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

Xingwei Ren[1,2] Yongqiang Li[1,2(✉)], and Mingsheng Wang[1,2]

[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
renxingwei@iie.ac.cn, liyongqiang@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
wangmingsheng@iie.ac.cn

Abstract. Recently, Picnic3 has introduced several alternative LowMC instances, which prompts the cryptanalysis competition for LowMC. In this paper, we provide new solutions to the competition with full S-box layers under single-data complexity. First, we present a new guess-and-determine attack framework that achieves the best trade-off in complexity, while effectively enhancing two algorithms applicable to 2-round LowMC cryptanalysis. Next, we present a new meet-in-the-middle attack framework for 2-/3-round LowMC, which can gradually reduce the number of variables and narrow down the range of candidate keys in stages. As a result, our 3-stage MITM attacks have both lower time complexity and memory complexity than the best previous 2-round attacks proposed by Banik et al. at ASIACRYPT 2021, with memory reduced drastically by a factor of $2^{29.7} \sim 2^{70.4}$.

Keywords: LowMC · Picnic · Algebraic attack · MITM · Low memory

1 Introduction

LowMC [1] is a family of block ciphers based on the flexible substitution permutation network (SPN) structure, first proposed by Albrecht et al. at EUROCRYPT 2015. It allows users to independently customize various parameters, including the block size n, the key size k, the number of S-boxes s in the nonlinear layer and the allowed data complexity D of attacks. LowMC employs 3-bit S-boxes, which are specifically designed to achieve low multiplicative complexity.

Since proposed, LowMC instances with partial S-box layers (e.g. $s = 1$) are used in security protocols such as secure multi-party computation (MPC), fully homomorphic encryption (FHE), and zero-knowledge proofs (ZK). Moreover, LowMC has demonstrated its effectiveness within signature schemes and the ability to impact the size of the resulting signature. Compared to standard functions, employing LowMC can reduce the signature size by approximately one order of magnitude [9]. With development, there have been a number of such novel primitives [21,2,8,12,3,14,15].

Following the proposal of LowMCv1 (with the initial version of the round calculation formula), several related cryptanalyses were conducted. Notably, Dobraunig et al. [13] introduced a high-order differential attack, and Dinur et al. [11] presented an interpolation attack. In response to these attacks, the design team upgraded LowMCv1 to LowMCv2 [2] by revising the round calculation formula. At ToSC 2018, Rechberger et al. [23] conducted a low-data cryptanalysis of LowMCv2, focusing on specific partial nonlinear layers. They introduced the framework of difference enumeration techniques based on meet-in-the-middle (MITM) principles, which breached the prescribed security boundaries and directly prompted the evolution of LowMCv2 into LowMCv3. Subsequently, Liu et al. [17,19], Qiao et al. [22], and Sun et al. [24] gradually improved the techniques, leading to the development of a new difference enumeration framework and key recovery techniques. Note that the above cryptanalysis of LowMC was carried out in a known plaintext attack mode, requiring a certain number of plaintexts, i.e. data complexity $D > 1$.

One significant application of LowMC is as the underlying block cipher for the Picnic signature scheme [9]. Picnic is a highly adaptable post-quantum signature scheme currently in the third round of NIST post-quantum cryptography (PQC) standardization process. Now, Picnic3 [16] employs LowMCv3[3] for instantiation and introduces several instances with full S-box layers, which can achieve faster signing and verification speeds while the signature size is almost kept the same compared to the previous version. Unlike previous cryptanalysis, the attacker can only choose a single known plaintext-ciphertext pair (P, C) to recover the key under the Picnic setting. This is because, in the instantiation of Picnic, the public key (verification key) is equivalent to the pair (P, C) of LowMC, while the secret key (signing key) corresponds to the encryption key K of LowMC. In other words, the attacker can only perform a key recovery attack on LowMC with data complexity $D = 1$ to genuinely jeopardize the security of Picnic. It presents an interesting avenue for cryptanalysis.

1.1 Previous Work

In May 2020, Rechberger et al. initiated a cryptanalysis competition[4] for LowMC under the Picnic setting, aimed at exploring key recovery attacks using a single known plaintext-ciphertext pair. The challenge contains 9 instances, including 3 instances with full S-box layers and 6 instances with partial S-box layers:

- $(n, k, s) = (129, 129, 43)$, $(192, 192, 64)$, and $(255, 255, 85)$.
- $(n, k, s) = (128, 128, 1)$, $(192, 192, 1)$, and $(256, 256, 1)$.
- $(n, k, s) = (128, 128, 10)$, $(192, 192, 10)$, and $(256, 256, 10)$.

Note that the number of rounds r for instances with full S-box layers is either 2, 3 or 4 and for instances with partial S-box layers can vary between $0.8 \times \lfloor \frac{n}{s} \rfloor$, $\lfloor \frac{n}{s} \rfloor$ and $1.2 \times \lfloor \frac{n}{s} \rfloor$. If these values are not integers, the number of rounds is rounded

[3] In the following, LowMCv3 is referred to as LowMC for short.
[4] https://lowmcchallenge.github.io/

up to the nearest higher integer. Furthermore, Picnic3 [16] recommends that 3 instances with full S-box layers: $(n, k, s, r) = (129, 129, 43, 4)$, $(192, 192, 64, 4)$ and $(255, 255, 85, 4)$.

At ToSC 2020, Banik et al. [5] first proposed a linearization technique for the LowMC S-box and a key recovery attack employing MITM and guess-and-determine (GnD) methods. They successfully solved the 2-round LowMC instances with full S-box layers and the $0.8 \times \lfloor \frac{n}{s} \rfloor$-round LowMC instances with partial S-box layers. Following this work, at ASIACRYPT 2021, Banik et al. [6] improved their attack methodology and presented a 2-stage MITM attack with a Gray-codes based approach, which can solve more LowMC challenge instances. Moreover, their results for instances with partial S-box layers were improved upon in [25].

At EUROCRYPT 2021, Dinur [10] presented an efficient algorithm for solving the polynomial system of degree d with n variables over $\mathbb{F}_2$. By transforming the key recovery problem of LowMC into a problem of solving a system of multivariate Boolean equations, this method can solve all instances with full S-box layers and even extend the number of attacked rounds to 5 rounds. However, Dinur's algorithm requires a large amount of memory that prompts cryptanalysts to develop low-memory attacks on LowMC.

