} \times 2^{1.86t-\omega}$ solutions of $(u_0, u_1, \cdots, u_{3m\ell-1})$ for each γ''. So the time complexity is

$$max(2^{1.86(mr_2+e)}, 2^{1.86(mr_2+e)+3mr_1-n-3e-1.14t}).$$

[2]The average number of reachable output differences over the S-box for a uniformly randomly chosen input difference is around $2^{1.86}$ [RST18].

[3]More details can be referred to [LSW+22].

2.3.2 Key-Recovery Phase

In the key-recovery phase, we show how to retrieve the full key from the recovered compact differential trails. Following the attack framework, there is no active S-box in the first r_0 rounds. So we target the last h S-boxes in the last $\lceil h/m \rceil \leq r_1 + r_2$ rounds.

For each active S-box, we can obtain 2 linear equations of the key and intermediate variables according to the following observations.

Observation 1. *[LIM21] For each valid non-zero difference transition* $(\Delta x_0,\ \Delta x_1,\ \Delta x_2) \to (\Delta z_0,\ \Delta z_1,\ \Delta z_2)$*, the inputs conforming to such a difference transition will form an affine space of dimension 1. In addition,* $(z_0,\ z_1,\ z_2)$ *becomes linear in* $(x_0,\ x_1,\ x_2)$*, i.e. the S-box is freely linearized for a valid non-zero difference transition. A similar property also applies to the inverse of the S-box.*

For each inactive S-box, we can introduce 3 intermediate variables to represent its input and obtain 14 quadratic equations in terms of the input bits and output bits as below:

$$z_0 = x_0 \oplus x_1x_2,\ z_1 = x_0 \oplus x_1 \oplus x_0x_2,\ z_2 = x_0 \oplus x_1 \oplus x_2 \oplus x_0x_1,\ x_0 = z_0 \oplus z_1 \oplus z_1z_2,$$
$$x_1 = z_1 \oplus z_0z_2,\ x_2 = z_0 \oplus z_1 \oplus z_2 \oplus z_0z_1,\ z_0x_1 = x_0x_1 \oplus x_1x_2,\ z_0x_2 = x_0x_2 \oplus x_1x_2,$$
$$z_1x_0 = x_0 \oplus x_0x_1 \oplus x_0x_2,\ z_1x_2 = x_1x_2,\ z_3x_0 = x_0 \oplus x_0x_2,\ z_2x_1 = x_1 \oplus x_1x_2,$$
$$z_0x_0 \oplus x_0 = z_1x_1 \oplus x_0x_1 \oplus x_1, z_1x_1 \oplus x_0x_1 \oplus x_1 = z_2x_2 \oplus x_0x_2 \oplus x_1x_2 \oplus x_2.$$

If we denote the number of active S-boxes and the number of inactive S-boxes by a and b, respectively, we can obtain $2a$ linear equations and $14b$ quadratic equations in terms of $k + 3b$ variables. If $2a \geq k + 3b$, the equation system can be easily solved. If $2a < k + 3b$, after performing Gaussian elimination on the $2a$ linear equations, there are $k + 3b - 2a$ free variables in $14b$ quadratic equations. Then the quadratic equation can be solved using the linearization technique by treating all quadratic terms as new variables. If

$$14b \geq (k + 3b - 2a) + (k + 3b - 2a)(k + 3b - 2a - 1)/2,$$

we can expect at most one solution for the full key, and the correctness can be checked via a single plaintext/ciphertext pair. However, this creates a strong constraint on the value of a mainly due to the number of independent variables introduced in the linearization technique. To achieve a high success probability of about 0.5, the lower bound of a is around $a_{min} = \lceil (7h)/8 \rceil$. If $a < a_{min}$, the attack fails.

2.4 Methods for Solving Multivariate Equation Systems

Consider a system of m Boolean equations in u variables denoted by $E(x)$ as:

$$E(x) : E_0(x) = E_1(x) = \cdots = E_{m-1}(x) = 0, \tag{4}$$

where the algebraic degree of each E_i is bounded by $\deg(E_i) \leq d$ and $x = (x_0, x_1, \cdots, x_{u-1}) \in \mathbb{F}_2^u$. The problem of solving the equation system is known to be an NP-hard problem. In this subsection, we revisit two of the current state-of-the-art methods of solving multivariate Boolean equation systems, i.e. the crossbred algorithm for $d = 2$ [BDT22] and Dinur's algorithm for $d \geq 2$ [Din21]. This will benefit our difference enumeration attack since more degrees of freedom could be saved.

2.4.1 The Crossbred Algorithm for Quadratic Equations

In our attack, we will use a simple version of the crossbred algorithm for an over-defined quadratic equation system, which is described in [BDT22, LMSI22]. Suppose

$E(x)$ is a quadratic Boolean equation system, i.e. $d = 2$. The u variables can be partitioned into two disjoint sets and considered independently, i.e. $y \in \mathbb{F}_2^{u-u_1}$ and $z \in \mathbb{F}_2^{u_1}$, where $y = (y_0, y_1, \cdots, y_{u-u_1-1}) = (x_{u_1}, x_{u_1+1}, \cdots, x_{u-1})$ and $z = (z_0, z_1, \cdots, z_{u_1-1}) = (x_0, x_1, \cdots, x_{u_1-1})$. If we view the polynomials of $y_0, y_1, \cdots, y_{u-u_1-1}$ as the coefficients of the monomials $z_0z_1, z_0z_2, \cdots, z_{u_1-2}z_{u_1-1}, z_0, z_1, \cdots, z_{u_1-1}$, the quadratic Boolean equations can be rewritten as:

$$M \cdot (z_0z_1, z_0z_2, \cdots, z_{u_1-2}z_{u_1-1}, z_0, z_1, \cdots, z_{u_1-1})^T = B,$$

where the coefficient matrix M only depends on y and has a size of $m \times (u_1(u_1-1)/2+u_1)$. B is a vector of size m. Since E is a quadratic Boolean equation system, the first $u_1(u_1-1)/2$ columns of M takes value from $\{0,1\}$ and the last u_1 columns are all linear functions of y. Then, we perform the Gaussian elimination on $M||B$ such that the first $u_1(u_1-1)/2$ columns of $M||B$ become reduced row echelon form. Denote the matrix after Gaussian elimination by $M'||B'$. Since the first $u_1(u_1-1)/2$ columns of the last $m - u_1(u_1-1)/2$ rows of M' are all zeros, we can deduce the following equations:

$$M^{''} \cdot (z_0, z_1, \cdots, z_{u_1-1})^T = B'', \tag{5}$$

where $M^{''}$ is a matrix of size $(m - u_1(u_1-1)/2) \times u_1$ and B'' is a vector of size $(m - u_1(u_1-1)/2)$. Note that each element in $M^{''}$ is a linear polynomial in y and each element in $B^{''}$ is a quadratic polynomial in y. We then use Gray code to exhaust all 2^{u-u_1} possible values of y. For each possible y, compute M'' and B'' and solve the Equation 5 . If there is a solution, check the correctness by verifying the remaining equations in the system.

