Algebraic Meet-in-the-Middle Attack on LowMC

Fukang Liu[1], Santanu Sarkar[4], Gaoli Wang[5], Willi Meier[6], Takanori Isobe[1,2,3]

[1] University of Hyogo, Hyogo, Japan
liufukangs@gmail.com,takanori.isobe@ai.u-hyogo.ac.jp
[2] National Institute of Information and Communications Technology, Tokyo, Japan
[3] PRESTO, Japan Science and Technology Agency, Tokyo, Japan
Indian Institute of Technology Madras, Chennai, India
santanu@iitm.ac.in
[4] East China Normal University, Shanghai, China
glwang@sei.ecnu.edu.cn
[5] FHNW, Windisch, Switzerland
willimeier48@gmail.com

Abstract. We propose a conceptually intuitive technique called algebraic meet-in-the-middle (MITM) attack in this paper. Unlike common MITM attacks where some intermediate state values are stored, several sets of linear equations will be stored in the algebraic MITM attack. Moreover, at the matching phase, it is necessary to first perform some linear transformations on the to-be-matched intermediate state value and only partial state bit information is used for the match. Once a match is found, the corresponding linear equation system is retrieved and solved to recover the full necessary information. This new technique fits very well with LowMC, a popular and important design using partial nonlinear layers. Based on it, we can reduce the memory complexity of the simple difference enumeration attack over the state-of-the-art. Moreover, while an efficient algebraic technique to retrieve the full key from a differential trail of LowMC has been proposed at CRYPTO 2021, its time complexity is still exponential in the key size. In this work, we show how to reduce it to constant time when there are a sufficiently large number of active S-boxes in the trail. Specifically, the guess-and-determine strategy is no more adopted at the key-recovery phase. Instead, we recover the full key by directly solving an overdefined system of quadratic equations. With the above new techniques, the attacks on LowMC and LowMC-M published at CRYPTO 2021 are further improved and some LowMC instances could be broken for the first time. Our results seem to indicate that partial nonlinear layers are still not well-understood.

Keywords: LowMC, LowMC-M, algebraic attack, linearization, key recovery, meet-in-the-middle

1 Introduction

Being the first dedicated symmetric-key primitive design for advanced protocols like secure multiparty computation (MPC) and fully homomorphic encryption

(FHE), the LowMC block cipher family [7] has attracted lots of attention from the cryptography community. Especially, one of the alternate third-round candidate signature schemes in NIST post-quantum cryptography competition [1] called Picnic [16,3] uses LowMC as the underlying block cipher, whose security is directly related to the difficulty to recover the key of LowMC from a single plaintext-ciphertext pair.

Most importantly, the proposal of LowMC directly starts a new trend to design symmetric-key primitives with different metrics, e.g. low AND depth and low AND gates. There is a long line of research focusing on such primitive designs in recent years, like Kreyvrium [15], FLIP [31], Rasta [20], MiMC [6], GMiMC [5], Jarvis [9], Hades [26], Poseidon [25], Vision [8], Rescue [8] and Ciminion [22].

On the other hand, these primitives also raise new challenges for cryptanalysts to evaluate their security. One reason is that some of them are defined over a (large) prime field, which received little attention in symmetric-key cryptanalysis in the past few decades [13,27]. Another important reason is that some of them adopt unusual designs, which make common cryptanalytic techniques difficult to apply. However, these also imply that some fatal errors may be overlooked at the design phase, which is the case of the algebraic attack on full Jarvis [4] and the guess-and-determine attack on the first version of full FLIP [23]. Moreover, due to the lack of tools to evaluate their security, some designs may be too aggressive and will soon turn out to be vulnerable against some novel attacks [13,19,24,28,30,33]. Therefore, developing new cryptanalytic techniques for these new designs becomes an important task for the fact that the security of a symmetric-key primitive is highly related to the evolution of cryptanalytic techniques.

In this paper, we focus on new attacks on LowMC [7] for its special design strategy of using partial nonlinear layers, which has inspired the designs of Hades and a real-world hash function Poseidon. Moreover, its applications in the Picnic signature scheme and the backdoored cipher LowMC-M [32] proposed at CRYPTO 2020 also make it meaningful to further understand its security.

Cryptanalysis of LowMC. Since its publication, LowMC has been quickly analyzed with the higher-order differential attack [21] and interpolation attack [19], which directly made LowMC move to LowMC v2.

To study its security with low data complexity, e.g. its application in the Picnic signature scheme, the difference enumeration attack [33] was proposed to break several instances of LowMC v2 with 3 or slightly more chosen plaintexts. Consequently, the formula to determine the secure number of rounds of LowMC was updated further [2], and this version is called LowMC v3. For convenience, LowMC simply refers to LowMC v3 in the following.

However, a recent work [28] at CRYPTO 2021 reveals that some important LowMC instances are still insecure and they could be broken with 2 chosen plaintexts only and negligible memory complexity. Devising the attacks with 2 chosen plaintexts is mainly due to the assumption used in the security proof of the Picnic signature scheme, where LowMC with 2 plaintexts is required to be secure. Moreover, the idea to attack LowMC with 2 chosen plaintexts can be

simply extended to the attack with a large number of chosen plaintexts, as can be seen from the powerful attacks on LowMC-M in the same paper [28], which has pushed the designers of LowMC-M to increase the number of rounds.

In another direction, the recent LowMC competition to recover the secret key from a single plaintext-ciphertext pair has motivated three teams to develop new attacks [10,11,18,29] on LowMC in this attack scenario. According to the announcement of the third-round results, the attacks with the MITM method [11] and the polynomial method [18] were selected as currently the best attacks. A common feature of both methods [11,18] is the consumption of huge memory. This seems to indicate that the designers are more concerned about the time complexity.

Regarding the memory complexity, we cite the statement in [18]: *There is no consensus among researchers on a model that takes memory complexity into account and the formal security claims of the Picnic (and LowMC) designers only involve time complexity.* Indeed, the memory complexity of an attack may be reduced as new techniques develop. A very recent related example is the algebraic technique in [28], which not only significantly reduces the memory complexity of the original difference enumeration attack [33] on LowMC v2 but also improves the number of attacked rounds.

Our contributions. We aim to further improve the difference enumeration attack on LowMC. The general idea of this attack is very simple. Given a pair of plaintexts and the corresponding ciphertexts, the input difference and output difference of the cipher is known. The first step is to enumerate all the possible differential trails such that this input difference can reach this output difference, i.e. to recover the possible difference transitions through each round. The second step is to retrieve the full secret key from each possible differential trail and test its correctness via the plaintext-ciphertext pair.

For the first step, the memory complexity of the simple MITM method in [33] is exponential in the number of attacked rounds. Although the algebraic technique [28] can make the memory complexity negligible, the number of attacked rounds is limited. Specifically, the algebraic technique converts the difference enumeration into the problem to solve a linear equation system. However, as the number of attacked rounds increases, the linear equation system will become under-determined, i.e. the number of equations is smaller than the number of variables, which will increase the time complexity to enumerate all the possible differential trails as there are many solutions to the under-determined linear equation system. In addition, it is unclear how to use additional memory to improve this algebraic technique.

Our first contribution is a new method to handle this under-determined linear equation system. This is based on the fact that the variables in this equation system are not independent and there are nonlinear relations inside them. Our first attempt is to convert solving such a linear equation system into solving a linear equation system and a nonlinear equation system based on a new observation on the LowMC S-box. However, it is difficult to bound its time complexity and it seems inefficient as the number of attacked rounds

increases. This motivates us to develop a MITM strategy to exploit the nonlinear relations inside these variables to efficiently solve this under-determined linear equation system. This new strategy is shown to outperform both the simple MITM strategy [33] and the simple algebraic technique [28] and is applicable to a wide range of LowMC parameters. Specifically, it can not only reduce the memory complexity of the simple MITM strategy [33] over the state-of-the-art but also improves the number of attacked rounds by using additional memory for the algebraic technique [28]. In a word, the algebraic technique and the MITM strategy are combined in the new strategy and this is why we call it the algebraic MITM attack.

As the number of attacked rounds increases, it is natural that the number of possible differential trails will increase significantly due to the effect of the S-boxes. Hence, it becomes crucial to further optimize the time complexity to retrieve the full key from a random differential trail for the second step. Otherwise, the final time complexity will exceed that of brute force. While a novel and efficient algebraic technique is proposed in [28] to achieve this purpose, i.e. recovering the key by solving a linear equation system, we observe that it is still not extremely optimized due to an inefficient way to deal with the inactive S-boxes in the differential trail. Specifically, the guess-and-determine strategy is still involved in order to process the inactive S-boxes, which makes the time complexity to retrieve the full key from a random differential trail still exponential in the key size. To handle the inactive S-boxes more efficiently, we directly exploit the nonlinear relations in their inputs and outputs. Specifically, by introducing intermediate variables for the inactive S-boxes, we convert the problem to recover the key into solving a linear equation system and a quadratic equation system in terms of more variables, i.e. the key and the intermediate variables. To ensure that the whole equation system can be efficiently solved, we make a compromise that we may fail to recover the correct key with probability of about 0.5. However, as the success probability is sufficiently high. i.e. 0.5, the new technique is useful in practice. In a word, by directly solving a quadratic equation system, the time complexity to retrieve the full key from a random differential trail is reduced to constant time, i.e. almost close to 1.

Due to the significant improvements for both steps, the attacks on LowMC and LowMC-M will naturally be improved. It can be found in Table 4 and Table 5 that the security margins (see the column $R - r$) of LowMC and LowMC-M decrease quickly and some parameters are extremely vulnerable against our new attacks.

Outline of this paper. In Section 2, we introduce the notations and briefly describe LowMC and LowMC-M. Then, an overview of the algebraic MITM attack is given in Section 3. Next, in Section 4, we revisit the previous difference enumeration attacks on LowMC and discuss the problems to improve the attacks. The details of the new methods to enumerate differential trails and to efficiently recover the full key from a differential trail are explained in Section 5 and Section 6, respectively. The summary of our new attacks on LowMC and LowMC-M is shown in Section 7 and the paper is concluded in Section 8.

2 Preliminaries

2.1 Notation

As there are many parameters for both LowMC [7] and LowMC-M [32], we use n, k, m, R and D to represent the block size in bits, the key size in bits, the number of S-boxes in each round, the total number of rounds and the allowed $\log_2$ data complexity for each key, respectively. In addition, $rank(M)$ represents the rank of the matrix M. $M_0||M_1$ represents the composition of two matrices M_0 and M_1 of the same number of rows. $V_0|V_1$ represents the composition of the two vectors V_0 and V_1. $E_{j\times j}$ represents an identity matrix of size $j \times j$. $(a_1, a_2, \ldots, a_i)^T$ also represents an i-bit vector. To number the elements in a vector V, we start the index from 1, i.e. $V[i]$ represents the i-th element in V and $V[1]$ is the first element. In addition, $V[i:j]$ represents a new vector by taking the i-th element to the j-th element from V. For example, when $V_0 = (0,0)^T$ and $V_1 = (1,1,0)^T$, we have $V_0|V_1 = (0,0,1,1,0)^T$ and $V_1[2:3] = (1,0)^T$. When $M_0 = \left(\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right)$ and $M_1 = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, we have $M_0||M_1 = \left(\begin{smallmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{smallmatrix}\right)$.