In order to achieve this goal, at ToSC 2022, Liu et al. [18] proposed a new attack on LowMC by using a simplified version of the crossbred algorithm combined with the naive guess strategy, which offered a better time-memory trade-off. Later, Banik et al. [4] and Sun et al. [24] further extended the low-memory attacks on LowMC, the memory required for the attacks on some instances was effectively reduced, although with a slight sacrifice in time complexity.

1.2 Our Contributions

In this paper, we provide new solutions to the LowMC cryptanalysis competition using only a single known plaintext-ciphertext pair.

- We present a new GnD attack framework designed for 2-round LowMC instances with full S-box layers. In this framework, two strategies are used to linearize LowMC S-boxes, aiming to achieve the best trade-off in complexity. Then, all free variables associated with the key can be reduced in two stages, leading to enhanced results when performing the fast exhaustive search algorithm and Dinur's algorithm to 2-round LowMC cryptanalysis.

- By revisiting our attack algorithm in a more fine-grained manner, we present a new MITM attack framework that combines the GnD method. In our optimal attacks for 2-/3-round LowMC instances with full S-box layers, we can reduce the number of variables while achieving a trade-off in guess complexity, deduce an additional collision equation to narrow down the range of candidate keys, and perform a final attack to recover the secret key.

- As shown in Table 1.2, our 3-stage MITM attacks outperform the best previous 2-round attacks proposed in ASIACRYPT 2021 paper [6] in terms of

time complexity and memory complexity, with memory drastically reduced by a factor of $2^{29.7} \sim 2^{70.4}$. Thus, our results set a new record in the LowMC cryptanalysis competition.

While our 3-round attack did not break the record, the concept of algebraic attacks based on the MITM idea still holds potential and serves as our future research work.

1.3 Organization

In Section 2, we introduce a description of LowMC, the linearization techniques for LowMC S-box, and an overview of both the fast exhaustive search algorithm and Dinur's algorithm. In Section 3, we present the GnD attack framework and its application effects. In Section 4, we present the MITM attack framework, including both a 2-stage MITM attack and a 3-stage MITM attack. Finally, we summarize the full paper in Section 5.

2 Preliminaries

2.1 Description of LowMC

LowMC [1] is a family of block ciphers with flexible SPN structures. Unlike conventional block ciphers, the instantiation of LowMC is not fixed, and users can independently select parameters: the block size n, the key size k, the number of S-boxes s in each round and the allowed data complexity D of attacks. The number of rounds r for LowMC is determined by the round calculation formula, to reach the security margins from these parameters.

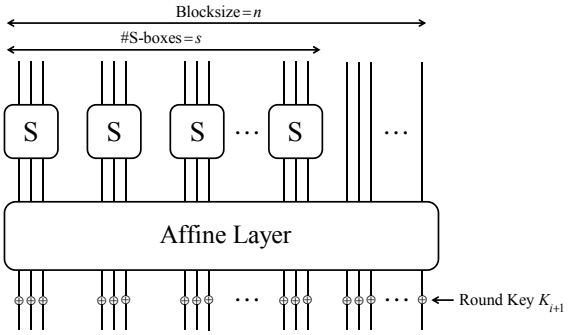

Fig. 1. LowMC Round Function

Specifically, LowMC encryption begins with a key whitening, and the round function given in Fig. 1 at the $(i+1)$-th $(0 \leq i \leq r-1)$ round can be described as follows:

Table 1. A summary of the cryptanalysis results for the 2-/3-round LowMC instances with full S-box layers under single-data complexity. Where the time complexity T is estimated in bit operations and the memory complexity M is estimated in bits. The Exh.Search denotes the $\log_2$ bit operations required for exhaustive search, which is based on a single r-round LowMC encryption requires about $2rn^2$ bit operations [6]. The values of (h, t) are the optimal parameters for minimizing the time complexity of our attacks.

n	k	s	r	(h,t)	$\log_2(T)$	$\log_2(M)$	Exh.Search	References
129	129	43	2	/	97	53	145	[6]
					118	92		[10]
					125.43	77.4		[4]
					128.4*	40.2*		[24]
				(28, 15)	94.4	23.3		Ours
192	192	64	2	/	139	75	209	[6]
					170	126		[10]
					181.91	112.58		[4]
					186.6*	55.9*		[24]
				(46, 18)	136.6	26.6		Ours
255	255	85	2	/	182	97	273	[6]
					222	173		[10]
					243.03	152.67		[4]
					244.5*	71.4*		[24]
				(67, 18)	178.7	26.6		Ours
129	129	43	3	/	140	53	146	[6]
					125	104		[10]
					127.2	16.9		[18]
					129.46	81.5		[4]
				(28, 15)	137.4	23.3		Ours
192	192	64	3	/	203	75	210	[6]
					180	150		[10]
					186.2	18.6		[18]
					187.88	118.41		[4]
				(46, 18)	200.6	26.6		Ours
255	255	85	3	/	267	97	274	[6]
					235	197		[10]
					246.8	19.8		[18]
					247.35	155.55		[4]
				(67, 18)	263.7	26.6		Ours

* The optimal complexity was recalculated using the formula in [24].

1. *SBoxLayer*: A 3-bit S-box $S(x_0, x_1, x_2) = (x_0 \oplus x_1x_2, x_0 \oplus x_1 \oplus x_0x_2, x_0 \oplus x_1 \oplus x_2 \oplus x_0x_1) = (y_0, y_1, y_2)$ 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. *LinearLayer*: The n-bit state is multiplied with a random matrix L_i, which is an invertible $n \times n$ matrix over $\mathbb{F}_2$.
3. *ConstantAddition*: The n-bit state is XORed with the n-bit round constant RC_i, which is randomly generated.
4. *KeyAddition*: The n-bit state is XORed with the n-bit round key K_{i+1}, where K_{i+1} is obtained by multiplying the k-bit master key with an $n \times k$ binary matrix M_{i+1}. The matrix is chosen independently and uniformly at random from all binary $n \times k$ matrices of full-rank.

Note that the combination of *LinearLayer* and *ConstantAddition* represents the Affine Layer displayed in Fig. 1. The whitened key is denoted by K_0 and it is also calculated by multiplying the master key with a randomly generated full-rank $n \times k$ binary matrix M_0. Moreover, a single r-round LowMC encryption requires about $2rn^2$ bit operations as in [6].