According to [LMSI22], to solve the quadratic equation system effectively, the Equation 5 required to be slightly over-determined, i.e.

$$\epsilon + u_1 = m - u_1(u_1-1)/2, \ \epsilon > 0.$$

The total time complexity is estimated as

$$m^2 \cdot \begin{pmatrix} u \\ \leq 2 \end{pmatrix} + 2^{u-u_1} \cdot (u_1 + \epsilon) \cdot (u_1^2 + u_1 \cdot \epsilon + u) \tag{6}$$

bit operations.

2.4.2 Dinur's Algorithm

In [Din21], Dinur proposed an ingenious method of finding roots for a multivariate Boolean equation system of degree $d \geq 2$. Let $E(x)$ be a polynomial system of degree d over $\mathbb{F}_2$. Consider the equivalent representation

$$A(x) = \prod_{i=0}^{m-1} (E_i(x) \oplus 1), \tag{7}$$

then $A(\hat{x}) = 1$ if and only if $\hat{x}$ is a solution of $E(x)$. Split the u variables in $E(x)$ into two disjoint sets, i.e. $x = (y, z) = (y_0, y_1, \cdots, y_{u-u_1-1}, z_0, z_1, \cdots, z_{u_1-1})$. [Din21] observes that $E(x)$ has an isolated solution with high probability when $u_1 \ll m$ holds. $u_1 \ll m$ is satisfied in all our instances.

Definition 2 (**Isolated Solution** [Din21])**.** A solution $\hat{x} = (\hat{y}, \hat{z})$ to $E(y, z)$ is called isolated (with respect to the variable partition (y, z)) if for all $\hat{z}' \neq \hat{z}$, $(\hat{y}, \hat{z}')$ is not a solution to E.

In order to recover $(\hat{y}, \hat{z})$, we first randomly take $l = u_1 + 1$ different equations from $E(x)$ and construct the new equation system $\tilde{E}(x)$ and the corresponding equivalent representation $\tilde{A}(x)$. It is obvious that a solution of E must be a solution of $\tilde{E}$. But a solution of $\tilde{E}$ is not always a solution of E. We choose to enumerate isolated solutions of $\tilde{E}$ instead of E and then verify them by E to take advantage of the low degree of $\tilde{A}(x)$. Then we can efficiently compute all $u_1 + 1$ partial sums to recover the isolated solutions. More details can be found in [Din21].

To reduce the cost of testing solutions, we perform four different choices for the equation system $\tilde{E}(x)$. The total time complexity can be estimated as

$$4 \cdot (2 \cdot d \cdot \log_2(u) \cdot 2^{u_1} \cdot \left(\begin{array}{c} u - u_1 \\ \leq d_{\tilde{A}} - u_1 + 1 \end{array} \right)) + 4 \cdot (u_1 + 1) \cdot (u - u_1) \cdot 2^{u - u_1} \tag{8}$$

bit operations, where $d_{\tilde{A}} = (u_1 + 1) \cdot d$.

To store the solution obtained from the exhaustive search of $\tilde{E}(y, z)$, the memory complexity can be estimated as $M_1 = 4 \cdot \frac{1}{2} \cdot \binom{u-u_1}{\leq d_{\tilde{A}} - u_1 + 1}$ bits. Using Dinur's memory efficient Möbius transform, the memory complexity of polynomial evaluation can be reduced to $M_2 = 8 \cdot (u_1 + 1) \cdot \binom{u-u_1}{\leq d_{\tilde{A}} - u_1 + 1}$ bits. So the total memory complexity is $M = M_1 + M_2$.

3 The New Difference Enumeration Attack

As mentioned in [LIM21, LSW+22], the problem of enumerating compact differential trails/retrieving the full key can be reduced to solving a linear equation system with non-independent variables. In this section, we show that the nonlinear relations inside the variables can be utilized in a more fine-grained way to benefit the equation solving.

3.1 Difference Enumeration Phase

In Section 2.3.1, the $3m\ell$ introduced variables are split into two parts, i.e. $(u_0, u_1, \cdots, u_{3t-1})$ and $(u_{3t}, u_{3t+1}, \cdots, u_{3m\ell-1})$ and solved in a time-memory trade-off way. That is, we construct the table D_u based on the nonlinear relations inside $(u_0, u_1, \cdots, u_{3t-1})$ and then enumerate the remaining $3mr_1 - 3t - n - 3e + \omega$ free variables based on table D_u. However, when the number of attacked rounds increases, the time complexity to enumerate all the free variables grows since we still assume them all independent variables. This inspires us to exploit the inherent relations of these free variables to solve the equation system more efficiently. An overview of the new attack framework is depicted in 3.

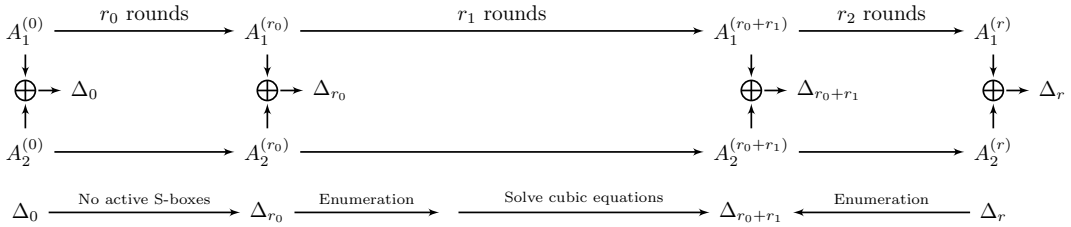

Figure 3: The new difference enumeration attack framework.

In the difference enumeration phase, we still split the r-round LowMC into three parts r_0, r_1, r_2. The general procedure to construct the linear equation system is similar to Section 2.3.1. Since the number of equations becomes smaller than the number of variables as the number of attacked rounds grows, we only consider the case when the equation

system is under-determined. After introducing $3m\ell$ variables $U = (u_0, u_1, \cdots, u_{3m\ell-1})$ to represent the output difference of the $m\ell$ S-boxes in the first $r_1 - 1$ round of the middle r_1 round and performing Gaussian elimination on Equation 2 to split the $3m\ell$ variables, we have the following linear equations:

$$\gamma' = M_1' \cdot (u_0, u_1, \cdots, u_{3t-1})^T \oplus M_2'(u_{3t}, u_{3t+1}, \cdots, u_{3m\ell-1})^T \oplus \alpha', \tag{9}$$

where M_2' is in reduced row echelon form of size $(n - 3m + 3e) \times (3m\ell - 3t)$ and the last $\omega = n - 3m + 3e - rank(M_2')$ rows are all zeros.

Let

$$\begin{aligned}\omega' &= n - 3m + 3e - (3m\ell - 3t)\\ &= n - 3mr_1 + 3e + 3t.\end{aligned}$$

We now only focus on the last ω' rows of M_2'. Then, the last ω' rows of M_2' are all zeros since $rank(M_2') \leq 3m\ell - 3t$ and $\omega' \leq \omega$. This will require

$$n - 3mr_1 + 3e + 3t > 0,\ 3m\ell - 3t > 0.$$

Then, we can immediately obtain ω' linear equations involving $3t$ variables as:

$$\gamma''' = M_1''' \cdot (u_0, u_1, \cdots, u_{3t-1})^T \oplus \alpha''', \tag{10}$$

where M_1''' is the submatrix of M_1' representing the last ω' rows of M_1' and

$$\begin{aligned}\gamma''' &= \gamma'[n - 3m + 3e - \omega' : n - 3m + 3e - 1],\\ \alpha''' &= \alpha'[n - 3m + 3e - \omega' : n - 3m + 3e - 1].\end{aligned}$$

Then we construct the table D_u' based on Equation 10 as the offline phase of Section 2.3.1.