2.2 Description of LowMC

LowMC [7] is a family of SPN block ciphers proposed at EUROCRYPT 2015. A notable feature of LowMC is that each user can independently choose parameters to instantiate it. LowMC follows a common encryption procedure as most block ciphers. Specifically, it starts with a key whitening (WK) and then iterates a round function R times. The round function at the $(i+1)$-th $(0 \leq i \leq R-1)$ round can be described as follows:

1. SBoxLayer (S): A 3-bit S-box $(z_0, z_1, z_2) = S(x_0, x_1, x_2)$ with $(z_0, z_1, z_2) = (x_0 \oplus x_1 x_2, x_0 \oplus x_1 \oplus x_0 x_2, x_0 \oplus x_1 \oplus x_2 \oplus x_0 x_1)$ is applied to the first $3m$ bits of the state in parallel, while an identity mapping is applied to the remaining $n - 3m$ bits.
2. MatrixLayer (L): A regular matrix $L_i \in \mathbb{F}_2^{n\times n}$ is randomly generated and the n-bit state is multiplied with L_i.
3. ConstantAddition (AC): An n-bit constant $C_i \in \mathbb{F}_2^n$ is randomly generated and is XORed to the n-bit state.
4. KeyAddition (AK): A full-rank $n \times k$ binary matrix U_{i+1} is randomly generated. The n-bit round key K_{i+1} is obtained by multiplying the k-bit master key with U_{i+1}. Then, the n-bit state is XORed with K_{i+1}.

The whitening key is denoted by K_0 and it is also calculated by multiplying the master key with a random full-rank $n \times k$ binary matrix U_0.

LowMC-M [32] is a family of tweakable block ciphers built on LowMC proposed at CRYPTO 2020. The only difference between them is that there is an additional operation AddSubTweak (AT) after AK and WK in LowMC-M, where the sub-tweaks are the output of an extendable-output-function (XOF) function by setting the tweak as the input. A detailed description can be referred

to [32]. As both the tweak and XOF are public, we can equivalently view AT as a known-constant addition operation.

As shown below, in the $(i+1)$-th round, the difference of the input state of S 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.

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

In addition, the **compact differential trail** is defined as below:

Definition 1. [28]*A differential trail $\Delta_0 \rightarrow \Delta_1 \rightarrow \cdots \rightarrow \Delta_r$ is called a r-round* ***compact differential trail*** *when all (Δ_j, Δ_j^S) $(0 \leq j \leq r-1)$ and Δ_r are known.*

3 Overview of the Algebraic MITM Attack

Before moving to the attacks on LowMC, we first give a high-level overview of the algebraic MITM attack. Consider a function $F(\nu_1, \nu_2, \ldots, \nu_i) : \mathbb{F}_2^i \rightarrow \mathbb{F}_2^j$. The common MITM attack procedure can be abstracted as follows:

1. **Offline phase:** Precompute the values of $F(\nu_1, \nu_2, \ldots, \nu_i)$ for a set of possible assignments to $(\nu_1, \nu_2, \ldots, \nu_i)$ and store them in a sorted table denoted by D_F.
2. **Online phase:** Given a challenge, i.e. a value of $F(\nu_1, \nu_2, \ldots, \nu_i)$, try to compute the corresponding solutions of $(\nu_1, \nu_2, \ldots, \nu_i)$. Benefiting from the pre-processing (offline) phase, this can be efficiently finished via the binary search in the table D_F.

For the algebraic MITM attack, we first divide the variables $(\nu_1, \nu_2, \ldots, \nu_i)$ into two sets denoted by $I = \{\nu_{I_0}, \nu_{I_1}, \ldots, \nu_{I_{i_0}}\}$ and $J = \{\nu_{J_0}, \nu_{J_1}, \ldots, \nu_{J_{j_0}}\}$ such that $I \cap J = \emptyset$ and $I \cup J = \{\nu_1, \nu_2, \ldots, \nu_i\}$. In addition, we further assume that when I is known, $F(\nu_1, \nu_2, \ldots, \nu_i)$ can be expressed in linear expressions in J. Note that the expressions will obviously vary for different values of I. Then, the algebraic MITM attack procedure can be abstracted as follows:

1. **Offline phase:** For each possible value of I, store the corresponding linear expressions $F(\nu_1, \nu_2, \ldots, \nu_i)$ in a table denoted by D_E.
2. **Online phase:** Given a challenge, i.e. a value of $\mu = F(\nu_1, \nu_2, \ldots, \nu_i)$, try to compute the corresponding solutions of $(\nu_1, \nu_2, \ldots, \nu_i)$. Different from the common MITM attack procedure, it is difficult to directly obtain which value of $(\nu_1, \nu_2, \ldots, \nu_i)$ will lead to the given challenge μ because the actual possible values of $F(\nu_1, \nu_2, \ldots, \nu_i)$ are not precomputed. However, according to D_E, we know that for each value of I, F is linear in J and the linear expressions are stored. Hence, **we aim to efficiently find a value of I without the knowledge of J such that it may lead to μ**. Once it is

found, we can immediately retrieve from D_E the set of linear expressions of $F(\nu_1, \nu_2, \ldots, \nu_i)$ according to this value of I and then solve the linear equation system $\mu = F(\nu_1, \nu_2, \ldots, \nu_i)$ to recover J, thus obtaining the full solution of $(\nu_1, \nu_2, \ldots, \nu_i)$.

If the algebraic MITM attack works as expected, i.e. there is an efficient method to find a value of I without the knowledge of J such that it may lead to μ and this can be finished with time complexity 1, the time complexity of the online phase in both the algebraic MITM attack and the common MITM attack will be the same. Hence, there is no sacrifice at the time complexity.

However, we can reduce the memory complexity of the common MITM attack over the state-of-the-art. The reduction in the memory complexity is obvious because the solutions of J are computed on-the-fly in the algebraic MITM attack, i.e. we only consider all the possible values of I rather than $(\nu_1, \nu_2, \ldots, \nu_i)$ and leave J as unknown variables at the offline phase of the algebraic MITM attack.

An intuitive comparison between the two attacks can be referred to Figure 1.

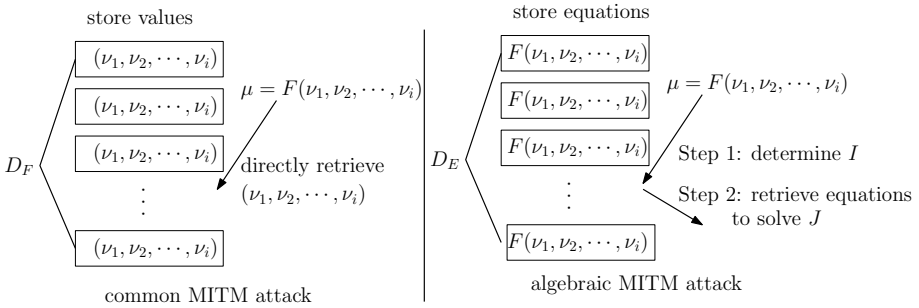

Fig. 1: Comparison between the common MITM attack and the algebraic MITM attack

4 Overview of Previous Difference Enumeration Attacks

The most important application of LowMC is the Picnic signature scheme, which is an alternate third-round candidate in NIST post-quantum cryptography competition. Although the security of Picnic is based on the difficulty to recover the secret key of LowMC under 1 plaintext-ciphertext pair, in its security proof, it is also assumed that the LowMC instance should be secure up to 2 plaintext-ciphertext pairs. This has motivated the LowMC team to devise the difference enumeration attack [33] and they succeeded in breaking many LowMC v2 instances with an extremely low data complexity.

To resist this attack, the formula to compute the secure number of rounds of LowMC is further updated. However, a recent improved attack with algebraic

techniques proposed at CRYPTO 2021 indicates that some latest LowMC parameters are still insecure. In the following, we will briefly describe the above two attacks.

4.1 The Original Attack Framework [33]

The difference enumeration attack [33] is a meet-in-the-middle-style attack, as depicted in Figure 2. From now on, we call the original difference enumeration attack [33] **Attack-O**. The attack procedure[6] in general consists of two crucial steps. First, recover the compact differential trail for 2 chosen plaintexts with a meet-in-the-middle method, which we call the **difference enumeration phase**. Second, compute the key from this recovered differential trail, which we call the **key-recovery phase**. Since there must exist a correct compact differential trail, by restricting that only on average one trail will survive after the difference enumeration phase, the trail obtained after this phase will be the correct one.

Some notations related to the complexity of the attack. Throughout this paper, we denote the time complexity of the difference enumeration phase by T_d and the expected time complexity to retrieve the full key from a compact differential trail by T_k. Moreover, we denote the number of potentially correct compact differential trails after the difference enumeration phase by N_d. In this way, the whole time complexity of the attack is $T_d + N_d T_k$.

An important feature of Attack-O [33] is that $N_d \approx 1$, i.e. only one trail will survive after the difference enumeration phase. Hence, the whole time complexity of the attack becomes $T_d + T_k$.

Constraints at the difference enumeration phase. In the following, we will briefly describe the difference enumeration phase of Attack-O. The key-recovery phase will not be detailed as it is inefficient and it has been significantly improved with algebraic techniques in [28].

As shown in Figure 2, when targeting a r-round attack, 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. $r = r_0 + r_1 + r_2$. The procedure to find the compact differential trails can be then summarized as follows:

Step 1. Find 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, $r_0 = \lfloor n/(3m) \rfloor$.
Step 2. Enumerate the state differences forwards from Δ_{r_0} for the next r_1 rounds and store the set of state differences $\Delta_{r_0+r_1}$. Denote such a set by D_f and the size of D_f by $|D_f|$.
Step 3. Encrypt a plaintext pair whose XOR difference is Δ_0 for r rounds and obtain the XOR difference Δ_r of the ciphertexts.

[6] We consider the standard XOR difference for simplicity.

Step 4. Enumerate the state differences backwards from Δ_r for r_2 rounds and again obtain the state difference $\Delta_{r_0+r_1}$. If the obtained $\Delta_{r_0+r_1}$ is in D_f, one compact differential trail is found and exit. Otherwise, repeat enumerating $\Delta_{r_0+r_1}$ from Δ_r.

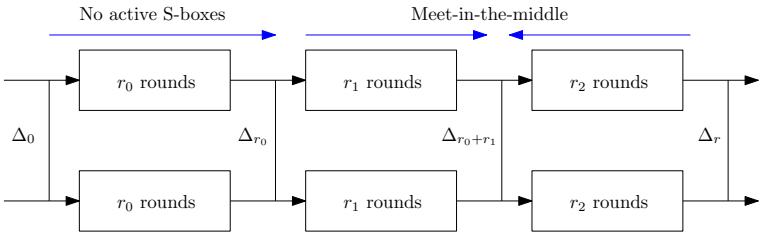

Fig. 2: The original difference enumeration attack framework

Since on average one differential trail is allowed to survive and the time complexity T_d cannot exceed 2^k, the constraints[7] specified in [33] are

$$2^{1.86mr_1} < 2^k, \quad 2^{1.86mr_2} < 2^k, \quad 2^{1.86m(r_1+r_2)} \leq 2^n. \tag{1}$$

For the first two constraints, it is necessary to know the fact that for an input difference of the 3-bit S-box of LowMC, there are on average $29/8 \approx 2^{1.86}$ output differences and vice versa. Hence, the first two constraints mean that the time complexity to construct the set D_f and to enumerate the difference backwards cannot exceed 2^k.

For the last constraint, as the block size is n bits and about $2^{1.86mr_1}$ possible state differences are stored in D_f, the probability to find a match is $2^{-n+1.86mr_1}$. As there are in total $2^{1.86mr_2}$ state differences $\Delta_{r_0+r_1}$ computed backwards, we can expect to find on average $2^{-n+1.86mr_1+1.86mr_2}$ matches, i.e. compact differential trails. To ensure on average one trail survives, we thus have the third constraint $2^{1.86m(r_1+r_2)} \leq 2^n$.