2.2 Linearization Techniques for the LowMC S-box

In our subsequent key recovery attacks, it is necessary to linearize a portion of the LowMC S-boxes. Due to the characteristics of the S-box, linearizing it is a straightforward process. The first linearization technique for the LowMC S-box was proposed by Banik et al. [5], which can be summarized as the following Lemma.

Lemma 1. [5] *Consider the LowMC S-box S defined over the input bits x_0, x_1, x_2. If we guess the value of any 3-variable quadratic Boolean function f which is balanced over the input bits of the S-box, then it is possible to re-write the S-box as an affine function of its input bits. And the same is true for the inverse LowMC S-box.*

Specifically, we consider the quadratic expressions on the LowMC S-box as shown below:

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

Take notice of the expressions of y_0, y_1 and y_2 are all 3-variable quadratic balanced Boolean functions over the input bits x_0, x_1 and x_2. Hence, we can linearize the S-box by guessing the value of any one output bit. For example, let $\alpha = x_0 \oplus x_1 x_2$, the output bits can be rewritten as

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

If we guess the value of α, then (y_0, y_1, y_2) can be expressed as linear functions in terms of (x_0, x_1, x_2), and the 3-bit S-box is fully linearized.

In addition, the LowMC S-box (or inverse S-box) can also be linearized through the naive guess strategy, which involves guessing the values of any two input bits. Each output bit can then be expressed as a linear expression of the input bits. This conclusion can be directly derived from the definition of the LowMC S-box and was first employed by Liu et al [18].

2.3 Algorithms for Solving Multivariate Equation Systems over $\mathbb{F}_2$

Given that our upcoming attack strategy will involve the utilization of two algorithms for solving the system of multivariate Boolean equations, it is essential to provide a concise overview of their capabilities.

Fast Exhaustive Search for Polynomial Systems in $\mathbb{F}_2$. At CHES 2010 Bouillaguet et al. [7] presented an exhaustive search algorithm that can evaluate a single Boolean polynomial of degree d with u variables requires just $d \cdot 2^u$ bit operations, and its initialization phase of complexity is about $O(u^{2d})$, which is negligible when $d \ll u$. In particular, this algorithm is highly suitable for evaluating polynomials of degree 2 or less. It allows us to evaluate any u-variable quadratic Boolean function within 2^{u+1} bit operations and any u-variable linear Boolean function within 2^u bit operations.

Furthermore, the extended fast exhaustive search algorithm can find all the common zeroes to a random Boolean polynomial system of degree d with u variables, its time complexity is only $2d \cdot \log_2 u \cdot 2^u$ bit operations, and the required memory to store the m polynomials is about $m \cdot \binom{u}{\leq d}$, where $\binom{u}{\leq d} = \sum_{i=0}^{d} \binom{u}{i}$.

The characteristic of this efficient exhaustive search algorithm is that it iterates in Gray-codes order instead of lexicographic order, effectively leveraging the properties of Gray-codes, i.e. the Hamming difference between two adjacent Gray-codes is equal to 1. On the other hand, the time complexity of the algorithm is independent of the number of equations, which allows this algorithm to be applicable in various scenarios, particularly in the field of cryptanalysis.

Efficient Algorithm for Solving Multivariate Boolean Equations based on Polynomial Method. At EUROCRYPT 2021, Dinur [10] presented a concretely efficient algorithm (called Dinur's algorithm subsequently) to attack all LowMC instances with full S-box layers. In fact, Dinur's algorithm is an algorithm for solving multivariate equation systems over $\mathbb{F}_2$, which is based on the polynomial method from [20]. Moreover, in our attacks on the 2-round LowMC instances with full S-box layers, we further enhance the effectiveness of Dinur's algorithm using a GnD attack framework.

Considering a system of m Boolean equations of degree d with u variables, which is denoted by $E(x)$. We choose a parameter u_1 and split these variables into two disjoint parts $y \in \mathbb{F}_2^{u-u_1}$ and $z \in \mathbb{F}_2^{u_1}$. As a result, $E(x)$ can be rewritten as $E(y, z)$. The fundamental concept of Dinur's algorithm is to randomly select four different choices for the system $\widetilde{E}(y, z)$, each containing $u_1 + 1$ equations

from $E(y,z)$, and assume that the correct solution is isolated in $E(y,z)$ with probability $1-2^{u_1-m}$, which is typically very close to 1[5].

Definition 1. [10] *A solution $\hat{x}=(\hat{y},\hat{z})$ to $E(y,z)$ is called **isolated** (with respect to the variable partition (y,z)), if $(\hat{y},\hat{z}')$ is not a solution to the system E for any $\hat{z}'\neq\hat{z}$.*

Then, we can efficiently enumerate all isolated solutions to the four derived systems by using the method described in [10]. When an identical solution is found in two derived equation systems, this solution is suggested as a candidate solution to $E(y,z)$. Next, all candidate solutions are tested within the original system, so that the cost of testing the solutions is negligible.

Hence, by transforming the key recovery problem of LowMC instances into a problem of solving the multivariate Boolean equation system, it can be effectively addressed using Dinur's algorithm. The total time complexity is estimated as

$$4\cdot\left(2d\cdot\log_2 u\cdot 2^{u_1}\cdot\binom{u-u_1}{\leq d_{\widetilde{F}}-u_1+1}+(u_1+1)\cdot(u-u_1)\cdot 2^{u-u_1}\right) \tag{1}$$

bit operations and the total memory complexity is estimated as

$$8\cdot(u_1+1)\cdot\binom{u-u_1}{\leq d_{\widetilde{F}}-u_1+1} \tag{2}$$

bits. Finally, we recommend referring to [10] for more details.

3 New GnD Attacks on the LowMC Instances with Full S-box Layers

In this section, we introduce a GnD attack framework that maximizes the linearization potential of S-boxes. Additionally, this strategy transforms the key recovery problem of 2-round LowMC into the problem of solving a smaller system of Boolean equations, and it makes the fast exhaustive search algorithm [7] and Dinur's algorithm [10] more effective.

3.1 The GnD Attack Framework for the 2-round LowMC Instances

Note that in the LowMC challenge instances, both the whitened key K_0 and all round keys K_{i+1} are generated by multiplying the master key K with a full-rank $n\times k$ binary matrix ($n=k$). As a result, the key schedule of LowMC is linear, and each bit of the subkey can be expressed as a linear function in terms of the master key K.