In this way, there are around $2^{1.86t}/2^{\omega'}$ corresponding solutions of $(u_0, u_1, \cdots, u_{3t-1})$ for each γ''', and we can easily retrieve them from D_u' once γ''' is determined. Then, only $(u_{3t}, u_{3t+1}, \cdots, u_{3m\ell-1})$ are remain unknown. Since all the input difference and output difference of the S-boxes from the $\Delta_{r_0+r_1-2}$-th round backwards involved can be represented by the $3m\ell - 3t$ variables, we introduce equations of degree 3 for g S-boxes using Observation 2. Then we obtain $2g$ equations of degree 3 in $3m(r_1 - 1) - 3t$ variables which can be solved efficiently using Dinur's algorithm [Din21].

Observation 2. *[LSW+22] Denote the input difference and output difference of the 3-bit LowMC S-box by $(\Delta x_0, \Delta x_1, \Delta x_2)$ and $(\Delta z_0, \Delta z_1, \Delta z_2)$, respectively. The following 2 cubic equations are sufficient to describe its DDT:*

$$\begin{aligned}&(1 \oplus \Delta x_0)(1 \oplus \Delta x_1)(1 \oplus \Delta x_2) = (1 \oplus \Delta z_0)(1 \oplus \Delta z_1)(1 \oplus \Delta z_2),\\ &(1 \oplus \Delta x_0)(1 \oplus \Delta x_1)(1 \oplus \Delta x_2) = \Delta x_0 \Delta z_0 \oplus \Delta x_1 \Delta z_1 \oplus \Delta x_2 \Delta z_2 \oplus 1.\end{aligned}$$

In the following, we give the whole process of solving Equation 9. The attack consists of two phases, i.e. the offline phase and the online phase.

The offline phase. In the offline phase, we construct the table D_u' according to Equation 10.

Step 1: Enumerate the state difference Δ_{r_0} forwards to obtain all the $2^{1.86t}$ solutions of $(u_0, u_1, \cdots, u_{3t-1})$. For each solution, move to Step 2. After all solutions are traversed, move to Step 3.

Step 2: Focus on the last ω' rows of M_2' and obtain Equation 10. Compute γ''' via Equation 10 and insert the tuple $(u_0, u_1, \cdots, u_{3t-1}, \gamma''')$ into a table denoted by D_u'. Each γ''' will average correspond to $2^{1.86t}/2^{\omega'}$ solutions of $(u_0, u_1, \cdots, u_{3t-1})$.

Step 3: Sort the table D'_u according to γ'''.

The online phase. For each $\gamma = \Delta_{r_0+r_1-1}[0:3e-1]|\Delta_{r_0+r_1-1}[3m:n-1]$ computed backwards, we need to find solutions of $(u_0, u_1, \cdots, u_{3m\ell-1})$

Step 1: Compute γ''' according to the value of γ.

Step 2: For the computed γ''', retrieve from D'_u the corresponding values of the tuple $(u_0, u_1, \cdots, u_{3t-1})$ and move to Step 3.

Step 3: Once $(u_0, u_1, \cdots, u_{3t-1})$ are uniquely determined, we use the remaining $3m(r_1-1)-3t$ variables to represent the input difference and output difference of g S-boxes from the $\Delta_{r_0+r_1-2}$-th round backwards and obtain $2g$ equations of degree 3 based on Observation 2. Then, we perform Dinur's algorithm on them and find the solution of $(u_{3t}, u_{3t+1}, \cdots, u_{3m\ell-1})$. For each solution, the correctness can be checked via DDT. If it is correct, a potentially correct compact r-round differential trail is found.

Complexity evaluation. For the offline phase, the time complexity is $T_1 = 2^{1.86t}$. In the online phase, the time complexity of Dinur's algorithm can be estimated as

$$\begin{aligned} T_2 = {} & 4 \cdot (2 \cdot 3 \cdot \log_2(3m(r_1-1)-3t) \cdot 2^{u_1} \cdot \begin{pmatrix} 3m(r_1-1)-3t-u_1 \\ \leq d_{\tilde{A}} - u_1 + 1 \end{pmatrix}) \\ & + 4 \cdot (u_1+1) \cdot (3m(r_1-1)-3t-u_1) \cdot 2^{3m(r_1-1)-3t-u_1} \end{aligned}$$

bit operations, where $d_{\tilde{A}} = 3 \cdot (u_1+1)$. We choose the parameters (u_1, t, e, g) such that

$$\begin{aligned} & n - 3mr_1 + 3e + 3t > 0,\ 3m\ell - 3t > 0,\ 4(u_1+1) = 2g, \\ & 3m(r_1-1) - 3t > u_1 > 0,\ 3m(r_1-1) - 3t - u_1 \geq d_{\tilde{A}} - u_1 + 1. \end{aligned}$$

The parameter selection is shown in Table 3. If we estimate a single encryption of r-round LowMC encryption as $2rn^2$ bit operations as in [BBVY21], the total complexity of the difference enumeration phase can be estimated as

$$T_d = \max(2^{1.86(mr_2+e)+1.86t-\omega'} \times T_2/2rn^2, 2^{1.86(mr_2+e)}) + 2^{1.86t}$$

times of LowMC encryptions, which implies

$$1.86(mr_2+e) < k,\ 1.86t < k.$$

For the offline phase, the memory complexity is $M_{11} = 2^{1.86t}$. In Dinur's algorithm, the memory complexity is estimated as

$$M_{12} = 4 \cdot \frac{1}{2} \cdot \begin{pmatrix} 3m(r_1-1)-3t-u_1 \\ \leq d_{\tilde{A}} - u_1 + 1 \end{pmatrix} + 8 \cdot (u_1+1) \cdot \begin{pmatrix} 3m(r_1-1)-3t-u_1 \\ \leq d_{\tilde{A}} - u_1 + 1 \end{pmatrix}.$$

The total memory complexity is $M_d = M_{11} + M_{12}$.

3.2 Key-recovery Phase

Based on the above strategy, we can increase the number of rounds for the compact differential trails. As $r_1 + r_2$ increases, there will be much more potentially correct r-round compact differential trails left, i.e. $N_d = 2^{1.86m(r_1+r_2)-n}$. To keep $N_d T_k < 2^k$, it

Table 3: Choices for (u_1, t, e, g) for different (n, k, m, r)

n	k	m	r	(u_1, t, e, g)
128	128	1	177	(1,63,0,4)
		1	179	(1,65,0,4)
		10	17	(1,57,8,4)
192	192	1	267	(2,96,1,6)
		1	270	(2,99,1,6)
		10	25	(2,86,10,6)
		10	26	(2,96,10,6)
256	256	1	357	(2,130,0,6)
		1	360	(1,134,0,4)
		10	34	(1,117,7,4)
		10	35	(1,127,7,4)

becomes crucial to further optimize T_k. Instead of using the linearization technique to add new independent variables for quadratic monomials, we choose to use the crossbred-like algorithm [BDT22] to solve these over-defined quadratic equations, which successfully reduces the time complexity of T_k. In addition, the crossbred-like algorithm relaxes the strong constraints on the number of active S-boxes a. This will significantly improve the success probability of the key-recovery phase to over 0.9. Denote the number of the S-boxes we used in the attack by h and $\lceil h/m \rceil \leq r_1 + r_2$. The attack procedure can be described as follows.