The drawbacks of Attack-O. The most obvious drawback is the consumption of memory to store D_f, which is directly related to the number r_1, i.e. the memory complexity is $2^{1.86mr_1}$. As r_1 increases, the memory complexity increases exponentially. The second drawback is the strong constraint on N_d, i.e. $N_d \approx 1$ is required. Note that the time complexity of the attack is $T_d + N_d T_k$. Hence,

[7] The constraints indeed can be improved with $2^{1.86mr_1-3\tau} < 2^k$, $2^{1.86mr_2} < 2^k$ and $2^{1.86m(r_1+r_2)-3\tau} \leq 2^n$, where $\tau = \lfloor (n-3mr_0)/3 \rfloor$. This is because we can also make τ S-boxes in the (r_0+1)-th round inactive. This slight improvement can work for $m > 1$. However, it is not used in [33]. One reason we believe is that this is not that useful to improve the number of attacked rounds and the improvement is slight. Hence, to make a fair comparison and to make the analysis simpler, this trivial trick to slightly improve the attack will not be considered in our new techniques.

the second drawback can be easily fixed as long as we can make T_k significantly small, which is indeed what the algebraic techniques [28] achieved.

4.2 The Improved Attack Framework

At CRYPTO 2021, Attack-O has been significantly improved with algebraic techniques [28]. The first strategy is to allow many possible r-round compact differential trails to survive after the difference enumeration phase, i.e. N_d can be much larger than 1. The second strategy is to significantly optimize T_k with advanced algebraic techniques. The third strategy is to reduce the memory complexity of the difference enumeration by converting the problem to find a r-round compact trail into the problem to solve a linear equation system, which is inspired by Bar-On et al.'s work [12]. For convenience, we call the improved attack in [28] **Attack-I**.

To avoid the abuse of notation, we still split the r-round LowMC into three parts: the first r_0 rounds, the middle r_1 rounds and the last r_2 rounds, while there will be different constraints for (r_1, r_2) in Attack-I.

The general idea is still the same with Attack-O, i.e. it consists of the difference enumeration phase and the key-recovery phase. As the key-recovery phase will be further optimized in our new attack, its general idea will be explained later and we will mainly focus on the constraints on (r_1, r_2) at the difference enumeration phase in this section.

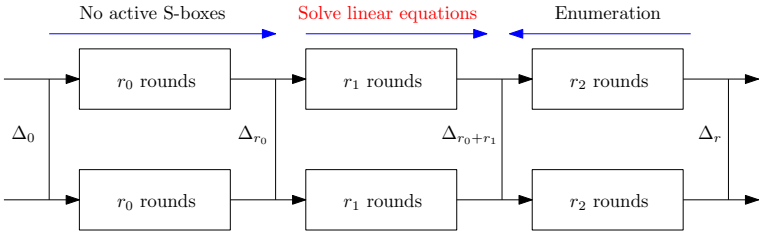

Fig. 3: The improved difference enumeration attack framework

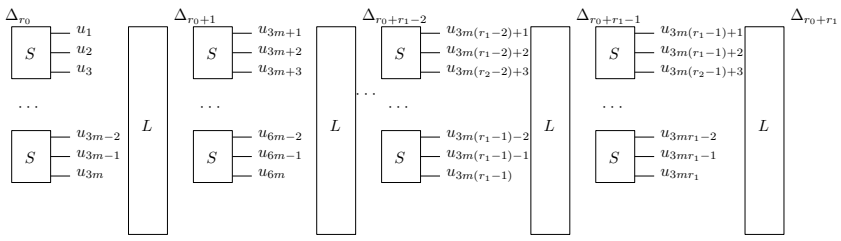

Fig. 4: Introduce variables to represent the output differences of the S-boxes.

As depicted in Figure 3, the general procedure to find a r-round compact differential trail can be described as follows:

Step 1. It is the same as Step 1 of Attack-O.
Step 2. It is the same as Step 3 of Attack-O.
Step 3. Introduce $3mr_1$ variables $U' = (u_1, u_2, \ldots, u_{3mr_1})^T$ to represent the output difference of the all the mr_1 S-boxes in the middle r_1 rounds, as depicted in Figure 4. In this way, in the forward direction, $\Delta_{r_0+r_1}$ can be written as linear expressions in these variables, i.e. $\Delta_{r_0+r_1} = H' \cdot U' \oplus c$, where both the coefficient matrix H' and the n-bit constant vector c are fixed.
Step 4. Enumerate the state differences backwards from Δ_r for r_2 rounds and obtain the state difference $\Delta_{r_0+r_1}$. According to $\Delta_{r_0+r_1}$, solve the linear equation system $\Delta_{r_0+r_1} = H' \cdot U' \oplus c$ and get the solutions of $U' = (u_1, u_2, \ldots, u_{3mr_1})^T$. For each solution, the difference transitions in these r_1 rounds are specified and the correctness can be checked via the differential distribution table (DDT). If it is correct, a potentially correct compact r-round differential trail is found.

The constraints. Indeed, the variables $(u_{3m(r_1-1)+1}, u_{3m(r_1-1)+2}, \ldots, u_{3mr_1})$ are not necessary. Moreover, based on some properties of the S-box, at least $n - 3m + 2m$ linear equations in terms of $(u_1, u_2, \ldots, u_{3m(r_1-1)})$ can be constructed. Hence, $T_d = max(2^{1.86mr_2}, 2^{1.86mr_2+(3mr_1-n-2m)})$. To reach the maximal number of attacked rounds, $r_2 = \lfloor k/(1.86m) \rfloor$. In this way, it can be found in [28] that the maximal value of r_1 is either $\lfloor (n + 2m)/(3m) \rfloor$ or $\lfloor (n + 2m)/(3m) \rfloor + 1$, i.e. $2^{1.86mr_2+(3mr_1-n-2m)} < 2^k$ should hold. Roughly speaking, r_1 is at most $\lfloor n/(3m) \rfloor + 1$.

The advantages of Attack-I. There is no more a strong constraint $N_d \approx 1$. In other words, $N_d = 2^{1.86m(r_1+r_2)-n}$ can be much larger than 1 as long as $N_d T_k < 2^k$. As T_k is significantly reduced in [28], the upper bound for N_d accordingly increases, which implies that the upper bound for $r_1 + r_2$ increases as well. Moreover, computing a compact differential trail is equivalent to solving a linear equation system in Attack-I and therefore there is no need to use a large amount of memory to store a set of possible values for $\Delta_{r_0+r_1}$ computed forwards, i.e. the memory complexity of Attack-I is negligible.

The drawbacks of Attack-I. The most evident drawback of Attack-I is that the constraint on r_1 is too strong, i.e. its maximal value is about $\lfloor n/(3m) \rfloor + 1$, which will directly limit the number of attacked rounds.

4.3 Problems to Improve the Attacks

The maximal values of (r_0, r_2) in both Attack-O and Attack-I are the same, which are specified below:

$$r_0 = \lfloor n/(3m) \rfloor, \quad r_2 = \lfloor k/(1.86m) \rfloor. \tag{2}$$

There seems to be little room to improve the upper bounds for (r_0, r_2).

However, this seems to be not the case for r_1. Assuming T_k can be reduced to 1, the maximal value of $N_d = 2^{1.86m(r_1+r_2)-n}$ will then be slightly smaller than 2^k, i.e. $T_k N_d < 2^k$ has to hold. In this way, with the simple MITM strategy of Attack-O, the maximal value for r_1 is $\lfloor k/(1.86m) \rfloor$ and the memory complexity is $2^{1.86mr_1}$. With the algebraic techniques of Attack-I, the maximal value for r_1 is about $\lfloor n/(3m) \rfloor + 1$ and the memory complexity is negligible.

When n is much larger than k and m is small, e.g. $(n, k, m) = (1024, 128, 1)$, the algebraic technique is more powerful than the simple MITM strategy as the upper bound for r_1 is much larger. Moreover, the memory complexity is negligible. However, it is not difficult to observe that even allowing attackers to use memory in Attack-I, it is unclear how to use it to further increase r_1, i.e. what should we store in advance?

When $n = k$, e.g. $(n, k, m) = (128, 128, 1)$, the simple MITM strategy is more powerful because it is possible to pick an r_1 such that $r_1 > \lfloor n/(3m) \rfloor + 1$ at the cost of using $2^{1.86mr_1}$ memory.

Hence, the to-be-solved problems now become clear, as stated below:

Problem 1. For the case when the simple MITM strategy is more powerful at the cost of using memory, e.g. $n = k$, how to significantly optimize the memory complexity?

Problem 2. For the case when the algebraic technique is more powerful, e.g. $n >> k$, how to further enlarge r_1 by using memory?

5 The Algebraic MITM Attack Framework

In Attack-I, finding a compact differential trail is reduced to solving a linear equation system with the introduction of intermediate variables. An implicit assumption in this attack is that these intermediate variables are independent. Obviously, this is not the fact because we still need to check the validity of the obtained solutions of these variables via DDT. This motivates us to consider whether it is possible to exploit some nonlinear relations in these variables.

5.1 A New Observation on the 3-bit S-box

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. We observe that the following 2 cubic equations are sufficient to fully describe its DDT:

$$(1 \oplus \Delta x_0)(1 \oplus \Delta x_1)(1 \oplus \Delta x_2) = (1 \oplus \Delta z_0)(1 \oplus \Delta z_1)(1 \oplus \Delta z_2),$$
$$(1 \oplus \Delta x_0)(1 \oplus \Delta x_1)(1 \oplus \Delta x_2) = \Delta x_0 \Delta z_0 \oplus \Delta x_1 \Delta z_1 \oplus \Delta x_2 \Delta z_2 \oplus 1.$$

Moreover, when $(\Delta x_0, \Delta x_1, \Delta x_2) \neq (0, 0, 0)$ and $(\Delta z_0, \Delta z_1, \Delta z_2) \neq (0, 0, 0)$, all the possible values of $(\Delta x_0, \Delta x_1, \Delta x_2, \Delta z_0, \Delta z_1, \Delta z_2)$ can be fully described

with only 1 quadratic equation, while all the invalid values do not satisfy it, as specified below:

$$\Delta x_0 \Delta z_0 \oplus \Delta x_1 \Delta z_1 \oplus \Delta x_2 \Delta z_2 \oplus 1 = 0. \tag{3}$$

This quadratic equation perfectly explains the property used in [28] that for each nonzero input difference, its output differences form an affine space of dimension 2 and vice versa.

Although the above equations are derived by hand, it is also possible to run a simple algorithm to find them. Specifically, we first guess the degree of the equations and then determine the coefficients of the terms in the equations, which can be converted into solving a linear equation system in terms of the to-be-determined coefficients. Each solution of the coefficients will correspond to a possible equation. This is a widely-used method [17]. Using this algorithm, we can simply prove that the DDT of the LowMC S-box cannot be described with only 1 quadratic or cubic equation.

What does the new property imply? With the algebraic technique to find a compact differential trail, it seems possible to enlarge r_1. Specifically, if not considering the dependency between the $3m(r_1 - 1)$ variables, we can obtain at least $n - m$ linear equations in these variables. When $n + 2m - 3mr_1 \approx 0$, these variables can be easily solved as in Attack-I. However, when $3mr_1 - n - 2m >> 0$, we may need to enumerate too many solutions to this linear equation system. However, we may reduce the cost to enumerate the solutions by solving nonlinear equations based on the new observations. Specifically, after performing Gaussian elimination on the $n - m$ linear equations, we obtain $3mr_1 - 2m - n$ free variables. Then, by utilizing the nonlinear relations in the input difference and output difference of the 3-bit S-box, equations of degree 3 in these free variables can be constructed because it is unclear which S-box is inactive. In this way, enumerating the solutions is equivalent to solving a multivariate system of degree-3 equations.

As r_1 increases, the number of free variables $3mr_1 - 2m - n$ increases in a very fast way. In addition, it is difficult to bound the time complexity to solve such a system of nonlinear equations. Moreover, when $2^{1.86r_2}$ becomes close to 2^k, it is required to solve such an equation system with time complexity close to 1. One way to remove this constraint is to decrease r_2 and increase r_1 at the cost to solve a system of nonlinear equations in much more variables, which will further make the complexity evaluation difficult. We leave this observation here. In the following, we will describe a different way to utilize these nonlinear relations, which is combining the MITM strategy and the algebraic techniques.