As shown in Fig. 2, we initiate the GnD attack on 2-round LowMC with full S-box layers. First, we linearize the last g S-boxes in the 1st round by guessing

[5] In our 2-round attack using Dinur's algorithm, all the instances satisfy $u_1 \ll m$, so that the correct key will be isolated with a probability very close to 1.

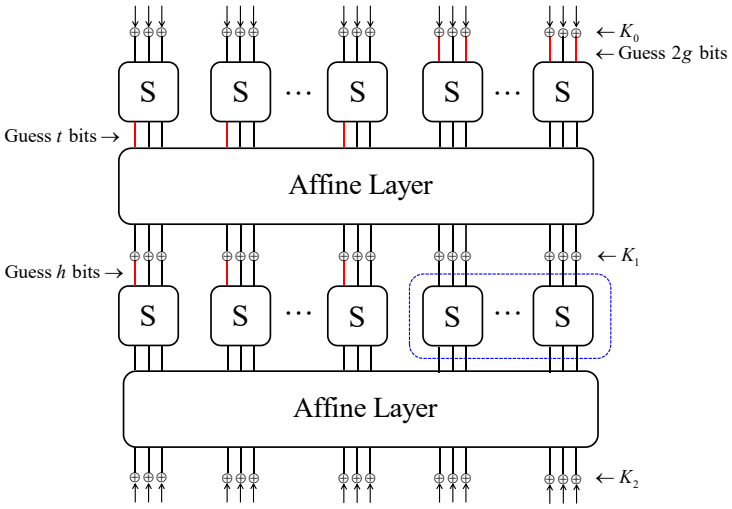

Fig. 2. The GnD Attack Framework for 2-round LowMC

two input bits for each S-box. Combined with the plaintext, it directly provides a system of $2g$ linear equations concerning the master key K. We can perform Gaussian elimination on this system to obtain $n - 2g$ free variables, which are denoted by $v = (v_1, v_2, \cdots, v_{n-2g})$. Then, we linearize the remaining t S-boxes by guessing one output bit for each S-box, so that the 1st round is fully linearized.

Next, we consider the inverse of the 2nd S-box layer and linearize the first h inverse S-boxes by guessing one output bit for each S-box. At this point, starting from the plaintext and ciphertext to reach the intermediate state of the first $3h$ circuits between two rounds, we can construct $3h$ linear equations about v. Hence, we can perform Gaussian elimination on these linear equations to obtain new free variables $\beta = (\beta_1, \beta_2, \cdots, \beta_{n-2g-3h})$. Note that the input and output states of the remaining $\frac{n}{3} - h$ S-boxes in the 2nd round can naturally be represented as linear functions of β. Through the quadratic expressions on the LowMC S-box, we can finally construct $n-3h$ quadratic Boolean equations about β. Since each key bit of K can be linearly represented by $(\beta_1, \beta_2, \cdots, \beta_{n-2g-3h})$, then finding the value of β is equivalent to recovering the secret key. The entire GnD attack framework can be summarized in Algorithm 1.

Complexity Estimation. Now, let us solve the target system of quadratic Boolean equations using the fast exhaustive search algorithm introduced in Section 2.3. Since $n - 2g - 3h \leq n - 3h$, it implies that the system has a unique solution, eliminating the need for additional key testing expenses. Thus the time complexity of the algorithm is estimated as $T_{FES} = \log_2(n-2g-3h) \cdot 2^{n-2g-3h+2}$,

Algorithm 1 The GnD attack framework for the 2-round LowMC

Input: A plaintext-ciphertext pair (P, C).
Output: The secret key K.

Linearize the last g S-boxes in the 1st round by guessing two input bits for each S-box. Denote the guess vector as Vec^1.
for each $Vec^1 \in \{0,1\}^{2g}$ **do**
  A system of $2g$ linear equations about K can be obtained directly, and perform Gaussian elimination on this system to yield $n-2g$ free variables v.
  Linearize the first t S-boxes ($t = s - g$) in the 1st round by guessing one output bit for each S-box. Denote the guess vector as Vec^2.
  for each $Vec^2 \in \{0,1\}^t$ **do**
    Linearize the first h inverse S-boxes in the 2nd round by guessing one output bit for each S-box. Denote the guess vector as Vec^3.
    for each $Vec^3 \in \{0,1\}^h$ **do**
      A system of $3h$ linear equations about v can be obtained through the first $3h$ circuits, and perform Gaussian elimination on this system to yield $n-2g-3h$ free variables β.
      The target system of $n-3h$ quadratic equations about β can be obtained through the remaining $n-3h$ circuits, and then use algebraic techniques to solve it and verify the correctness via (P, C).
      if the solution is correct **then**
        Output it and end the algorithm.
      end if
    end for
  end for
end for

and the total time complexity is estimated as

$$T_b = 2^{2g} \cdot \left(T_{G,1} + 2^{t+h} \cdot (T_{G,2} + T_{FES})\right)$$

bit operations, where $T_{G,1}$, $T_{G,2}$ represent the first and second Gaussian elimination complexity respectively, with $T_{G,1} = (2g)^2 \cdot n$ and $T_{G,2} = (3h)^2 \cdot (n-2g)$. In the entire attack framework, we only need to store the system of equations for two Gaussian eliminations and $n-3h$ quadratic polynomials. Therefore, the total memory complexity is estimated as

$$M_b = 2g \cdot n + 3h \cdot (n-2g) + (n-3h) \cdot \binom{n-2g-3h}{\leq 2}$$

bits. Specifically, we perform the GnD attack on the 2-round LowMC instances, and the results are displayed in Table 2.

3.2 Improving Dinur's Attacks on LowMC using the GnD Strategy

Likewise, we can employ this GnD attack framework to enhance the results of Dinur's algorithm [10] when applied to 2-round LowMC cryptanalysis. Assuming that the symbol definitions and attack process remain the same as before, we

Table 2. Results for the 2-round instances with full S-box layers. Where FES denotes the time complexity of using the fast exhaustive search algorithm directly. The values of (t, g, h) are the optimal parameters for minimizing the time complexity of our attacks.

n	k	s	r	(t, g, h)	T_b	M_b	FES
129	129	43	2	$(43, 0, 38)$	$2^{102.1}$	$2^{14.0}$	2^{134}
192	192	64	2	$(64, 0, 58)$	$2^{145.3}$	$2^{15.2}$	2^{197}
255	255	85	2	$(85, 0, 79)$	$2^{188.2}$	$2^{16.0}$	2^{260}

have constructed a target system of $n - 3h$ quadratic Boolean equations related to $x = (x_1, x_2, \cdots, x_{n-2g-3h})$ in Algorithm 1, denoted by $E(x)$.