Step 1: Set two counters a and b and initialize them by 0. Choose a threshold a_{min}.

Step 2: If there are fewer than a_{min} active S-boxes in these h S-boxes, exit and return **Failure**. Otherwise, move to Step 3.

Step 3: Starting from a ciphertext, check the S-boxes in the backward direction round by round and one by one.

1) If the S-box is active, increase a by 1. According to Observation 1, we can obtain 2 linear equations in terms of the key and the intermediate variables.

2) If the S-box is inactive, increase b by 1. For each of its three input bits, we introduce 3 intermediate variables.

3) If

$$2a \geq 3b + k, \tag{11}$$

we only need to solve $2a$ linear equations to uniquely determine the k-bit key and move to Step 5. Otherwise, we still check the S-boxes until $a + b = h$. Then move to Step 4.

Step 4: Applying Gaussian elimination on the first $2a$ linear equations will allow us to obtain $3b + k - 2a = 3h - 5a + k$ free variables. For the first λ inactive S-boxes, we can construct 14λ quadratic equations in these free variables, $1 \leq \lambda \leq b$. In other words, the problem is now reduced to solving 14λ quadratic equations in $3h - 5a + k$ variables. We use the crossbred-like algorithm for this problem.

Step 5: The correctness of the solution can be easily verified via a plaintext/ciphertext pair. If it is the correct key, output it and return **Success**.

If $2a \geq 3(h-a)+k$, we only need to solve $2a$ linear equations.If $2a < 3(h-a)+k$, there will be $k-5a+3h$ variables. The quadratic equation system in Step 4 must be slightly overdefined. Hence, we constrain that

$$h - a \geq 1$$

and there exists λ, uu_1, ϵ such that

$$\begin{aligned} &14\lambda - uu_1(uu_1-1)/2 = \epsilon + uu_1, \\ &\epsilon > 0,\ 0 < uu_1 < 3(h-a)+k, \\ &1 \leq \lambda \leq (h-a). \end{aligned}$$

Then, for any a, the crossbred-like algorithm can be adopted to successfully solve the quadratic equation system in Step 4 and the time complexity can be estimated according to Equation 6.

Choice of the threshold. Since the time complexity of the key-recovery phase is affected by the number of active S-boxes a and the choose of λ, uu_1, we choose the threshold a_{min} such that $a = a_{min}$ is the worst case, i.e., when $a > a_{min}$, the time complexity will not be higher. To ensure a success rate of around 0.9, we choose the threshold $a_{min} = \lceil (67h)/81 \rceil$ based on the well-known statistical property

$$\Pr[a' \geq a_{min}] = \sum_{i=a_{min}}^{h} \left(\begin{array}{c} h \\ i \end{array} \right) \times \left(\frac{7}{8}\right)^{i} \times \left(\frac{1}{8}\right)^{h-i} \approx 0.9.$$

However, when $a < a_{min}$, we can still recover the key with the crossbred algorithm.

In Table 4, we select the best (h, a_{min}) for different key sizes. For example, when $k = 128$, the best choice is $(h, a_{min}) = (98, 82)$. For the worst case $a' = 82 \geq a_{min}$, we need to solve 164 linear equations and 28 quadratic equations and the success rate is around 0.9. If we estimate a single encryption of r-round LowMC encryption as $2rn^2$ bit operations, the total cost is $2^{22.22}/(2rn^2)$ times of r-round LowMC encryptions.

Table 4: Choices for different parameters for different key sizes

k	h	a_{min}	$(\lambda, uu_1, \epsilon)$	Pro.	Linear	Quadratic	Cost (T_k)
128	98	82	(2,6,7)	0.90	164	28	$2^{22.22}/(2rn^2)$
192	150	125	(4,10,1)	0.95	250	56	$2^{24.05}/(2rn^2)$
256	207	172	(4,10,1)	0.97	344	56	$2^{25.37}/(2rn^2)$

4 Low-Memory Attacks against LowMC with Single-Data Complexity

In this section, we focus on LowMC instances with the full S-box layer using only a single plaintext/ciphertext pair. The main idea is to combine the linearization techniques of [LMSI22] and the equation-solving methods of [Din21, BBCV22] to reduce the memory complexity when solving the key. Denote the key variables by $K = (k_0, k_1, \cdots, k_{n-1})$, where $n = k$. Without loss of generality, we fix the plaintext to all-zero strings and the input of the S-boxes in the first round is exactly the corresponding key variables.

4.1 Construct the Equation System Using Linearization Technique

Obviously, the problem of key recovery attacks for LowMC with a given plaintext/ciphertext pair can be reduced to solving a system of equations $E(K)$ as

$$E(K)\colon E_0(K) = E_1(K) = \cdots = E_{n-1}(K) = 0.$$

To reduce the algebraic degree of the polynomials, we begin by linearizing the S-boxes of the first round based on the guessing strategy in [LMSI22]. Consider the LowMC S-box $S(x_0, x_1, x_2) = (z_0, z_1, z_2)$, if we guess the two input bits of the S-box, (z_0, z_1, z_2) can be rewritten as linear expressions of (x_0, x_1, x_2) as:

$$z_0 = x_0 \oplus x_1^* x_2^*,\ z_1 = x_0 \oplus x_1^* \oplus x_0 x_2^*,\ z_2 = x_0 \oplus x_1^* \oplus x_2^* \oplus x_0 x_1^*.$$

Then, the input of the second round can be expressed as affine polynomials of the key bits due to the linear key schedule and we can construct the equation system with a meet-in-the-middle method as below.

Odd number of rounds. For an odd number of rounds $r = 2\rho + 1$, we first guess $2m$ input bits in the first round to linearize it. Consider the rounds 2 to $\rho + 1$, the output state of $(\rho + 1)$-th round $A^{(\rho+1)}$ can be expressed by an equation system of degree 2^ρ in key variables. Similarly, consider rounds $2\rho + 1$ to $\rho + 2$, $A^{(\rho+1)}$ can also be expressed as an equation system of degree 2^ρ in key variables. Then we can obtain n equations of degree 2^ρ in n variables by simply equating the two sets of equations.