5.2 A New Attack Framework

As mentioned above, it is necessary to develop a new technique to solve an under-determined system of linear equations by utilizing the nonlinear relations in the variables. Moreover, it is expected that the time complexity to solve these variables is 1.

To avoid the abuse of notation, we still split the r-round LowMC into three parts as in Attack-O and Attack-I, i.e. the first r_0 rounds, the middle r_1 rounds and the last r_2 rounds. An overview of the new attack framework is depicted in Figure 5 and we call it **Attack-N**.

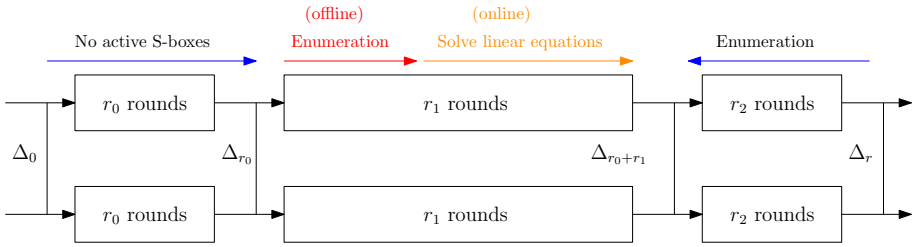

Fig. 5: The new difference enumeration attack framework

Focus on $(\Delta_{r_0+r_1-1}[1:3e], \Delta^S_{r_0+r_1-1})$, where $\Delta^S_{r_0+r_1-1}$ is computed by performing the inverse of the linear transform on $\Delta_{r_0+r_1}$ and $\Delta_{r_0+r_1-1}[1:3e]$ is obtained by further enumerating the input differences for the first e S-boxes starting from $\Delta^S_{r_0+r_1-1}$. Note that $\Delta^S_{r_0+r_1-1}[3m+1:n] = \Delta_{r_0+r_1-1}[3m+1:n]$ due to the partial nonlinear layer.

Let

$$l = r_1 - 1, \tag{4}$$

$$\gamma = \Delta_{r_0+r_1-1}[1:3e] | \Delta_{r_0+r_1-1}[3m+1:n], \tag{5}$$

i.e. γ is a $(3e+n-3m)$-bit vector representing the concatenation of the 1st bit to the $3e$-th bit and the $(3m+1)$-th bit to the n-th bit of $\Delta_{r_0+r_1-1}$.

Similarly, we introduce $3ml$ variables $U = (u_1, u_2, \ldots, u_{3ml})$ variables to represent the output differences of the S-boxes in the first $r_1 - 1$ rounds of the middle r_1 rounds. In this way, we have

$$\gamma = M \cdot (u_1, u_2, \ldots, u_{3ml})^T \oplus \alpha, \tag{6}$$

where M is a fixed matrix of size $(n-3m+3e) \times 3ml$ and $\alpha \in \mathbb{F}_2^{n-3m+3e}$ is uniquely determined by the fixed state difference Δ_{r_0}.

We now only consider the case when Equation 6 is under-determined, i.e. $n-3m+3e < 3ml$. Our aim is to efficiently solve the variables $U = (u_1, u_2, \ldots, u_{3ml})$ given an arbitrary γ. For this purpose, we adopt a two-phase method, i.e. the offline phase and online phase.

Solving the under-determined linear equation system. Before moving to the details of the two phases, we need to perform some analysis for this under-determined linear equation system. Our critical observation is that both the coefficient matrix M and the constant vector α are fixed.

Let $M = M_0||M_1$, where M_0 represents the first q columns of M while M_1 represents the last $3ml - q$ columns of M, i.e. M_0 is of size $(n - 3m + 3e) \times q$ and M_1 is of size $(n - 3m + 3e) \times (3ml - q)$.

First, let

$$Q' = M_1||E_{(n-3m+3e)\times(n-3m+3e)}.$$

Then, we perform the Gaussian elimination on the matrix Q' such that M_1 becomes the reduced row echelon form. Denote the new matrix after Gaussian elimination by $Q = Q_0||Q_1$, where Q_0 is of size $(n - 3m + 3e) \times (3ml - q)$ and Q_1 is of size $(n - 3m + 3e) \times (n - 3m + 3e)$. In this way, we have

$$Q_0 = Q_1 \cdot M_1$$

and Q_0 is in reduced row echelon form.

Let

$$\omega = n - 3m + 3e - rank(Q_0). \tag{7}$$

Then, the elements in the last ω rows of Q_0 are all zero.

After obtaining the transform matrix Q_1, we apply it to Equation 6, as shown below:

$$Q_1 \cdot \gamma = Q_1 \cdot M \cdot (u_1, u_2, \ldots, u_{3ml})^T \oplus Q_1 \cdot \alpha.$$

Note that the above new equation system is equivalent to the original equation system Equation 6.

Let

$$\beta = Q_1 \cdot \gamma,\ \epsilon = Q_1 \cdot \alpha,\ P = Q_1 \cdot M,\ P_0 = Q_1 \cdot M_0. \tag{8}$$

In this way, $P = P_0||Q_0$ due to $Q_0 = Q_1 \cdot M_1$ and we can further obtain an equivalent representation of Equation 6, as specified below:

$$\beta = P_0 \cdot (u_1, u_2, \ldots, u_q)^T \oplus Q_0 \cdot (u_{q+1}, u_{q+2}, \ldots, u_{3ml})^T \oplus \epsilon. \tag{9}$$

Note that Q_0 is in reduced row echelon form.

Analyzing Equation 9. Since the elements in the last ω rows of Q_0 are all zero, we immediately obtain ω linear equations only involving β, ϵ and $(u_1, u_2, \ldots, u_q)$, as shown below:

$$\beta' = P_0' \cdot (u_1, u_2, \ldots, u_q)^T \oplus \epsilon', \tag{10}$$

where P_0' is the submatrix of P_0 representing the last ω rows of P_0, while β' and ϵ' are both an ω-bit vector representing the last ω elements of the bit vectors β and ϵ, respectively. Formally speaking,

$$\beta' = \beta[n - 3m + 3e - \omega + 1 : n - 3m + 3e],$$

$$\epsilon' = \epsilon[n - 3m + 3e - \omega + 1 : n - 3m + 3e]. \tag{11}$$

Suppose that there are N_u possible values of $(u_1, u_2, \ldots, u_q)$, which can be computed independent of $(u_{q+1}, u_{q+2}, \ldots, u_{3ml})$. In this case, for each possible value of $(u_1, u_2, \ldots, u_q)$, we can uniquely determine β' as ϵ' is a constant vector. Therefore, we can precompute N_u possible values for $(u_1, u_2, \ldots, u_q, \beta')$.

As β' represents an ω-bit vector, it can take in total 2^ω values. Hence, for the computed N_u possible values of $(u_1, u_2, \ldots, u_q, \beta')$, we can equivalently say that on average each β' corresponds to $N_u/2^\omega$ solutions of $(u_1, u_2, \ldots, u_q)$.

5.3 The Algebraic MITM Strategy

With the above analysis in mind, it is now easy to explain how to combine the MITM strategy and the algebraic technique. Note that $(u_{3i+1}, u_{3i+2}, u_{3i+3})$ represents the output difference of an S-box. Let us focus on the first $3t$ variables, i.e. the variables $(u_1, u_2, \ldots, u_{3t})$.

The offline phase. For such a configuration, we have $q = 3t$. Moreover, there are about $2^{1.86t}$ possible values for $(u_1, u_2, \ldots, u_{3t})$ if we enumerate the state difference Δ_{r_0} in the forward direction, thus resulting in $N_u = 2^{1.86t}$. In other words, on average each β' will correspond to $2^{1.86t}/2^\omega$ solutions of $(u_1, u_2, \ldots, u_{3t})$, which can be computed in advance via Equation 10. Hence, the offline phase can be described as follows.

Step 1: Enumerate the state difference Δ_{r_0} forwards to obtain all the solutions of $(u_1, u_2, \ldots, u_{3t})$. For each solution, move to step 2. After all solutions are traversed, move to Step 3.
Step 2: Compute β' via Equation 10 and insert the tuple $(u_1, u_2, \ldots, u_{3t}, \beta')$ into a table denoted by D_u.
Step 3: Sort the table D_u according to β'.

The online phase. For each value of $(\Delta_{r_0+r_1-1}[1 : 3e], \Delta^S_{r_0+r_1-1})$ computed backwards, we need to find the solutions of $(u_1, u_2, \ldots, u_{3ml})$. The procedure can be stated as follows:

Step 1: Compute $\gamma = \Delta_{r_0+r_1-1}[1 : 3e] | \Delta^S_{r_0+r_1-1}[3m + 1 : n]$ and $\beta = Q_1 \cdot \gamma$ as well as $\beta' = \beta[n - 3m + 3e - \omega + 1, n - 3m + 3e]$.
Step 2: For the computed β', retrieve from D_u the corresponding values of the tuple $(u_1, u_2, \ldots, u_{3t})$. For each retrieved $(u_1, u_2, \ldots, u_{3t})$, move to Step 3.
Step 3: Only $(u_{3t+1}, u_{3t+2}, \ldots, u_{3ml})$ in Equation 9 remain unknown, where $q = 3t$. As Q_0 is in reduced row echelon form, there will be in total $2^{(3ml-3t)-rank(Q_0)} = 2^{(3ml-3t)-(n-3m+3e-\omega)} = 2^{3m(l+1)-3t-n-3e+\omega} = 2^{3mr_1-3t-n-3e+\omega}$ solutions of $(u_{3t+1}, u_{3t+2}, \ldots, u_{3ml})$, which can be easily enumerated. For each solution of $(u_1, u_2, \ldots, u_{3ml})$, the difference transitions in the middle r_1 rounds are fully specified and the correctness can be easily verified via DDT. If it passes the verification, a possible r-round compact differential trail is obtained.

Complexity evaluation. For the offline phase, the time and memory complexity are both $2^{1.86t}$. For the online phase, for each β', there are on average $2^{1.86t}/2^{\omega}$ solutions of $(u_1, u_2, \ldots, u_{3t})$. For each such solution, there are $2^{3mr_1-3t-n-3e+\omega}$ solutions of $(u_{3t+1}, u_{3t+2}, \ldots, u_{3ml})$. In other word, for each given γ, there will be on average $2^{1.86t-\omega+(3mr_1-3t-n-3e+\omega)} = 2^{3mr_1-n-3e-1.14t}$ full solutions of $(u_1, u_2, \ldots, u_{3ml})$.