Now, let us choose a parameter u_1 and split the $n - 2g - 3h$ variables x into two disjoint parts $y \in \{0,1\}^{n-2g-3h-u_1}$ and $z \in \{0,1\}^{u_1}$. As a result, $E(x)$ can be rewritten as $E(y, z)$, and we randomly select four different choices for $\widetilde{E}(y, z)$. Then, it has reached the usage scenario of Dinur's algorithm [10].

Complexity Estimation. We employ Dinur's algorithm to solve the above quadratic Boolean equations ($d = 2$). Let $u = n - 2g - 3h$, referring to Equation (1) and Equation (2), the time complexity is estimated as

$$\begin{aligned} T' &= 4 \cdot \left(2d \cdot \log_2 u \cdot 2^{u_1} \cdot \binom{u - u_1}{\leq d_{\widetilde{F}} - u_1 + 1} + (u_1 + 1) \cdot (u - u_1) \cdot 2^{u - u_1} \right) \\ &= \log_2 u \cdot 2^{u_1 + 4} \cdot \binom{u - u_1}{\leq d_{\widetilde{F}} - u_1 + 1} + (u_1 + 1) \cdot (u - u_1) \cdot 2^{u - u_1 + 2} \end{aligned}$$

bit operations. Note that $d_{\widetilde{F}}$ can be optimally determined by using Dinur's observation on the 3-round LowMC instances [10]. Specifically, if $\ell = u_1 + 1 \equiv 0 \pmod 3$, then $d_{\widetilde{F}} \leq 4 \cdot \frac{\ell}{3}$; if $\ell \equiv 1 \pmod 3$, then $d_{\widetilde{F}} \leq 4 \cdot \frac{\ell - 1}{3} + 2$; and if $\ell \equiv 2 \pmod 3$, then $d_{\widetilde{F}} \leq 4 \cdot \frac{\ell - 2}{3} + 3$.

Furthermore, the memory complexity of the algorithm is estimated as

$$M' = 8 \cdot (u_1 + 1) \cdot \binom{u - u_1}{\leq d_{\widetilde{F}} - u_1 + 1}$$

bits. Therefore, the total time complexity of the improved attack is estimated as

$$T_d = 2^{2g} \cdot \left(T_{G,1} + 2^{t+h} \cdot (T_{G,2} + T') \right)$$

bit operations and the total memory complexity is estimated as

$$M_d = 2g \cdot n + 3h \cdot (n - 2g) + M'$$

bits. As shown in Table 3, it can be found that our results[6] are better than Dinur's results [10], with a significant advantage of requiring only negligible memory.

[6] We simply restrict $2^{u_1} > 1000u$ to satisfy the condition that $2^{u_1} \gg u$, as discussed in Remark 2.2 of [10].

Table 3. Results for the 2-round instances with full S-box layers. Where the time complexity T_D is estimated in bit operations and memory complexity M_D is estimated in bits, which are derived from [10]. The values of (t, g, h) are the optimal parameters for minimizing the time complexity of our attacks.

n	k	s	r	(t, g, h)	$(u_1, d_{\widetilde{F}})$	T_d	M_d	T_D	M_D
129	129	43	2	$(41, 2, 34)$	$(15, 22)$	$2^{108.2}$	$2^{15.5}$	2^{118}	2^{92}
192	192	64	2	$(62, 2, 55)$	$(15, 22)$	$2^{150.2}$	$2^{16.0}$	2^{170}	2^{126}
255	255	85	2	$(83, 2, 76)$	$(15, 22)$	$2^{192.2}$	$2^{16.5}$	2^{222}	2^{173}

4 New MITM Attacks on the LowMC Instances with Full S-box Layers

Indeed, the GnD attack framework described above is essentially an MITM attack. Through the construction of linear equations in a portion of the state that meets in the middle of the two rounds, the key K can be transformed into free variables using Gaussian elimination. Subsequently, another portion of the state is used to create quadratic equations about free variables over $\mathbb{F}_2$. These equations are then solved to recover the secret key. However, inspired by the MITM attack proposed by Banik et al. [5,6], we revisit our attack algorithm and transform it into a more fine-grained MITM attack in this section.

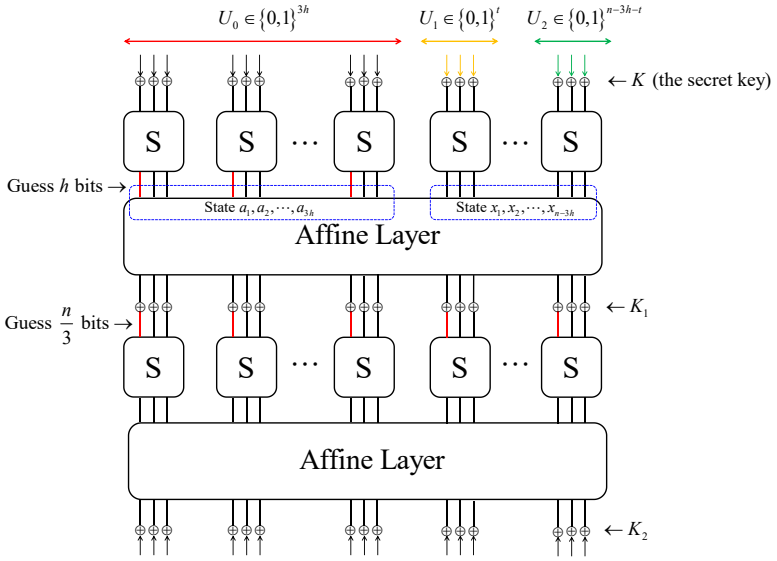

Fig. 3. 2-stage MITM Attack Framework for 2-round LowMC

4.1 2-stage MITM Attack Framework for the 2-round LowMC Instances

Due to the linear key schedule of LowMC, we can regard the whitened key as the secret key $K = [k_1, k_2, \cdots, k_n]$ for cryptanalysis. As illustrated in Fig. 3, we have devised the 2-stage MITM attack framework on the 2-round LowMC instances with full S-box layers. In the following, we will elucidate this attack framework in stages.