Even number of rounds. For an even number of rounds $r = 2\rho$, we guess $2m$ input bits in the first round to linearize it. Consider the rounds 2 to ρ, the input state of $(\rho+1)$-th round $A^{(\rho)}$ can be expressed by an equation system of degree $2^{\rho-1}$ in key variables. Similarly, consider rounds 2ρ to $\rho + 2$, the output state of $(\rho + 1)$-th round $A^{(\rho+1)}$ can also be expressed as an equation system of degree $2^{\rho-1}$ in key variables. Since the algebraic degree of the round function is 2, we can easily obtain n equations of degree 2^ρ in n variables by constructing polynomials according to the round function. Specifically, consider the i-th S-box in the $(\rho + 1)$-th round, $0 \le i \le m - 1$. We denote the input polynomials by $p_{3i}(K), p_{3i+1}(K), p_{3i+2}(K)$ and the output polynomials by $q_{3i}(K), q_{3i+1}(K), q_{3i+2}(K)$, respectively, and all polynomials are of degree $2^{\rho-1}$. Then we have

$$\begin{aligned} q_i(K) + p_i(K) + p_{i+1}(K)p_{i+2}(K) &= 0 \\ q_{i+1}(K) + p_i(K) + p_{i+1}(K) + p_i(K)p_{i+2}(K) &= 0 \\ q_{i+2}(K) + p_i(K) + p_{i+1}(K) + p_{i+2}(K) + p_i(K)p_{i+1}(K) &= 0 \end{aligned}$$

Although each polynomial is of degree 2^ρ, if we multiply the 3 polynomials together, the upper bound for the algebraic degree is $4 \times 2^{\rho-1}$ instead of $3 \times 2^\rho$ as pointed out in [Din21]. Moreover, if we multiply 2 of these polynomials then the algebraic degree is $3 \times 2^{\rho-1}$ instead of $2 \times 2^\rho$.

4.2 Black-Box Function over Partial Space

4.2.1 Partial Space and Associated Function

The guessing-and-determining strategy effectively divides the whole key space of $E(K)$ into several partial spaces. Specifically, consider the i-th S-box $S(k_{3i}, k_{3i+1}, k_{3i+2}) = (z_{3i}, z_{3i+1}, z_{3i+2})$ in the first round, $0 \le i \le m - 1$. Let $y_i = z_{3i}$. If we guess two input bits as $k_{3i+1} = g_{2i}$ and $k_{3i+2} = g_{2i+1}$, the 3-bit key variable takes value from the set

$$B^{g_{2i}g_{2i+1}} = \{(y_i \oplus g_{2i}g_{2i+1}, g_{2i}, g_{2i+1}) \mid y_i \in \{0, 1\}\},$$

where $g_{2i}, g_{2i+1} \in \{0,1\}$. For example, consider the case when $g_{2i} = 0$ and $g_{2i+1} = 0$, then we have $B^{00} = \{(0,0,0),(1,0,0)\}$. Similarly, we have

$$B^{01} = \{(0,0,1),(1,0,1)\},\ B^{10} = \{(0,1,0),(1,1,0)\},\ B^{11} = \{(1,1,1),(0,1,1)\}.$$

Then, after guessing $2m$ bits $G = (g_0, g_1, \cdots, g_{2m-1}) \in \{0,1\}^{2m}$, we obtain n equations of degree $d = 2^\rho$ for $2\rho/2\rho+1$ round LowMC instance over the partial space $K \in B^G = B^{g_0 g_1} \times \cdots \times B^{g_{2m-2} g_{2m-1}}$. Denote the equation system by $E_G(K)$. Obviously, for any given $G = (g_0, g_1, \cdots, g_{2m-1})$, there is a one-to-one correspondence between $K = (k_0, k_1, \cdots, k_{3m-1}) \in B^G$ and $Y = (y_0, y_1, \cdots, y_{m-1})$, i.e.

$$E_i(K) = E_{G,i}(K) = \boldsymbol{E}_{G,i}(Y), 0 \le i \le n-1,$$

where for any $0 \le j \le m-1$ we have

$$k_{3j} = y_j \oplus g_{2j} g_{2j+1},\ k_{3j+1} = g_{2j},\ k_{3j+2} = g_{2j+1},\ y_j \in \{0,1\}.$$

We call $\boldsymbol{E}_{G,i}$ the **associated function** of $E_{G,i}$ and Y the **associated vector** of K.

4.2.2 Find the Black-Box Function $F_G, \tilde{F}_G$

Then we are interested in finding roots of E_G for a given $G = (g_0, g_1, \cdots, g_{2m-1}) \in \{0,1\}^{2m}$. To solve the equation system effectively, we first partition the $3m$-bit key variables $K = (k_0, k_1, \cdots, k_{3m-1})$ into two subsets $K_1 = (k_0, k_1, \cdots, k_{3m-3m_1-1})$ and $K_2 = (k_{3m-3m_1}, k_{3m-3m_1+1}, \cdots, k_{3m-1})$ of size $n - n_1 = 3m - 3m_1$ and $n_1 = 3m_1$. The partition of the key variables naturally indicates a partition of G into $G_1 = (g_0, g_1, \cdots, g_{2m-2m_1-1})$ and $G_2 = (g_{2m-2m_1}, g_{2m-2m_1+1}, \cdots, g_{2m-1})$ of size $2m - 2m_1$ and $2m_1$. Let $E_G(K_1, K_2)$ denote the equation system after guessing $G \in \{0,1\}^{2m}$ as

$$E_G(K_1, K_2)\colon E_{G,0}(K_1, K_2) = E_{G,1}(K_1, K_2) = \cdots = E_{G,n-1}(K_1, K_2) = 0.$$

Following the core idea of [Din21], we construct the equivalent representation of E_G as $A_G(K_1, K_2) = \prod_{i=0}^{n-1}(E_{G,i}(K_1, K_2) \oplus 1)$. Let $F_G(K_1) = \bigoplus_{K_2 \in B^{G_2}} A_G(K_1, K_2)$. If $K^* = (K_1^*, K_2^*)$ is a common root of all $E_{G,i}$, then we have $A_G(K^*) = 1$. Otherwise, $A_G(K^*) = 0$. Hence, assume that (K_1^*, K_2^*) is an isolated solution of $E_G(K)$ over B^G, we have

$$F_G(K_1^*) = \bigoplus_{K_2 \in B^{G_2}} A_G(K_1^*, K_2) = 1. \tag{12}$$

Then, we randomly select l equations from the n equations and denote the new equation system as:

$$\tilde{E}_G(K_1, K_2)\colon R_{G,0}(K_1, K_2) = R_{G,1}(K_1, K_2) = \cdots = R_{G,l-1}(K_1, K_2) = 0.$$

So we have $\tilde{A}_G(K_1, K_2) = \prod_{i=0}^{l-1}(R_{G,i}(K_1, K_2) \oplus 1)$ and $\tilde{F}_G(K_1) = \bigoplus_{K_2 \in B^{G_2}} \tilde{A}_G(K_1, K_2)$. Note that a solution of E_G must be a solution of $\tilde{E}_G$, but a solution of $\tilde{E}_G$ is not always a solution of E_G. We choose to enumerate solutions of $\tilde{E}_G$ and then verify them by E_G in order to benefit from the low degree of $\tilde{A}_G$.

The following lemma proves the probability that the root is isolated in $\tilde{A}_G$.