5.4 How to Choose Parameters

The time complexity and the memory complexity of the offline phase cannot exceed 2^k and therefore we have

$$2^{1.86t} < 2^k. \tag{12}$$

Second, the total number of potentially correct compact differential trails cannot exceed 2^k. Hence,

$$N_d = 2^{1.86m(r_1+r_2)-n} < 2^k. \tag{13}$$

Finally, the time complexity to enumerate the differences cannot exceed 2^k. Note that as we need to compute $\gamma = \Delta_{r_0+r_1-1}[1:3e]|\Delta^S_{r_0+r_1-1}[3m+1:n]$, it is necessary to enumerate the difference backwards for the last r_2 rounds to compute $\Delta^S_{r_0+r_1-1}[3m+1:n]$ and further enumerate the input differences for the first e S-boxes in the (r_0+r_1)-th round to compute $\Delta_{r_0+r_1-1}[1:3e]$. Therefore, it is equivalent to say that $e+mr_2$ S-boxes are taken into account at the backward difference enumeration phase and the time complexity becomes $2^{1.86(mr_2+e)}$. For each γ computed backwards, we need to perform the online phase to retrieve the full solution of $(u_1, u_2, \ldots, u_{3ml})$. Hence, the time complexity to enumerate the difference is

$$\begin{aligned} &max(2^{1.86(mr_2+e)}, 2^{1.86(mr_2+e)+3mr_1-n-3e-1.14t}) \\ =\ &max(2^{1.86(mr_2+e)}, 2^{1.86mr_2+3mr_1-n-1.14e-1.14t}), \end{aligned} \tag{14}$$

which implies

$$1.86(mr_2+e) < k,\ 1.86mr_2 + 3mr_1 - 1.14e - 1.14t < k + n. \tag{15}$$

5.5 Maximizing the Attacked Rounds Using Less Memory

To ensure the memory complexity is less than that of the simple MITM strategy, the following additional constraint should be added:

$$2^{1.86t} < 2^{1.86m(r_1-1)} \rightarrow t < m(r_1-1). \tag{16}$$

We should emphasize that $t \leq m(r_1-1)$ holds because we do not care about the S-boxes in the last round of the middle r_1 rounds. Indeed, we can also apply this to the simple MITM strategy, i.e. ignoring the S-boxes in the last round of the middle r_1 rounds. This is why we use the constraint $2^{1.86t} < 2^{1.86m(r_1-1)}$ rather than $2^{1.86t} < 2^{1.86mr_1}$.

With these constraints in mind, it is possible to discuss Problem 1 and Problem 2.

The parameter $(n, k, m, D) = (128, 128, 1, 1)$**.** In this case, $n - 3m = 125$. First, choose the maximal values for (r_0, r_2), which are $r_0 = 42$ and $r_2 = 68$ based on $r_0 = \lfloor (n/3m) \rfloor$ and $r_2 = \lfloor (k/1.86m) \rfloor$. Therefore, $e = 0$ according to Equation 15. Then, according to Equation 15 and Equation 16, there will be $1.14t > 3r_1 - 127$ and $t < r_1 - 1$. As the memory complexity of the offline phase is $2^{1.86t}$, we expect that t takes the minimal value, i.e. $t = \lceil (3r_1 - 129.52)/1.14 \rceil$. With the constraint $t < r_1 - 1$, the maximal value for r_1 is 68 and $t = 66$. For $r_1 = 68$ and $t = 66$, the memory complexity of our new algebraic MITM strategy is $2^{122.7}$, while it is $2^{124.6}$ for the simple MITM strategy, which shows the advantage of the new technique. When r_1 becomes smaller, the advantage is more clear, as shown in Table 1.

Table 1: Comparison between the memory complexity of the algebraic MITM strategy ($\mathtt{M}_1$) and the simple MITM strategy ($\mathtt{M}_0$) for different r_1.

r_1	≤ 42	43	44	...	61	62	63	64	65	66	67	68
t	0	1	3	...	47	50	53	55	58	61	63	66
$\mathtt{log_2 M_0}$	$1.86(r_1 - 1)$	78.1	79.9	...	111.6	113.4	115.3	117.1	119.0	120.9	122.7	124.6
$\mathtt{log_2 M_1}$	0	1.8	5.5	...	87.4	93	98.5	102.3	107.8	113.4	117.1	122.7

The parameter $(n, k, m, D) = (128, 128, 10, 1)$**.** In this case, $n - 3m = 98$. First, $r_0 = 4$ and $r_2 = 6$ are determined in a similar way. Then, we choose e such that $1.86(mr_2 + e) < k$ and therefore $e = 8$. Finally, we determine r_1 and t according to Equation 15 and Equation 16. Similarly, $t = \lceil (3mr_1 - k - n - 1.14e + 1.86mr_2)/1.14 \rceil = \lceil (3mr_1 - 153.52)/1.14 \rceil$ and $t < m(r_1 - 1)$. Hence, the maximal value for r_1 is 7 and $t = 50$. This implies that our attack requires less memory when $r_1 \leq 7$. Specifically, for $r_1 = 7$, the memory complexity of the simple MITM strategy is $2^{111.6}$, while our new technique only requires 2^{93} memory.

The parameter $(n, k, m, D) = (1024, 128, 1, 1)$**.** For such a parameter, $r_0 = 341$, $r_2 = 68$ and $e = 0$. Based on Equation 12, the maximal value for t is 68. According to Equation 15, we have $1.14t > 3mr_1 - k - n - 1.14e + 1.86mr_2 = 3r_1 - 1025.52$. Therefore, the maximal value for r_1 is 367. In other words, by using $2^{1.86t} = 2^{126.48}$ memory, our attack can reach up to $341 + 68 + 367 = 776$ rounds and the claimed secure number of rounds is exactly 776.

Since there must exist one valid compact differential trails and $N_d = 2^{-214.9}$, after the difference enumeration, there will be only 1 valid compact differential trail surviving. Based on the key-recovery technique in [28], we can recover the correct key from this trail with time complexity much smaller than 2^{128}, i.e. $2^{128} >> T_k$. Hence, even without optimizing the key-recovery technique in [28], according to Equation 14, we have already broken the full rounds of such an instance with time complexity $2^{126.48} + 2^{126.48} = 2^{127.48}$ and memory complexity $2^{126.48}$.

5.6 Advantages of the Algebraic MITM Technique

Based on the above discussions, **Problem 1** and **Problem 2** have been successfully addressed. Indeed, the two problems are the same, which is how to reduce the memory complexity of the simple MITM strategy. For example, with the simple MITM strategy, the attack on the parameter $(n, k, m, D) = (1024, 128, 1, 1)$ with $(r_0, r_1, r_2) = (341, 367, 68)$ will require $2^{1.86r_1} = 2^{682.62}$ memory and it soon becomes ineffective. However, our new technique can reduce this to $2^{126.48}$ and an effective attack is immediately obtained. Compared with the simple MITM strategy to find differential trails, our new MITM technique is applicable to a wide range of parameters, i.e. $n >> k$. Compared with the algebraic technique, this new technique sheds new insight into how to combine the algebraic technique and the usage of memory to increase r_1. Hence, the algebraic MITM technique is more generic and can optimize the memory complexity of the simple MITM strategy over the state-of-the-art.

5.7 Relation to the General Algebraic MITM Attack

In the general framework discussed in Section 3, it is necessary to split the variables into two parts, i.e. I and J. For our new attacks on LowMC, I corresponds to $(u_1, u_2, \ldots, u_{3t})$ and J corresponds to $(u_{3t+1}, u_{3t+2}, \ldots, u_{3ml})$. Our first observation is that there are in total $2^{1.86t}$ rather than 2^{3t} possible values of I due to the nonlinear relations inside I. Moreover, these possible values can be precomputed independent of J, which is why we can split the variables into two sets. Moreover, when I is known and a challenge (γ in Equation 4) is given, J can be computed by solving a linear equation system and the coefficient matrix will not vary for different values of (I, γ), which implies we do not need to store the whole linear equation system for different I. To efficiently determine I for a given γ, we observe that it is possible to first find a deterministic linear equation system only in terms of (I, γ), which is Equation 10 where β' is obtained by applying a fixed linear transform to γ. Then, we can precompute the tuples (I, β') at the offline phase. At the online phase, we thus first apply the fixed linear transform to γ to obtain β', and then retrieve I from the stored table, and finally solve the linear equation system in J according to (I, γ). In a word, due to the deterministic linear equation system in (I, β'), an efficient method to determine which I will lead to the given challenge γ is found, which addresses the most important issue in the general algebraic MITM attack framework.

6 Recovering the Key by Solving Quadratic Equations

Based on the above algebraic MITM strategy, we can increase r_1 while using less memory. As r_1 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 becomes crucial to further optimize T_k.

Our new key-recovery strategy is essentially built on the algebraic technique proposed in [28]. Therefore, we first revisit the technique and then describe how to further optimize it.

6.1 The Algebraic Technique in [28]

The algebraic technique to retrieve the full key from a random r-round compact differential trail is based on the following critical property of the LowMC S-box.

Observation 1 *[28] For each valid non-zero difference transition* $(\Delta x_0, \Delta x_1, \Delta x_2) \rightarrow (\Delta z_0, \Delta z_1, \Delta z_2)$*, 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.*

Example. We give an example for better understanding. If $(\Delta x_0, \Delta x_1, \Delta x_2) = (0, 0, 1)$ and $(\Delta z_0, \Delta z_1, \Delta z_2) = (0, 0, 1)$, there will be $x_0 = 0$ and $x_1 = 0$. In addition, the S-box is freely linearized, i.e. we have $z_0 = 0$, $z_1 = 0$ and $z_2 = x_2$.

Based on this simple property, given a random r-round compact differential trail, the procedure to recover the key can be roughly described[8] as follows, which is further illustrated in Figure 6.

1. Starting from a ciphertext, check the S-boxes in the backward direction round by round and one by one.
   (a) If the S-box is active, write its input as linear expressions in terms of the output and obtain 2 linear equation[9] in terms of the key, i.e. there are two bits conditions on the output to ensure such a difference transition and the S-box is freely linearized based on Observation 1.
   (b) If the S-box is inactive, guess 2 output bits and write the input as linear expressions in the output. From the 2 guessed bits, we again obtain 2 linear equations in the key bits.
   (c) If more than k linear equations are obtained, move to Step 2.
2. After obtaining more than k linear equations, solve the linear equation system to determine the full key and test its correctness via the plaintext-ciphertext pair. If it is the wrong key, try another guess and repeat the same procedure until all possible guesses are traversed.

Although a novel and efficient way is used to process the active S-boxes in the algebraic technique, processing the inactive S-boxes is rather inefficient. We are thus motivated to consider whether there is a more efficient way to handle them. Although the authors also mentioned to introduce intermediate variables to represent the input of the inactive S-box [28], the collected equations are still from the active S-boxes, which makes the complexity evaluation difficult. Hence, they simply adopted the above guess-and-determine method and provided a loose upper bound for the average time complexity to retrieve the full key from a random r-round compact differential trail. While this bound is sufficient to break some instances, we need to further optimize it because $N_d = 2^{1.86m(r_1+r_2)-n}$ will become very huge in Attack-N due to the increase of r_1.

[8] There are some optimizations, but the general idea is still guess-and-determine.

[9] This is because the output can be written as linear expressions in the key bits. Specifically, each S-box is linearized and the round function can be treated as a linear function. Similar explanations also apply to the inactive S-boxes.

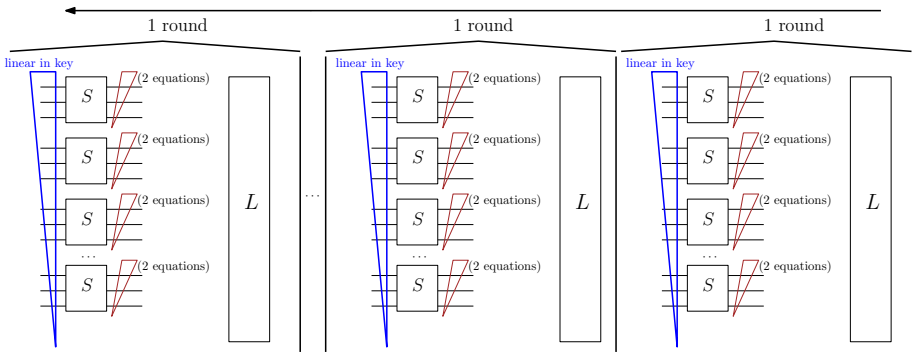

Fig. 6: Illustration of the key-recovery phase in [28]

6.2 A New Method to Handle the Inactive S-boxes

In this part, we show that from a random r-round compact differential trail together with 2 plaintext-ciphertext pairs, it is possible to recover the full key with time complexity 1 and with success rate of about 0.5.