1st MITM Stage. First, we split K into three parts: $U_0 = [k_1, k_2, \cdots, k_{3h}]$, $U_1 = [k_{3h+1}, k_{3h+2}, \cdots, k_{3h+t}]$ and $U_2 = [k_{3h+t+1}, k_{3h+t+2}, \cdots, k_n]$, where $t = \lfloor (n-3h)/6 \rfloor \cdot 3$. Then, each bit of the round key K_1, K_2 can be expressed as a linear function in terms of U_0, U_1, U_2.

Next, let us guess one input bit value for each S-box in the 2nd round so that the inverse of the 2nd round is fully linearized. Additionally, we guess one output bit value for each of the first h S-boxes in the 1st round and define $X = (a_1, \cdots, a_{3h}, x_1, \cdots, x_{n-3h})$ to be the output state of the 1st S-box layer. Obviously, starting from the plaintext and ciphertext to reach the intermediate state $(a_1, a_2, \cdots, a_{3h})$, a system of $3h$ linear equations involving U_0, U_1, U_2 can be constructed. The system can be rewritten as

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

where A is an $3h \times 3h$ matrix over $\mathbb{F}_2$, and B is a formal vector whose elements are affine functions in terms of U_1, U_2. Assuming that the matrix A is full-rank, after the Gaussian elimination is applied to Equation (3), each bit of U_0 can be an affine function over U_1, U_2.

2nd MITM Stage. Note that starting from the ciphertext side to arrive at the intermediate state x_b ($b \in [1, n-3h]$), each of them can be expressed as an affine function in terms of U_1, U_2. Since $3|t$, in the process of starting from the plaintext side to reach x_b, the key bits of U_1 and U_2 are never multiplied. In the end, the intermediate state x_b can be expressed as follows:

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

where each f_i, g_j are quadratic functions about U_1, U_2 and each A_i, B_i, C_j, D_j are affine functions about U_1, U_2, and each c_i, d_j are single bit constants. Hence, we can rearrange the terms to obtain the following collision equations:

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

Now, we start to describe the entire 2-stage MITM attack as follows:

1. Calculate the functional forms of f_i, g_j and the values of c_i, d_j for $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.
2. Use Gray-codes to enumerate all possible values of $U_1 \in \{0,1\}^t$, precompute all the values of $f_i(U_1)$ for $\forall i \in [1,t]$ and store them in table L'. Similarly, enumerate all possible values of $U_2 \in \{0,1\}^{n-3h-t}$ based on Gray-codes order, precompute all the values of $g_j(U_2)$ for $\forall j \in [t+1, n-3h]$ and store them in table L''.
3. Linearize the first h S-boxes of the 1st round and the entire S-box layer in the 2nd round, which requires $2^{n/3+h}$ guesses in the worst case.
   - **1st MITM stage:**
   - In the intermediate state $(a_1, a_2, \cdots, a_{3h})$, construct a system of $3h$ linear equations about U_0, U_1, U_2, then perform Gaussian elimination on this system. After that, each bit of U_0 can be an affine function over U_1, U_2.
   - **2nd MITM stage:**
   - According to the current guessing scenario, use Gray-codes to enumerate all possible values of $U_1 \in \{0,1\}^t$, create hash table L_1 indexed by the $(n-3h)$-bit vector $[f_i(U_1) + A_i(U_1) + c_i, \cdots, C_j(U_1)]$, $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.
   - Enumerate all possible values of $U_2 \in \{0,1\}^{n-3h-t}$ based on Gray-codes order, create hash table L_2 indexed by the $(n-3h)$-bit vector $[B_i(U_2), \cdots, g_j(U_2) + D_j(U_2) + d_j]$, $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.
   - Find possible collisions between L_1 and L_2, we expect the number of collisions to be about $2^{t+n-3h-t} \cdot 2^{3h-n} = 1$.
   - When a collision is found, the values of U_1 and U_2 are derived to check for correctness. If the key $K = (U_0, U_1, U_2)$ is correct, the algorithm terminates, otherwise it proceeds to the next guess values and continues iteratively.

Complexity Estimation. Let us analyze the complexity of this attack algorithm, beginning with the time cost of precomputation. Through the fast exhaustive search algorithm based on Gray-codes, we can precompute all the values of $f_i(U_1)$ for $\forall i \in [1,t]$ in $t \cdot 2^{t+1}$ bit operations. Similarly, we precompute all the values of $g_j(U_2)$ for $\forall j \in [t+1, n-3h]$ in $(n-3h-t) \cdot 2^{n-3h-t+1}$ bit operations. Thus, the time complexity of the precomputation is estimated as

$$T_{pre} = t \cdot 2^{t+1} + (n-3h-t) \cdot 2^{n-3h-t+1}$$

bit operations and the required memory is estimated as

$$M_{pre} = t \cdot 2^t + (n-3h-t) \cdot 2^{n-3h-t}$$

bits to store L' and L''.

In fact, the time cost of precomputation is negligible, because the total complexity of the attack algorithm is dominated by the process of Gaussian elimination and the generation of two hash tables. Specifically, in the first MITM stage, the time complexity of Gaussian elimination is about $(3h)^3$. For the second

MITM stage, creating the indexes of table L_1 requires calculating all values of $A_i(U_1)+c_i$ and $C_j(U_1)$, and then adding each precomputed $f_i(U_1)$ for $\forall i \in [1,t]$. The time complexity is estimated as

$$T_{L,1} = (n-3h)\cdot 2^t + t\cdot 2^t = (n-3h+t)\cdot 2^t$$

bit operations. In the same way, creating the indexes of table L_2 requires calculating all values of $D_j(U_2)+d_j$ and $B_i(U_2)$, and then adding each precomputed $g_j(U_2)$ for $\forall j \in [t+1, n-3h]$. The time complexity is estimated as

$$T_{L,2} = (n-3h)\cdot 2^{n-3h-t} + (n-3h-t)\cdot 2^{n-3h-t} = (2n-6h-t)\cdot 2^{n-3h-t}$$

bit operations. Therefore, the total time complexity of this attack is estimated as

$$\begin{aligned} T_H &= T_{pre} + 2^{\frac{n}{3}+h}\cdot\left((3h)^3 + T_{L,1} + T_{L,2}\right) \\ &\approx 2^{\frac{n}{3}+h}\cdot\left((3h)^3 + T_{L,1} + T_{L,2}\right) \end{aligned}$$

bit operations, while the total memory complexity of this attack is estimated as

$$M_H = 3h\cdot n + (n-3h)\cdot(2^t + 2^{n-3h-t}) + M_{pre}$$

bits to store the system of linear equations and L', L'', L_1, L_2.