Lemma 1. *Assuming that the underlying LowMC encryption admits a unique root $K^* = (K_1^*, K_2^*) \in B^{G^*}$, then we have the probability that the root is isolated in the system identified by $\tilde{A}_G$ is around $1 - 2^{n_1/3-l}$.*

Proof. Any $K \in B^G$ is a root of any $E_{G,i}$ can be considered to be around $1/2$ for $i \in [0, n-1]$. For $G \neq G^*$, $K_1 = K_1^*$ but $K_2 \neq K_2^*$, the probability that any $K \in B^G$ is a common root of l equations is thus approximately 2^{-l}. So, we have that $Pr[\tilde{A}_G(K_1^*, K_2) = 0] \approx 1 - 2^{-l}$ for all $K_2 \neq K_2^*$. For $G = G^*$, $K_1 = K_1^*$ but $K_2 \neq K_2^*$, the probability that any $K \in B^G$ is a common root of l equations is thus approximately 2^{-l}, too. So, we have that $Pr[\tilde{A}_{G^*}(K_1^*, K_2) = 0] \approx 1 - 2^{-l}$ for all $K_2 \neq K_2^*$. By union bound over all $K_2 \in B^{G_2}$ that are not equal to K_2^* we have the result. For $l = n_1/3 + 1$, this probability is around $\frac{1}{2}$. □

Let $\tilde{\boldsymbol{A}}_G, \tilde{\boldsymbol{F}}_G$ denotes the associated function of $\tilde{A}_G$ and $\tilde{F}_G$, respectively, and $Y = (Y_1, Y_2)$ denotes the associated vector of $K = (K_1, K_2)$ under B^G, we can easily deduce that

$$\tilde{\boldsymbol{F}}_G(Y_1) = \bigoplus_{Y_2 \in \{0,1\}^{m_1}} \tilde{\boldsymbol{A}}(Y_1, Y_2).$$

and

$$\tilde{\boldsymbol{F}}_{G,i}[0](Y_1) = \bigoplus_{Y_2 \in \{0,1\}^{m_1}, y_{m-m_1+i}=0} \tilde{\boldsymbol{A}}(Y_1, Y_2), \text{for } i \in [0, m_1 - 1].$$

Proposition 1. *Assume (K_1^*, K_2^*) to be an isolated solution of $\tilde{E}_G$ and (Y_1^*, Y_2^*) to be the associated vector of (K_1^*, K_2^*). If $Y_2^* = (y_{m-m_1}^*, y_{m-m_1+1}^*, \cdots, y_{m-1}^*)$, then we have $\tilde{\boldsymbol{F}}_G(Y_1^*) = 1$ and $\tilde{\boldsymbol{F}}_{G,i}[0](Y_1^*) = y_{m-m_1+i}^* + 1$ for $i \in [0, m_1 - 1]$.*

Proof. Since (K_1^*, K_2^*) is an isolated solution, we have $\tilde{F}_G(K_1^*) = 1$ and thus $\tilde{\boldsymbol{F}}_G(Y_1^*) = 1$. If $Y_2^*[i] = 0$, we have $\tilde{\boldsymbol{F}}_{G,i}[0](Y_1^*) = 1$. Otherwise, we have $\tilde{\boldsymbol{F}}_{G,i}[0](Y_1^*) = 0$. Hence, $\tilde{\boldsymbol{F}}_{G,i}[0](Y_1^*) = y_{m-m_1+i}^* \oplus 1$ for $i \in [0, m_1 - 1]$. □

4.2.3 Interpolation and Evaluation of $\boldsymbol{F_G}, \tilde{\boldsymbol{F}}_{\boldsymbol{G}}$ over $\boldsymbol{B^G}$

Under the assumption of the isolated solutions, the problem of recovering $K = (K_1, K_2)$ under G can be reduced to finding the roots of $\boldsymbol{F}_G(Y_1) = 1$ or $\tilde{\boldsymbol{F}}_G(Y_1) = 1$ and then recover Y_2 bit-by-bit according to Proposition 1. To find the roots of $\boldsymbol{F}_G(Y_1)/\tilde{\boldsymbol{F}}_G(Y_1)$, we need to find the ANF of them and then evaluate them. Let us take a closer look at F_G and we have

$$\begin{aligned} F_G(K_1) &= \bigoplus_{K_2 \in B^{G_2}} A_G(K_1, K_2) \\ &= \bigoplus_{K_2 \in \{0,1\}^{3m_1}} A_G(K_1, K_2) \prod_{i=m-m_1}^{m-1} (g_{2i} \oplus 1 \oplus k_{3i+1})(g_{2i+1} \oplus 1 \oplus k_{3i+2}). \end{aligned}$$

Obviously, the algebraic degree of A_G is dn. Since $F_G(K_1)$ takes a cube of dimension $3m_1$, it is a function of degree at most $dn + 2m_1 - 3m_1 = dn - m_1 = dn - n_1/3$ over $n - n_1$ variables. Similarly, $\tilde{F}_G(K_1)$ is of degree at most $dl - n_1/3$ over $n - n_1$ variables.

Then we show how to evaluate and interpolate $\boldsymbol{F}_G/\tilde{\boldsymbol{F}}_G$, i.e. the associated function of $F_G/\tilde{F}_G$ over B^G. Obviously, functions $\tilde{\boldsymbol{F}}_G$ and $\tilde{\boldsymbol{F}}_{G,i}[0]$ are both functions of degree at most $D = dl - n_1/3$ over $(n - n_1)/3$ variables. Computing the whole truth table of $\tilde{\boldsymbol{F}}_G$ is equivalent to calculating all possible values of $\tilde{F}_G$ over B^G. For the interpolation, we need a total of $J(N_u, D) = \binom{N_u}{\leq D}$ evaluations of $\tilde{\boldsymbol{F}}_G$, where $N_u = (n - n_1)/3$. Once we obtain the algebraic expression of $\tilde{\boldsymbol{F}}_G$, we can proceed with its evaluation over B^G.

When performing interpolation and evaluation of $\tilde{\boldsymbol{F}}_G$ over B^G, we adopt the algorithm described in [[BBCV22], Appendix A] and the total time complexity for generating the algebraic expression for $\tilde{\boldsymbol{F}}_G$ and the truth table is

$$T(N_u, D) = N_u \cdot \sum_{i=0}^{D-1} \binom{N_u - 1}{i} + D \cdot 2^{N_u}, \tag{13}$$

The total space required is around

$$M(N_u, D) = N_u \cdot J(N_u, D) \cdot \log_2 J(N_u, D) + 2^{N_u} \approx 2^{N_u}. \tag{14}$$

4.3 Details of the Algorithm 1

The pseudo-code of our new attack algorithm is given in Algorithm 1. We now describe it in detail and then analyze its complexity.

The initial partition of key is chosen as $K = (u, v)$ where $u \in \mathbb{F}_2^{n-n_1}$ and $v \in \mathbb{F}_2^{n_1}$, $n_1 = 3m_1$. For $r = 2\rho/2\rho + 1$ rounds of LowMC, we first construct equation systems of degree $d = 2^\rho$ by guessing $G = (G_1, G_2) \in \{0,1\}^{2m}$ (with respect to the partition (u, v)) according to the Section 4.1. Let $E_G(u, v)$ denote the equation system after linearization and l denote the number of equations to construct $\tilde{A}_G$. We then perform the following steps to recover the full key K^*.

1. **Construct table T_G:** Set $D = dl - n_1/3$, $N_u = (n - n_1)/3$. In order to interpolate $\tilde{\boldsymbol{F}}_G$ and $\tilde{\boldsymbol{F}}_{G,i}[0]$, we first pre-compute $E_G(u, v)$ for each of the $J(N_u, D)$ points $u \in B^{G_1}$ and each of the $2^{n_1/3}$ points $v \in B^{G_2}$ and store them in table T_G.

2. **Recover potential solutions:** The following steps iterate for N times.