We still follow the general procedure to recover the key described above, apart from using a new method to process the inactive S-boxes. First, let us discuss a useful property of the LowMC S-box. In [29], it is revealed that there are at most 14 linearly independent quadratic equations to describe the LowMC S-box, as specified below:

$$
\begin{aligned}
&z_0 = x_0 \oplus x_1x_2,\ z_1 = x_0 \oplus x_1 \oplus x_0x_2,\ z_2 = x_0 \oplus x_1 \oplus x_2 \oplus x_0x_1,\\
&x_0 = z_0 \oplus z_1 \oplus z_1z_2,\ x_1 = z_1 \oplus z_0z_2,\ x_2 = z_0 \oplus z_1 \oplus z_2 \oplus z_0z_1,\\
&z_0x_1 = x_0x_1 \oplus x_1x_2,\ z_0x_2 = x_0x_2 \oplus x_1x_2,\ z_1x_0 = x_0 \oplus x_0x_1 \oplus x_0x_2,\\
&z_1x_2 = x_1x_2,\ z_2x_0 = x_0 \oplus x_0x_2,\ z_2x_1 = x_1 \oplus x_1x_2,\\
&z_0x_0 \oplus x_0 = z_1x_1 \oplus x_0x_1 \oplus x_1,\ z_1x_1 \oplus x_0x_1 \oplus x_1 = z_2x_2 \oplus x_0x_2 \oplus x_1x_2 \oplus x_2.
\end{aligned}
$$

Note that the above 14 equations can also be found with the algorithm in [17].

These quadratic equations are useful to handle the inactive S-boxes. Specifically, instead of only deriving linear equations from the active S-boxes, we can also derive these quadratic equations from the inactive S-boxes. Hence, the attack procedure can be described as follows once we utilize the last h S-boxes in the last $\lceil h/m \rceil \leq r_1 + r_2$ rounds for the key recovery.

Step 1: Choose a threshold a_{min} and initialize two counters a and b as 0.
Step 2: If there are fewer than a_{min} active S-boxes in these h S-boxes[10], exit and return **Failure**. Otherwise, move to Step 3.

[10] After understanding the ideas, one can observe that this strong condition can be relaxed. More explanations can be referred to Appendix B. We choose this condition mainly for easily bounding the success probability.

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

(a) If the S-box is active, increase a by 1 and write its input as linear expressions in terms of the output and obtain **2 linear equations** in terms of the key and the intermediate variables, i.e. there are two bits conditions on the output to ensure such a difference transition and the S-box is freely linearized based on Observation 1.

(b) If the S-box is inactive, increase b by 1 and introduce 3 intermediate variables to represent its input and obtain **14 quadratic equations** in terms of the input bits and output bits.

(c) If

$$2a \geq k + 3b, \tag{17}$$

or

$$\begin{aligned} &2a < k + 3b, \\ &14b \geq (k + 3b - 2a) + (k + 3b - 2a)(k + 3b - 2a - 1)/2, \end{aligned} \tag{18}$$

move to Step 4.

Step 4: At this step, we have collected $2a$ linear equations and $14b$ quadratic equations in terms of $k+3b$ variables, i.e. the key bits and the intermediate variables. If we reach this step according to Equation 17, we only need to solve $2a$ linear equations to uniquely determine the k-bit key. If we reach this step according to Equation 18, we need to first perform the Gaussian elimination on the $2a$ linear equations to obtain $k + 3b - 2a$ free variables. Then, we rewrite the $14b$ quadratic equations in these $k + 3b - 2a$ free variables and perform the Gaussian elimination on it, where each quadratic term is viewed as a new variable[11]. As Equation 18 holds, we can expect to obtain a unique solution to these $k+3b-2a$ free variables and then compute the remaining variables according to the $2a$ linear equations. For both cases, we obtain a unique solution to the full key and the correctness can be easily verified via a plaintext-ciphertext pair. If it is the correct key, output it and return **Success**.

For better comparison, the new algebraic key-recovery technique is depicted in Figure 7.

How to choose (a_{min}, h)**.** According to the above procedure, we start solving the equation system to recover the key only when (a, b) satisfy some constraints. In addition, before moving to Step 3, we will directly reject some compact differential trails by counting the number of active S-boxes among the last h S-boxes. It is possible that the correct differential trail is the rejected one and our attack will then fail to recover the key. Once moving to Step 3, we expect that either Equation 17 or Equation 18 must hold. To achieve a high success

[11] This is called the linearization technique.

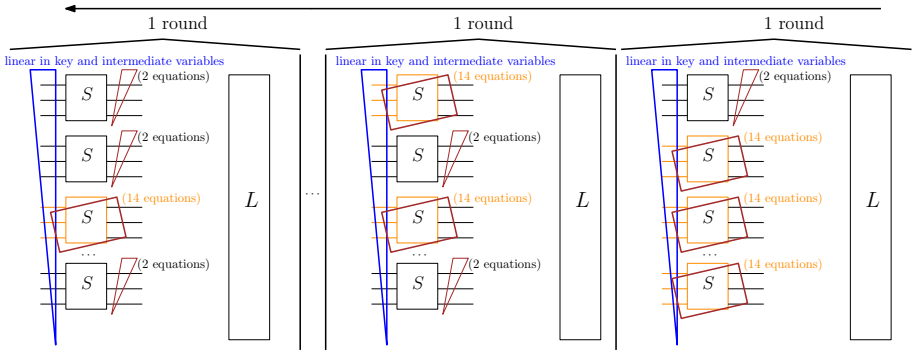

Fig. 7: Illustration of the new key-recovery phase, where the inactive S-boxes are colored in orange

probability and to make either Equation 17 or Equation 18 hold, we thus add the following constraints on (a_{min}, h):

$$\begin{aligned} a_{min} &= \lceil (7h)/8 \rceil, \\ 14h - 14a_{min} &\geq (k + 3h - 5a_{min}) + (k + 3h - 5a_{min})(k + 3h - 5a_{min} - 1)/2, \\ k + 3h - 5a_{min} &> 0. \end{aligned}$$

Suppose there are a' active S-boxes among the h S-boxes. The first constraint is based on the well-known statistical property that when $a_{min} \approx (7h)/8$, there is $Pr[a' \geq a_{min}] = \sum_{i=a_{min}}^{h} \binom{h}{i} \times (7/8)^i \times (1/8)^{h-i} \approx 0.5$.

The last two constraints can ensure that when the number of free variables is a positive integer, i.e. there are exactly a_{min} active S-boxes among the last h S-boxes and $(k + 3h - 5a_{min}) > 0$, the quadratic equation system can still be efficiently solved with the linearization technique. Obviously, if $(k+3h-5a_{min}) \leq 0 \rightarrow 2a_{min} \geq k + 3(h - a_{min})$, all the unknowns can be directly computed by solving the $2a_{min}$ linear equations. Then, for any a' satisfying $h \geq a' \geq a_{min}$, either Equation 17 or Equation 18 must hold[12] if moving to Step 3. Specifically, the key can be directly recovered by solving an equation system.

Therefore, with the above constraints on (a_{min}, h), we can ensure a success probability[13] of about 0.5 to recover the key.

[12] For the proof of this claim, one may refer to Appendix C, which is indeed very intuitive.

[13] The computed probability is just a lower bound, as explained in Appendix B. This is also intuitive. Note that one may choose different (a_{min}, h) such that $a_{min} < \lceil (7h)/8 \rceil$ as long as $\lceil h/m \rceil \leq r_1 + r_2$ to increase the success probability and follow the same procedure to recover the key at the cost to solve a relatively large system of equations. Choosing $a_{min} = \lceil (7h)/8 \rceil$ is mainly to utilize a simple statistical property to give a sufficiently good lower bound for the probability and a good upper bound for T_k. As N_d becomes very large and $N_d T_k < 2^k$ has to hold, the upper bound for T_k becomes sensitive and it should be kept small.

Some concrete choices for (h, a_{min}) for different key sizes are specified in Table 2. For example, the best choice is $(h, a_{min}) = (81, 71)$ for $k = 128$, which will result in $Pr[a' \geq a_{min}] \approx 0.56$. Then, it is required to solve at most 142 linear equations and at most 140 quadratic equations[12], which corresponds to the worst case when $a' = a_{min} = 71$. We simply estimate the cost to solve these equation systems as $142^3/(rn^2)$ times of LowMC encryptions because each LowMC encryption costs about $2rn^2$ binary operations.

Table 2: Choices for (h, a_{min}) for different k

k	h	a_{min}	Pro.	linear	quadratic	cost (T_k)
128	81	71	0.56	142	140	$142^3/(rn^2)$
192	124	109	0.51	218	210	$218^3/(rn^2)$
256	169	148	0.54	296	294	$296^3/(rn^2)$

Comparison. Compared to the algebraic key-recovery technique in [28], the new method requires no guessing phase and the key is directly computed via solving a quadratic boolean equation system and a linear equation system. This is based on a new way to handle the inactive S-box, i.e. we exploit the nonlinear relations between its input and output rather than linearize it by guessing some input or output bits as in [28]. The only drawback is that this new technique cannot work for an arbitrarily given compact differential trail, i.e. its success rate is about 0.5. However, 0.5 is high enough to claim an effective attack.

6.3 Recovering the Key in the Extended Attack Framework

In the extended framework [28], by using sufficiently many plaintext pairs, i.e. when $D >> 1$, it is possible to find a pair of plaintext such that there is no active S-box in the last $r_3 = \lfloor (D-1)/(3m) \rfloor$ rounds either, as depicted in Figure 8. The time complexity and data complexity of this phase are both 2^{3mr_3+1} because we need to try 2^{3mr_3} pairs of plaintexts. As all the S-boxes are inactive in the last r_3 rounds, the constraints on (r_1, r_2) will not change, i.e. we can still choose (r_1, r_2) based on the constraints specified in Section 5.4. However, the key-recovery phase will change as the last mr_3 S-boxes are always inactive. Hence, it is required to modify the constraints on (a_{min}, h) in the extended framework in order to efficiently recover the key.

Modifying the constraints is not difficult as we can equivalently consider the case where the last mr_3 S-boxes are always inactive in the last h S-boxes used for key recovery. The modified constraints are specified below:

$$\begin{aligned} h' &= h - mr_3, \\ a_{min} &= \lceil (7h')/8 \rceil, \end{aligned}$$

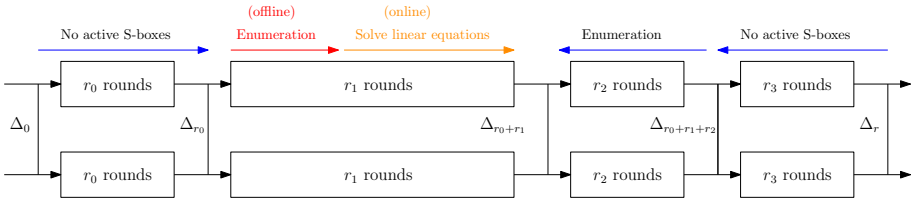

Fig. 8: The extended attack framework [28] embedded with the algebraic MITM strategy

$$14h - 14a_{min} \geq (k + 3h - 5a_{min}) + (k + 3h - 5a_{min})(k + 3h - 5a_{min} - 1)/2.$$

The success probability is then computed with $\sum_{i=a_{min}}^{h'} \binom{h'}{i} \times (7/8)^i \times (1/8)^{h'-i}$.

Table 3: Choices for (h', a_{min}) for different (mr_3, k)

k	mr_3	h'	a_{min}	Pro.	linear	quadratic	cost (T_k)
128	20	114	100	0.54	228	476	$476^3/(rn^2)$
128	21	116	102	0.51	232	490	$490^3/(rn^2)$
256	20	203	178	0.52	406	630	$630^3/(rn^2)$
256	21	205	180	0.50	410	644	$644^3/(rn^2)$

In Table 3, we provide the accurate values of (h', a_{min}) for some (mr_3, k) that are relevant to our attacks on LowMC-M. The explanation of this table can be referred to that for Table 2. Note that in this framework, $\lceil h'/m \rceil \leq r_1 + r_2$ has to hold.