As shown in Table 4, it displays the parameters h, t that minimize the complexity of our attack. For example, when considering the 2-round LowMC instance of $(n,k,s,r) = (129,129,43,2)$ with optimal parameters, the total time complexity of the attack is approximately $2^{97.9}$. However, the time complexity of precomputation is about $2^{20.9}$, which is negligible compared to $2^{97.9}$.

Table 4. Results for the 2-round instances with full S-box layers

n	k	s	r	h	t	T_H	M_H
129	129	43	2	33	15	$2^{97.9}$	$2^{21.5}$
192	192	64	2	54	15	$2^{140.8}$	$2^{21.5}$
255	255	85	2	73	18	$2^{183.2}$	$2^{24.8}$

4.2 3-stage MITM Attack Framework for the 2-round and 3-round LowMC Instances

To further reduce the time complexity of attacks, we can expand our algorithm into a 3-stage MITM attack framework. In this attack framework, we will deduce an additional collision equation by utilizing properties of the linear code, which refines the key candidates in the final MITM stage.

1st MITM Stage. Similar to Section 4.1, we regard the whitened key as K and split it into three parts: $V_0 = [k_1, k_2, \cdots, k_{3h}]$, $V_1 = [k_{3h+1}, k_{3h+2}, \cdots, k_{3h+t}]$ and $V_2 = [k_{3h+t+1}, k_{3h+t+2}, \cdots, k_n]$, note that $t = \lfloor (n-3h)/9 \rfloor \cdot 3$ here. Consequently, each bit of the round key K_1, K_2 can be expressed as a linear function in terms of V_0, V_1, V_2.

Next, we linearize the inverse of the entire 2nd round by guessing one input bit value for each S-box, and linearize each of the first h S-boxes in the 1st round by guessing one output bit value. Define $Y = (b_1, \cdots, b_{3h}, y_1, \cdots, y_{n-3h})$ to be the output state of the 1st S-box layer, then starting from both the plaintext and ciphertext sides to reach the intermediate state $(b_1, b_2, \cdots, b_{3h})$, we can construct a system of $3h$ linear equations in terms of V_0, V_1, V_2, which can be rewritten as

$$A' \cdot V_0 = A' \cdot [k_1, k_2, \cdots, k_{3h}]^T = B', \tag{4}$$

where A' is an $3h \times 3h$ matrix over $\mathbb{F}_2$, and B' is a formal vector whose elements are affine functions in terms of V_1, V_2. Similarly, we assume that the matrix A' is full-rank, after the Gaussian elimination is applied to Equation (4), each bit of V_0 can be an affine function over V_1, V_2.

2nd MITM Stage and 3rd MITM Stage. The process is the same as in Section 4.1, starting from both the plaintext and ciphertext sides to reach the intermediate state y_c ($c \in [1, n-3h]$), each of them can be expressed as follow:

$$\begin{aligned} y_c &= p_i(V_1) + w_i = E_i(V_1) + F_i(V_2) \ \text{ for } \ \forall c = i \in [1, t], \\ y_c &= q_j(V_2) + s_j = G_j(V_1) + H_j(V_2) \ \text{ for } \ \forall c = j \in [t+1, n-3h], \end{aligned}$$

where each p_i, q_j are quadratic functions about V_1, V_2 and each E_i, F_i, G_j, H_j are affine functions about V_1, V_2, and each w_i, s_j are single bit constants. Therefore, rearranging the terms to obtain the following collision equations:

$$p_i(V_1) + E_i(V_1) + w_i = F_i(V_2) \ \text{ for } \ \forall i \in [1, t], \tag{5}$$

$$G_j(V_1) + s_j = q_j(V_2) + H_j(V_2) \ \text{ for } \ \forall j \in [t+1, n-3h]. \tag{6}$$

In fact, even though the expressions on the LowMC S-box are quadratic, $S(x_0, x_1, x_2)$ is an affine function on $(x_0, x_1, x_2, x_0x_1, x_1x_2, x_0x_2)$. Then, we let $k'_i = k_{3h+i}$ for $\forall i \in [1, t]$ and define

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

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

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

for $\forall i \in [1, t]$, $\forall j \in [t+1, n-3h]$. Next, Equation (5) and Equation (6) can be rewritten as

$$\overline{p}_i(\overline{V}_1) + \overline{E}_i(\overline{V}_1) + w_i = F_i(V_2) \ \text{ for } \ \forall i \in [1, t], \tag{7}$$

$$\overline{G}_j(\overline{V}_1) + s_j = q_j(V_2) + H_j(V_2) \ \text{ for } \ \forall j \in [t+1, n-3h]. \tag{8}$$

Next, let us define a map ϕ:

$$\overline{V}_1 \to \left[\overline{p}_1(\overline{V}_1) + \overline{E}_1(\overline{V}_1), \cdots, \overline{p}_t(\overline{V}_1) + \overline{E}_t(\overline{V}_1), \overline{G}_{t+1}(\overline{V}_1), \cdots, \overline{G}_{n-3h}(\overline{V}_1)\right]^T,$$

which can be seen as a linear code of length $n-3h$ and dimension $2t$. Thus, we can find the $(n-3h) \times 2t$ generator matrix $\mathbf{G}$ and the $(n-3h-2t) \times (n-3h)$ check matrix $\mathbf{H}$ of ϕ, note that they satisfy $\mathbf{H} \cdot \mathbf{G} = 0$.