   2.1 Select l equations randomly from E_G and construct the equation system $\tilde{E}_G(u, v)$.

   2.2 Let (U', V') be the associated vector of (u, v). For each of the $J(N_u, D)$ points $u \in B^{G_1}$, we can calculate the value of $\tilde{\boldsymbol{F}}_G(U')$, $\tilde{\boldsymbol{F}}_{G,i}[0](U')$ by enumerating the $2^{n_1/3}$ points $v \in B^{G_2}$ and computing $\tilde{A}_G(u, v)$ based on T_G.

   2.3 Interpolate $\tilde{\boldsymbol{F}}_G, \tilde{\boldsymbol{F}}_{G,i}[0]$ on $J(N_u, D)$ points $u \in B^{G_1}$ and evaluate $\tilde{\boldsymbol{F}}_G$, $\tilde{\boldsymbol{F}}_{G,i}[0]$ on all the points in B^{G_1} for $i \in [0, m_1 - 1]$.

   2.4 Enumerate $u \in B^{G_1}$ and calculate the associate vector U. If $\tilde{\boldsymbol{F}}_G(U) = 1$, then we can recover V bit-by-bit using Proposition 1 and store $W^* = (U, V)$. If W^* appears twice, we will take it as the potential solution and verify it via a plaintext/ciphertext pair.

If we choose the most primitive method during the verification of the solution, we validate by substituting the solved key into the encryption formula. Testing solutions this way would require around $2^{(n-2n_1)/3} \cdot (2rn^2)$ bit operations for each guess of G. However, one can test the solution in batches, for instance based on their most significant bits as described in [[Din21], Appendix B] which makes the amortized complexity negligible in comparison to the solving complexity.

Complexity evaluation. The attack needs to repeat for $2^{2n/3}$ times, once for each guess of G. We choose $N = 4$, $l = n_1/3 + 1$ and $N_u = (n - n_1)/3$. For the odd number of rounds $R = 2\rho + 1$, the algebraic degree of E_G is $d = 2^\rho$, and the algebraic degree of $\tilde{F}_G$ is $D = dl - n_1/3$. For the even number of rounds $R = 2\rho$, if $l \equiv 0 \bmod 3$, we can randomly choose $l/3$ S-boxes in the $(\rho + 1)$-th round and construct the equations. This reduces the algebraic degree of $\tilde{A}_G$ from $2^\rho \cdot l$ to $4 \cdot 2^{\rho-1} \cdot l/3$ and $D = 4 \cdot 2^{\rho-1} \cdot l/3 - n_1/3$. If $l \equiv 1 \bmod 3$, then we have to choose one additional equation, which makes $D = 4 \cdot 2^{\rho-1} \lfloor l/3 \rfloor + 2^\rho - n_1/3$. If $l \equiv 2 \bmod 3$, then we need to choose 2 additional equations within the same S-box. In this case, $D = 4 \cdot 2^{\rho-1} \lfloor l/3 \rfloor + 3 \cdot 2^{\rho-1} - n_1/3$.

In step 1, the time complexity to compute T_G is

$$T_1 = 2^{2n/3} \cdot J(N_u, D) \cdot 2rn^2 \cdot 2^{n_1/3}.$$

In step 2, the time complexity is estimated as

$$T_2 = N \cdot 2^{2n/3} \cdot (\frac{n_1}{3} + 1) \cdot T(N_u, D).$$

Based on the above discussion, the complexity of verifying the correctness of the potential solutions can be significantly smaller than the complexity of finding the solution themselves, which can be ignored. So the total time complexity is $T = T_1 + T_2$.

The memory complexity required to store T_G in step 1 is

$$M_1 = J(N_u, D) \cdot n \cdot 2^{n_1/3}.$$

In step 2, the memory complexity for interpolation and evaluation is estimated as

$$M_2 = N \cdot (\frac{n_1}{3} + 1) \cdot M(N_u, D).$$

So the total memory complexity is $M = M_1 + M_2$.

5 Experiments

To verify the correctness of our difference enumeration attacks, we performed experiments on concrete LowMC instances. These LowMC instances are generated with the official reference code[4]. The source code of our experiments is available at https://github.com/Momoqw/New-LowMC.git.

During the difference enumeration phase, we chose LowMC instances with parameters $(n, k, m, r) = (128, 128, 1, 103)$ for the purpose of efficiency. In this case, we have $r_0 = 128/3 = 42$, $r_2 = 13$, $r_1 = 48$, and $e = 0$. In order to decrease the memory consumption of storing D'_u, we set $t = 13$ and $u_1 = 4$. Based on Section 3.1, we introduced 141 variables $(u_0, u_1, \cdots, u_{140})$ to represent the output difference of the S-boxes. This allows us to obtain $\omega' = 23$ linear equations in terms of $(u_0, \cdots, u_{38})$. For the offline phase, the size of the table D'_u is 17134432, which is close to our expected value $2^{1.86t} = 2^{24.18}$. Each 23-bit γ''' corresponds to $2^{1.86t/23} \approx 2^{1.05}$ value of $(u_0, \cdots, u_{38})$ in D'_u. For the online phase, we computed γ''' according to Equation 10 for each computed γ and then retrieved the corresponding $(u_0, u_1, \cdots, u_{38})$ using D'_u. Now, only $(u_{39}, u_{40}, \cdots, u_{140})$ remain unknown and we used these 102 variables to construct cubic equations for the last $g = 10$ S-boxes from the 88-th round backwards. The length of the coefficient vector for each cubic equation is 176953. In this way, we can construct an equation system of 20 cubic equations in 102 variables, which can be solved effectively using Dinur's algorithm. All the possible compact differential trails can be recovered by solving the equation system.

In the key recovery phase, we chose LowMC instance with parameters $(n, k, m, r) = (128, 128, 1, 177)$. In this case, we have $r_0 = 42$ and $r_1 + r_2 = 135 > 98$. For the actual success probability, we randomly selected 8000 plaintext pairs with no active S-boxes in the last 82 rounds. We recorded the corresponding differential trails after r rounds of encryption and the number of active S-boxes in the last 98 rounds for each plaintext pair. Denote the number of plaintext pairs that the active S-boxes in the last 98 rounds is not smaller than 82 by N, we find that $N \approx 0.9$, which confirms the correctness of the theory. Furthermore, we successfully reproduced the crossbred algorithm. We chose $uu_1 = 6$, $\lambda = 2$, $a = 82$, which gave us an equation system of 28 quadratic equations in 12 variables. We then used the crossbred algorithm to solve this system. Based on the experimental results, we observed that once the guessed values of the 6 variables are correct, the remaining 6 variables can also be accurately derived, leading to the successful recovery of the whole key.