7 Improved Attacks on LowMC and LowMC-M

With the above new techniques, we could significantly improve the number of attacked rounds for both LowMC and LowMC-M. The correctness of these techniques has been verified by experiments, as shown in Appendix A.

The attacks on LowMC. For the attacks on LowMC, we only consider the parameters where $D = 1$, i.e. the data complexity is 2. When $D >> 1$, we can trivially use the extended attack framework and we only need to slightly modify the key-recovery phase. Indeed, the attacks on LowMC-M are achieved under the extended attack framework. As already mentioned, the attack consists of two phases: the difference enumeration phase and the key-recovery phase. For an attack on $r = r_0 + r_1 + r_2$ rounds of LowMC (refer to Figure 5), we first determine r_0 with $r_0 = \lfloor n/(3m) \rfloor$. Then, we determine (r_1, r_2) as well as (t, e) based on the constraints Equation 12, Equation 13 and Equation 15. Examples have already been given in Section 5.5.

The time complexity of the difference enumeration phase is

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

and the memory complexity is $2^{1.86t}$. The time complexity of the key-recovery phase is $N_d T_k = 2^{1.86m(r_1+r_2)-n} T_k$, where the accurate estimation of T_k has been explained at Section 6.2. The whole time complexity T_w is thus

$$T_w = 2^{1.86m(r_1+r_2)-n} T_k + 2^{1.86t} + max(2^{1.86(mr_2+e)}, 2^{1.86mr_2+3mr_1-n-1.14e-1.14t}).$$

For attacks with success probability of about 0.5, $T_w < 2^{k-1}$ should hold. For our attacks on the parameters where $n >> k$ in Table 4, the success probability is 1 because we simply use the key-recovery technique in [29]. In this case, $2^{1.86t} + max(2^{1.86(mr_2+e)}, 2^{1.86mr_2+3mr_1-n-1.14e-1.14t})$ dominates T_w because only 1 differential trail will survive after the difference enumeration phase, i.e. $1 >> 2^{1.86m(r_1+r_2)-n}$. Our results are summarized in Table 4.

Table 4: Summary of the attacks on LowMC, where D, T, M, Pro. and $R-r$ represent the $\log_2$ data/time/memory complexity, success probability and security margin, respectively. Moreover, $-$ represents negligible memory.

n	k	m	D	R	r_0	r_1	r_2	t	e	r	D	T	M	Pro.	$R-r$	Ref.
128	128	1	1	182	42	43	67	0	0	152	1	124.62	–	1	30	[28]
					42	51	59	21	0	152	1	110.8	39.06	0.56	30	this paper
					42	68	67	66	0	177	1	125.38	122.76	0.56	5	this paper
128	128	10	1	20	4	5	6	0	0	15	1	122.8	–	1	5	[28]
					4	7	6	53	7	17	1	125.2	98.58	0.56	3	this paper
192	192	1	1	273	64	64	101	0	0	229	1	187.86	–	1	44	[28]
					64	101	102	98	0	267	1	189.72	182.28	0.51	6	this paper
192	192	10	1	30	6	7	10	0	0	23	1	186	–	1	7	[28]
					6	9	10	67	2	25	1	189.72	124.62	0.51	5	this paper
256	256	1	1	363	85	86	137	0	0	306	1	254.82	–	1	57	[28]
					85	136	136	133	0	357	1	253.34	247.38	0.54	9	this paper
256	256	10	1	38	8	9	13	0	0	30	1	241.8	–	1	8	[28]
					8	13	13	101	6	34	1	253.82	187.86	0.54	4	this paper
1024	128	1	1	776	341	342	66	0	0	749	1	122.76	–	1	27	[28]
					341	367	68	68	0	776	1	127.48	126.48	1	0	this paper
1024	256	1	1	819	341	342	136	0	0	819	1	253	–	1	0	[28]
					341	393	136	136	0	870	1	253.96	252.96	1	−51	this paper

The attacks on LowMC-M. LowMC-M is almost the same as LowMC. The only difference is that after the key addition operation, there is a subtweak addition operation and the subtweak can be derived from a public tweak. It has been studied in [14] that by exploiting the freedom of the tweak, in the difference

enumeration attack, r_0 can be increased to $\lfloor (2k+n)/(3m) \rfloor$ by finding a proper tweak pair with time complexity $2^{(3mr_0-n)/2}$, i.e. the first r_0 rounds contain no active S-boxes. A detailed explanation can be referred to [28]. Moreover, $D = 64$ in all the specified parameters for LowMC-M. Hence we utilize the extended difference enumeration attack framework depicted in Figure 8 where we choose $mr_3 \in \{20, 21\}$ to make full use of the allowed data complexity, i.e. the data complexity of our attacks on LowMC-M is either 2^{61} or 2^{64} and the last r_3 rounds contain no active S-boxes. As for (r_1, r_2), we determine their values and estimate the corresponding time/memory complexity as in the above attacks on LowMC. Due to the costly phase to find a proper tweak pair, the whole time complexity becomes $2^{(3mr_0-n)/2} + T_w$. The improved attacks on LowMC-M are summarized in Table 5. Note we consider the latest version of LowMC-M, which only differs from the original LowMC-M [32] in the number of rounds.

Table 5: Summary of the attacks on LowMC-M

n	k	m	D	R	r_0	r_1	r_2	r_3	t	e	r	D	T	M	Pro.	$R-r$	Ref.
128	128	1	64	294	122	43	64	21	0	0	250	64	120	–	1	44	[28]
					124	66	66	21	60	0	277	64	124.36	111.6	0.51	17	this paper
128	128	2	64	147	61	22	32	10	0	0	125	61	120	–	1	22	[28]
					62	33	33	10	60	0	138	61	124.36	111.6	0.52	9	this paper
128	128	3	64	99	40	15	21	7	0	0	83	64	118.18	–	1	16	[28]
					41	22	22	7	60	0	92	64	124.36	111.6	0.51	7	this paper
128	128	10	64	32	12	5	6	2	0	0	25	61	118	–	1	7	[28]
					12	7	6	2	53	7	27	61	125.2	98.58	0.52	5	this paper
256	256	1	64	555	253	86	136	21	0	0	496	64	252.96	–	1	59	[28]
					253	136	136	21	133	0	546	64	253.34	247.38	0.50	9	this paper
256	256	3	64	186	83	29	45	7	0	0	164	64	250.1	–	1	22	[28]
					84	45	45	7	129	1	181	64	252.96	239.94	0.50	5	this paper
256	256	20	64	30	12	5	6	1	0	0	24	61	232	–	1	6	[28]
					12	7	6	1	115	15	26	61	251.1	213.9	0.52	4	this paper

8 Conclusion

We propose a simple yet novel technique called algebraic MITM attack to analyze LowMC. This new technique can better capture the feature of partial nonlinear layers. Since using partial nonlinear layers is a relatively new design strategy, developing new techniques to understand its security is both important and meaningful. As a consequence of this new technique and an extremely optimized algebraic key-recovery technique for LowMC, the attacks on LowMC and LowMC-M are significantly improved. Regarding the LowMC S-box, some new algebraic properties are discovered, though they are not exploited in a pure

"algebraic" way in this work. It is interesting to investigate whether they can be used to mount another type of algebraic attack on LowMC and whether studying similar equations for DDT is useful for differential attacks on other ciphers.

References

1. https://csrc.nist.gov/projects/post-quantum-cryptography.
2. Reference Code, 2017. https://github.com/LowMC/lowmc.
3. The Picnic signature algorithm specification, 2019. Available at https://microsoft.github.io/Picnic/,.
4. M. R. Albrecht, C. Cid, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, and M. Schofnegger. Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 371–397. Springer, 2019.
5. M. R. Albrecht, L. Grassi, L. Perrin, S. Ramacher, C. Rechberger, D. Rotaru, A. Roy, and M. Schofnegger. Feistel Structures for MPC, and More. In *ESORICS (2)*, volume 11736 of *Lecture Notes in Computer Science*, pages 151–171. Springer, 2019.
6. M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, 2016.
7. M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.
8. A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. *IACR Trans. Symmetric Cryptol.*, 2020(3):1–45, 2020.
9. T. Ashur and S. Dhooghe. MARVELlous: a STARK-Friendly Family of Cryptographic Primitives. Cryptology ePrint Archive, Report 2018/1098, 2018. https://eprint.iacr.org/2018/1098.
10. S. Banik, K. Barooti, F. B. Durak, and S. Vaudenay. Cryptanalysis of LowMC instances using single plaintext/ciphertext pair. *IACR Trans. Symmetric Cryptol.*, 2020(4):130–146, 2020.
11. S. Banik, K. Barooti, S. Vaudenay, and H. Yan. New Attacks on LowMC Instances with a Single Plaintext/Ciphertext Pair. In *ASIACRYPT (1)*, volume 13090 of *Lecture Notes in Computer Science*, pages 303–331. Springer, 2021.
12. A. Bar-On, I. Dinur, O. Dunkelman, V. Lallemand, N. Keller, and B. Tsaban. Cryptanalysis of SP Networks with Partial Non-Linear Layers. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 315–342. Springer, 2015.
13. T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems. In *CRYPTO (3)*, volume 12172 of *Lecture Notes in Computer Science*, pages 299–328. Springer, 2020.
14. T. Beyne and C. Li. Cryptanalysis of the MALICIOUS Framework. Report 2020/1032, 2020. https://ia.cr/2020/1032.

15. A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. *J. Cryptol.*, 31(3):885–916, 2018.
16. M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives. In *CCS*, pages 1825–1842. ACM, 2017.
17. N. T. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.
18. I. Dinur. Cryptanalytic Applications of the Polynomial Method for Solving Multivariate Equation Systems over GF(2). In *EUROCRYPT (1)*, volume 12696 of *Lecture Notes in Computer Science*, pages 374–403. Springer, 2021.
19. I. Dinur, Y. Liu, W. Meier, and Q. Wang. Optimized Interpolation Attacks on LowMC. In *ASIACRYPT (2)*, volume 9453 of *Lecture Notes in Computer Science*, pages 535–560. Springer, 2015.
20. C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger. Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In *CRYPTO (1)*, volume 10991 of *Lecture Notes in Computer Science*, pages 662–692. Springer, 2018.
21. C. Dobraunig, M. Eichlseder, and F. Mendel. Higher-Order Cryptanalysis of LowMC. In *ICISC*, volume 9558 of *Lecture Notes in Computer Science*, pages 87–101. Springer, 2015.
22. C. Dobraunig, L. Grassi, A. Guinet, and D. Kuijsters. Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields. In *EUROCRYPT (2)*, volume 12697 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2021.
23. S. Duval, V. Lallemand, and Y. Rotella. Cryptanalysis of the FLIP Family of Stream Ciphers. In *CRYPTO (1)*, volume 9814 of *Lecture Notes in Computer Science*, pages 457–475. Springer, 2016.
24. M. Eichlseder, L. Grassi, R. Lüftenegger, M. Øygarden, C. Rechberger, M. Schofnegger, and Q. Wang. An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC. In *ASIACRYPT (1)*, volume 12491 of *Lecture Notes in Computer Science*, pages 477–506. Springer, 2020.
25. L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *USENIX Security Symposium*, pages 519–535. USENIX Association, 2021.
26. L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In *EUROCRYPT (2)*, volume 12106 of *Lecture Notes in Computer Science*, pages 674–704. Springer, 2020.
27. T. Jakobsen and L. R. Knudsen. The Interpolation Attack on Block Ciphers. In *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 1997.
28. F. Liu, T. Isobe, and W. Meier. Cryptanalysis of Full LowMC and LowMC-M with Algebraic Techniques. In *CRYPTO (3)*, volume 12827 of *Lecture Notes in Computer Science*, pages 368–401. Springer, 2021.
29. F. Liu, T. Isobe, and W. Meier. Low-Memory Algebraic Attacks on Round-Reduced LowMC. Report 2021/255, 2021. `https://ia.cr/2021/255`.
30. F. Liu, S. Sarkar, W. Meier, and T. Isobe. Algebraic Attacks on Rasta and Dasta Using Low-Degree Equations. In *ASIACRYPT (1)*, volume 13090 of *Lecture Notes in Computer Science*, pages 214–240. Springer, 2021.