[4] https://github.com/LowMC/lowmc

Algorithm 1 The algorithm for solving for the key

Input : (p, c), n_1: Internal partition, l: #Equations to construct $\tilde{A}_G$,
N: #Instances algorithm per guess of G, R: #LowMC rounds.
Output: The key K^* such that $Enc_{K^*}(p) = c$.

for Each guess vector $G = (G_1, G_2)$ **do**
  Set $D = dl - n_1/3$, $m_1 = n_1/3$, $N_u = (n - n_1)/3$;
  for Each of the $J(N_u, D)$ points $u \in B^{G_1}$ **do**
    for Each of the $2^{n_1/3}$ points $v \in B^{G_2}$ **do**
      Compute $E_G(u, v)$ and store in table T_G;
    end for
  end for
  for $j = 0 \to N - 1$ **do**
    Select l equations randomly from E_G, denoted as $R_{G,\ell}$, where $\ell \in [0, l-1]$.
    for Each of the $J(N_u, D)$ points $u \in B^{G_1}$ **do**
      $(U^{'}, V^{'}) \leftarrow$ Associated vector of (u, v);
      Set $\tilde{\boldsymbol{F}}_G(U^{'}), \tilde{\boldsymbol{F}}_{G,i}[0](U^{'}) \leftarrow 0, \forall i \in [0, m_1 - 1]$;
      Parse $V^{'} = [y_{m-m_1}, y_{m-m_1+1}, \cdots, y_{m-1}]$;
      for Each of the $2^{n_1/3}$ points $v \in B^{G_2}$ **do**
        for $\ell = 0 \to l - 1$ **do**
          $\tilde{A}_G(u, v) \leftarrow 1$ if all $R_{G,\ell}(u, v) = 0$ else $\tilde{A}_G(u, v) \leftarrow 0$;
          $\tilde{\boldsymbol{F}}_G(U^{'}) \leftarrow \tilde{\boldsymbol{F}}_G(U^{'}) \oplus \tilde{A}_G(u, v)$;
          for $i = 0 \to m_1 - 1$ **do**
            if $y_i = 0$ **then**
              $\tilde{\boldsymbol{F}}_{G,i}[0](U^{'}) \leftarrow \tilde{\boldsymbol{F}}_{G,i}[0](U^{'}) \oplus \tilde{A}_G(u, v)$;
            end if
          end for
        end for
      end for
    end for
    Interpolate $\tilde{\boldsymbol{F}}_G, \tilde{\boldsymbol{F}}_{G,i}[0]$ on $J(N_u, D)$ points $u \in B^{G_1}$, $i \in [0, m_1 - 1]$;
    Evaluate $\tilde{\boldsymbol{F}}_G, \tilde{\boldsymbol{F}}_{G,i}[0]$ on all points in B^{G_1}, $i \in [0, m_1 - 1]$;
    for Each $u \in B^{G_1}$ **do**
      $U \leftarrow$ Associated vector of u;
      if $\tilde{\boldsymbol{F}}_G(U) = 1$ **then**
        $V \leftarrow (1 \oplus \tilde{\boldsymbol{F}}_{G,0}[0](U), \cdots, 1 \oplus \tilde{\boldsymbol{F}}_{G,m_1-1}[0](U))$;
        $W^* \leftarrow (U, 1 \oplus \tilde{\boldsymbol{F}}_{G,0}[0](U), \cdots, 1 \oplus \tilde{\boldsymbol{F}}_{G,m_1-1}[0](U))$, W^* is the associated vector of K^*;
        PotentialSolutionsList$[j][U] \leftarrow V$;
        for $j_1 = 0 \to j$ **do**
          if V=PotentialSolutionsList$[j_1][U]$ **then**
            if $Enc_{K^*}(p) = c$ **then**
              return K^*;
            end if
          end if
        end for
      end if
    end for
  end for
end for

6 Conclusion

We perform an in-depth study of the security strength of LowMC under 1 or 2 chosen plaintext/ciphertext pairs. Benefiting from the parallel application of the 3-bit LowMC S-box of degree 2, both the difference enumeration attacks and the problem of the key recovery for a single plaintext/ciphertext pair can be effectively reduced to the problem of solving a low-degree multivariate equation system. We then propose new algebraic methods according to different equation systems to effectively solve them. For difference enumeration attacks, we significantly push the security margin of difference instances of LowMC with partial nonlinear layers to 3/4 rounds. For the picnic setting, we drastically reduce the memory complexity for 5-/6-round LowMC instances with full nonlinear layers. Our results seem to show that the security evaluation of different instances of LowMC under extremely low data complexity has not yet come to an end.

Acknowledgements

We sincerely thank the anonymous reviewers for providing valuable comments to help us improve the overall quality of the paper. This work is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the National Natural Science Foundation of China (Grant No. 62032014), the Major Basic Research Project of Natural Science Foundation of Shandong Province (Grant No. ZR202010220025), Department of Science & Technology of Shandong Province (No. SYS202201), and Quan Cheng Laboratory (Grant No. QCLZD202306).

References

[AGR+16] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 191–219. Springer, 2016.

[ARS+15] Martin R Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for mpc and fhe. In *Advances in Cryptology–EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34*, pages 430–454. Springer, 2015.

[BBCV22] Subhadeep Banik, Khashayar Barooti, Andrea Caforio, and Serge Vaudenay. Memory-efficient single data-complexity attacks on lowmc using partial sets. *Cryptology ePrint Archive*, 2022.

[BBDV20] Subhadeep Banik, Khashayar Barooti, F Betul Durak, and Serge Vaudenay. Cryptanalysis of lowmc instances using single plaintext/ciphertext pair. *IACR Transactions on Symmetric Cryptology*, 2020(ARTICLE):130–146, 2020.

[BBVY21] Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Hailun Yan. New attacks on lowmc instances with a single plaintext/ciphertext pair. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part I 27*, pages 303–331. Springer, 2021.

[BCC+10] Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 203–218. Springer, 2010.

[BDT22] Charles Bouillaguet, Claire Delaplace, and Monika Trimoska. A simple deterministic algorithm for systems of quadratic polynomials over $\mathbb{F}_2$. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 285–296. SIAM, 2022.

[CCF+18] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. *Journal of Cryptology*, 31(3):885–916, 2018.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *Proceedings of the 2017 acm sigsac conference on computer and communications security*, pages 1825–1842, 2017.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: a cipher with low anddepth and few ands per bit. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I 38*, pages 662–692. Springer, 2018.

[DEM16] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Higher-order cryptanalysis of lowmc. In *Information Security and Cryptology-ICISC 2015: 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers 18*, pages 87–101. Springer, 2016.

[DGGK21] Christoph Dobraunig, Lorenzo Grassi, Anna Guinet, and Daniël Kuijsters. Ciminion: symmetric encryption based on toffoli-gates over large finite fields. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–34. Springer, 2021.

[Din21] Itai Dinur. Cryptanalytic applications of the polynomial method for solving multivariate equation systems over gf(2). In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 374–403. Springer, 2021.

[DLMW15] Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized interpolation attacks on lowmc. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 535–560. Springer, 2015.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for {Zero-Knowledge} proof systems. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 519–535, 2021.

[KZ20] Daniel Kales and Greg Zaverucha. Improving the performance of the picnic signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 154–188, 2020.

[LIM21] Fukang Liu, Takanori Isobe, and Willi Meier. Cryptanalysis of full lowmc and lowmc-m with algebraic techniques. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part III 41*, pages 368–401. Springer, 2021.

[LMSI22] Fukang Liu, Willi Meier, Santanu Sarkar, and Takanori Isobe. New low-memory algebraic attacks on lowmc in the picnic setting. *IACR Transactions on Symmetric Cryptology*, pages 102–122, 2022.

[LSW+22] Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic meet-in-the-middle attack on lowmc. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2022.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient fhe with low-noise ciphertexts. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I 35*, pages 311–343. Springer, 2016.

[RST18] Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. Cryptanalysis of low-data instances of full lowmcv2. *IACR Transactions on Symmetric Cryptology*, pages 163–181, 2018.