31. P. Méaux, A. Journault, F. Standaert, and C. Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In *EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 311–343. Springer, 2016.
32. T. Peyrin and H. Wang. The MALICIOUS Framework: Embedding Backdoors into Tweakable Block Ciphers. In *CRYPTO (3)*, volume 12172 of *Lecture Notes in Computer Science*, pages 249–278. Springer, 2020.
33. C. Rechberger, H. Soleimany, and T. Tiessen. Cryptanalysis of Low-Data Instances of Full LowMCv2. *IACR Trans. Symmetric Cryptol.*, 2018(3):163–181, 2018.

A Experimental Verifications

In our experiments, the concrete LowMC instances are generated with the official reference code [2].

Verifying the algebraic difference enumeration. Considering the cost of the matrix multiplication in the difference enumeration phase, for efficient verifications, we choose the parameters such that the time complexity of this phase is about 2^{25}. Therefore, we choose to perform experiments on the parameter $(n, k, m, r) = (128, 128, 1, 103)$. For such a parameter, we choose $r_0 = 128/3 = 42$, $r_2 = 13$, $r_1 = 48$ and $e = 0$. Then, to keep $2^{1.86r_2+3r_1-n-1.14t-1.14e}$ be about 2^{25}, we choose $t = 13$. Hence, the theoretical memory complexity is $2^{1.86t} = 2^{24.18}$ and the theoretical time complexity to enumerate the differences backwards is

$$max(2^{1.86r_2}, 2^{1.86r_2+3r_1-n-1.14t-1.14e}) = 2^{25.36}.$$

For such a configuration, it is necessary to introduce $3 \times (r_1 - 1) = 141$ variables $(u_1, u_2, \ldots, u_{141})$ to represent the output differences of the S-boxes in the middle $r_1 - 1$ rounds. First, we construct the matrices M, M_1, Q_0, Q_1 and P_0 according to the generated LowMC instance [2], the sizes of which are 125×141, 125×102, 125×102 and 125×125, respectively. It is found that $\omega = n - 3m - rank(Q_0) = 128 - 3 - 102 = 23$ and hence we obtain 23 linear equations only in terms of $(u_1, u_2, \ldots, u_{39})$ and $(\beta_{103}, \beta_{104}, \ldots, \beta_{125})$.

For the offline phase, according to the experiments, the size of the table D_u is $17134432 \approx 2^{24.03}$, which is almost the same as the expected value $2^{1.86t} = 2^{24.18}$. As β' is a 23-bit value, each β' will correspond to about $2^{1.18}$ different values of $(u_1, u_2, \ldots, u_{39})$ in D_u.

At the online phase, for each computed γ in the backward direction, we first compute β' according to Equation 8 and Equation 11. Then, retrieve the corresponding $(u_1, u_2, \ldots, u_{39})$ from D_u according to β'. Finally, determine the remaining unknowns $(u_{40}, u_{41}, \ldots, u_{141})$ by solving Equation 9, which can be efficiently solved as Q_0 is in reduced row echelon form and $rank(Q_0) = 102$. In this way, the difference transitions in the middle r_1 rounds are fully known and their correctness can be easily verified via DDT. After the online phase, we succeed in recovering all the possible compact differential trails with time complexity of about $2^{25.45}$, which is almost consistent with the theoretical value $2^{24.18+1.18} = 2^{25.36}$.

Verifying the optimized key-recovery phase. There are two main concerns regarding the optimized key-recovery phase. First, what is the actual success probability? Second, can the key be really efficiently computed via solving an overdefined system of quadratic equations with the linearization technique? To deal with these concerns, we choose to perform experiments on the parameter $(n, k, m, r) = (128, 128, 1, 177)$. In this case, $r_0 = 42$ and $r_1 + r_2 = 135 > 81$.

For the success probability, we randomly choose 10000 plaintext pairs such that there is no difference in the first 42 rounds. For each plaintext pair, we record the corresponding r-round differential trail by tracing the encryption phase and count the number of active S-boxes in the last 81 rounds. Finally, we compute the number of plaintext pairs denoted by N_p such that the number of active S-boxes in the last 81 rounds is not smaller than 71. It is found that $N_p/10000 \approx 0.56$, which means the success probability is correct.

To verify the correctness of the key recovery, for each recorded r-round differential trail where there are at least 71 active S-boxes in the last 81 rounds, we first construct the corresponding overdefined system of quadratic equations and then solve it with the linearization technique. It is found that the key can be correctly recovered, thus demonstrating the correctness of the optimized key-recovery strategy.

B More Explanations for the Success Probability

In our key-recovery procedure, it is required to have at least a_{min} active S-boxes among the last h S-boxes. Once this condition is not satisfied, we reject the corresponding trail. Here, we show that this is a strong condition and even if there are fewer than a_{min} active S-boxes, it is still possible to move to Step 4 of our key-recovery procedure.

Specifically, suppose we still move to Step 3 of the key-recovery procedure even if the number of active S-boxes in the given trail is smaller than a_{min}. Note that at Step 3, we consider the S-box one by one and round by round. Then, it is possible that after the first $h - j$ S-boxes in the backward direction are considered, Equation 17 or Equation 18 holds. Specifically, in this case, we have

$$a + b = h - j$$

and either Equation 17 or Equation 18 holds. In this way, whether the last j S-boxes among the h S-boxes in the backward direction are active or not will not affect the key recovery. Supposing there are a' active S-boxes among these last j S-boxes, even if $a + a' < a_{min}$, the key can still be recovered by directly solving equations, though the total number $a + a'$ of active S-boxes is smaller than a_{min}. Hence, the condition is a strong one. Moreover, this also implies that our computed success probability is just a lower bound.

In the following, we provide a more accurate procedure to judge whether it is possible to recover the key from a given trail by directly solving equations. For simplicity, let us consider the case $m = 1$. The cases $m > 1$ can be processed in a similar way.

Step 1. Initialize a counter B with $\lceil k/2 \rceil$ as we need at least $\lceil k/2 \rceil$ S-boxes.
Step 2. If $B > h$, exit. Otherwise, compute the number of active S-boxes among the first B S-boxes in the backward direction and denote it by a. Then, let $b = B - a$ and move to Step 3.
Step 3. If Equation 17 or Equation 18 holds, return **Feasible** and exit. Otherwise, increase B by 1 and move to Step 2.

If the procedure exits according to the condition $B > h$, it means we cannot efficiently compute the key by solving nonlinear equations with the linearization technique. Otherwise, we can always efficiently recover the key.

Although the above procedure is more accurate, computing the success probability that **Feasible** is returned seems difficult. To compute it, we can utilize the conditional probability. Denote the event that **Feasible** is returned at $B = i$ by A_i and the event that **Feasible** is not returned at $B = i$ by $\overline{A_i}$. Let $d' = \lceil k/2 \rceil$. Then, the success probability can be expressed as

$$Pr[A_{d'}] + Pr[A_{d'+1}|\overline{A_{d'}}] + Pr[A_{d'+2}|\overline{A_{d'}}\ \overline{A_{d'}+1}] + \ldots +$$
$$Pr[A_{d'+i}|\overline{A_{d'}}\ \overline{A_{d'+1}} \ldots \overline{A_{d'+i-1}}] + \ldots + Pr[A_h|\overline{A_{d'}}\ \overline{A_{d'+1}} \ldots \overline{A_{h-1}}].$$

This is difficult to compute and is obviously not as intuitive as our simple way to exploit a well-known statistical property.

C More Explanations for the Key-recovery Phase

After fixing (h, a_{min}), we claim that when $a > a_{min}$, either Equation 17 or Equation 18 must hold if moving to Step 3. Moreover, in the time complexity evaluation, we claim that we only need to use at most $2a_{min}$ linear equations and at most $14(h - a_{min})$ quadratic equations to recover the unknowns. Here, we give a simple proof for these claims.

First, according to our way to choose (h, a_{min}), when $H = k+3h-5a_{min} > 0$, there is

$$14h - 14a_{min} \geq H + H(H-1)/2.$$

Note that when $H \leq 0$, we can directly solve all the $2a_{min}$ linear equations to recover all the unknowns since $2a_{min} \geq k + 3(h - a_{min})$.

Given a satisfying $a > a_{min}$, let $a = a_{min} + i$ $(i > 0)$. Then, there will be $h - a = h - a_{min} - i$ inactive S-boxes among the last h S-boxes. In other words, there will be $k + 3(h - a_{min} - i)$ unknowns in the constructed equation system. Although we can collect $2(a_{min} + i)$ linear equations, we only use $2a_{min}$ linear equations for this case and ignore the remaining $2i$ useful linear equations. In this way, we need to prove either

$$2a_{min} \geq k + 3(h - a_{min} - i) \rightarrow H - 3i \leq 0$$

or

$$H - 3i > 0,$$

$$14h - 14(a_{min} + i) \geq (H - 3i) + (H - 3i)(H - 1 - 3i)/2.$$

Then, it suffices to only prove

$$14h - 14(a_{min} + i) \geq (H - 3i) + (H - 3i)(H - 1 - 3i)/2$$

when $H - 3i > 0$.

Then, it suffices to prove

$$-14i \geq -3i - 3Hi + \frac{3i}{2} + \frac{9i^2}{2} \rightarrow H \geq \frac{25}{6} + \frac{3i}{2}.$$

As $H > 3i$ and i is a positive integer, it then suffices to prove

$$3i \geq \frac{25}{6} + \frac{3i}{2} \rightarrow i \geq 3.$$

This means when $i \geq 3$, we can only use at most $2a_{min}$ linear equations and $14(h - a_{min}) - 14i$ quadratic equations to compute all the $k + 3(h - a_{min} - i)$ unknowns with the linearization technique. In other words, the worst case is still $a = a_{min}$ where we need to use $2a_{min}$ linear equations and $14(h - a_{min})$ quadratic equations.

Then, we are only left with the cases $i = 1$ and $i = 2$. It is easy to observe that the above deduction is not tight. Therefore, it is still possible that when $i = 1$ and $i = 2$, we can also only use at most $2a_{min}$ linear equations and $14(h - a_{min}) - 14i$ quadratic equations. Specifically, for a concrete choice of (h, a_{min}), we can simply check whether

$$14h - 14(a_{min} + i) \geq (H - 3i) + (H - 3i)(H - 1 - 3i)/2$$

holds for $i \in \{1, 2\}$, where $H = k + 3h - 5a_{min}$. This is equivalent to checking

$$14(h - a_{min}) \geq H + H(H - 1)/2 + 17 - 3H,$$
$$14(h - a_{min}) \geq H + H(H - 1)/2 + 43 - 6H.$$

It is found that $17 - 3H < 0$ and $43 - 6H < 0$ for all the choices specified in Table 2 and Table 3, i.e. $H > 8$.

All in all, when $a > a_{min}$, either Equation 17 or Equation 18 must hold if moving to Step 3. In addition, we need to use at most $2a_{min}$ linear equations and $14(h - a_{min}) - 14i$ quadratic equations to compute all the $k + 3(h - a_{min} - i)$ unknowns with the linearization technique.