Define V_c to be the vector $[w_1, w_2, \cdots, w_t, s_{t+1}, \cdots, s_{n-3h}]^T$. It is evident that the left side of Equation (7) and Equation (8) can be written as $\phi(\overline{V}_1) + V_c$. After that, we let $\mathbf{H} \cdot \left[\phi(\overline{V}_1) + V_c\right] = \mathbf{H} \cdot \left[\mathbf{G} \cdot \overline{V}_1 + V_c\right] = \mathbf{H} \cdot V_c$ and try to obtain a new collision equation. Specifically, we split V_2 into two parts $V_2' \in \{0,1\}^t$, $V_2'' \in \{0,1\}^{n-3h-2t}$ and rewrite

$$\begin{aligned} F_i(V_2) &= F_i^{(1)}(V_2') + F_i^{(2)}(V_2''), \\ q_j(V_2) &= q_j^{(1)}(V_2') + q_j^{(2)}(V_2''), \\ H_j(V_2) &= H_j^{(1)}(V_2') + H_j^{(2)}(V_2'') \end{aligned}$$

for $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$. Then define

$$\begin{aligned} N_1 &= \left[F_i^{(1)}(V_2'), \cdots, q_j^{(1)}(V_2') + H_j^{(1)}(V_2')\right]^T, \\ N_2 &= \left[F_i^{(2)}(V_2''), \cdots, q_j^{(2)}(V_2'') + H_j^{(2)}(V_2'')\right]^T \end{aligned}$$

for $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$. Then, the right side of Equation (7) and Equation (8) can be written as $N_1 + N_2$. Hence, let us make

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

which is an additional collision equation. Now, we start to describe the entire 3-stage MITM attack as follows:

1. Calculate the functional forms of p_i, q_j, $\overline{p}_i$, $q_j^{(1)}$, $q_j^{(2)}$ and the values of w_i, s_j for $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.
2. Use Gray-codes to enumerate all possible values of $V_1 \in \{0,1\}^t$, $V_2' \in \{0,1\}^t$, $V_2'' \in \{0,1\}^{n-3h-2t}$, precompute all the values of $p_i(V_1)$, $q_j^{(1)}(V_2')$, $q_j^{(2)}(V_2'')$ for $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$ and store them in table I', I'', I''', respectively.
3. Linearize the first h S-boxes of the 1st round and the entire S-box layer in the 2nd round, which requires $2^{n/3+h}$ guesses in the worst case.
   - **1st MITM stage:**
   - In the intermediate state $(b_1, b_2, \cdots, b_{3h})$, construct a system of $3h$ linear equations about V_0, V_1, V_2, then perform Gaussian elimination on this system. After that, each bit of V_0 can be an affine function over V_1, V_2.
   - **2nd MITM stage:**
   - Based on the current guessing scenario, calculate the functional forms of $\overline{E}_i$, $\overline{G}_j$, $F_i^{(1)}$, $F_i^{(2)}$, $H_j^{(1)}$, $H_j^{(2)}$ for $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.

- Define the map ϕ as before, then we can obtain its generator matrix $\mathbf{G}$ and check matrix $\mathbf{H}$.
- Define V_c, N_1, N_2 as before, then we can calculate the value of $\mathbf{H} \cdot V_c$.
- Use Gray-codes to enumerate all possible values of $V_2' \in \{0,1\}^t$, create hash table I_1, I_a indexed by the $(n-3h-2t)$-bit vector $\mathbf{H} \cdot N_1$, N_1 respectively, $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.
- Use Gray-codes to enumerate all possible values of $V_2'' \in \{0,1\}^{n-3h-2t}$, create hash table I_2, I_b indexed by the $(n-3h-2t)$-bit vector $\mathbf{H} \cdot N_2 + \mathbf{H} \cdot V_c$, N_2 respectively, $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.
- Find possible collisions between I_1 and I_2, we expect the number of collisions to be about $2^{t+n-3h-2t} \cdot 2^{3h+2t-n} = 2^t$. Therefore, we can obtain 2^t candidate keys $V_2 = (V_2', V_2'')$ and store them in table I_0.
- **3rd MITM stage:**
- Use Gray-codes to enumerate all possible values of $V_1 \in \{0,1\}^t$, create hash table I_3 indexed by the $(n-3h)$-bit vector $[p_i(V_1) + E_i(V_1) + w_i, \cdots, G_j(V_1) + s_j]$, $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.
- For all values of $V_2 \in I_0$, create hash table I_4 indexed by the $(n-3h)$-bit vector $[F_i(V_2), \cdots, q_j(V_2) + H_j(V_2)]$, $\forall i \in [1,t]$, $\forall j \in [t+1, n-3h]$.
- Find possible collisions between I_3 and I_4, we expect the number of collisions to be about $2^{2t} \cdot 2^{3h-n} \approx 2^{-(n-3h)/3} < 1$.
- When a collision is found, the values of V_1 and V_2 are derived to check for correctness. If the key $K = (V_0, V_1, V_2)$ is correct, the algorithm terminates, otherwise it proceeds to the next guess values and continues iteratively.

Complexity Estimation. First, we analyze the complexity of the precomputation phase. Based on the fast exhaustive search algorithm, we can precompute all the values of $p_i(V_1)$, $q_j^{(1)}(V_2')$, $q_j^{(2)}(V_2'')$ within

$$T_0 = t \cdot 2^{t+1} + (n-3h-t) \cdot (2^{t+1} + 2^{n-3h-2t+1})$$

bit operations, which is actually negligible, while the memory of precomputation is estimated as

$$M_0 = t \cdot 2^t + (n-3h-t) \cdot (2^t + 2^{n-3h-2t})$$

bits to store I', I'', I'''. Similar to Section 4.1, the total time complexity is dominated by the process of Gaussian elimination and the generation of six hash tables. Specifically, the cost of Gaussian elimination is still about $(3h)^3$. To create I_a, I_b, we need to calculate all values of N_1, N_2 respectively. Note that $q_j^{(1)}$, $q_j^{(2)}$ have been precomputed. Then, the time complexity is estimated as

$$\begin{aligned} T_1 &= (n-3h) \cdot (2^t + 2^{n-3h-2t}) + (n-3h-t) \cdot (2^t + 2^{n-3h-2t}) \\ &= (2n-6h-t) \cdot (2^t + 2^{n-3h-2t}) \end{aligned}$$

bit operations. Similarly, to create I_1, I_2, we need to calculate all values of quadratic expressions $\mathbf{H} \cdot N_1$ and $\mathbf{H} \cdot N_2 + \mathbf{H} \cdot V_c$, and then the time complexity is estimated as

$$T_2 = (n - 3h - 2t) \cdot (2^{t+1} + 2^{n-3h-2t+1})$$

bit operations. Next, to create I_3 that requires calculating all values of $E_i(V_1)+w_i$ and $G_j(V_1) + s_j$, and adding each precomputed $p_i(V_1)$ for $\forall i \in [1, t]$, thus the time complexity is estimated as

$$T_3 = t \cdot 2^t + (n - 3h - t) \cdot 2^t + t \cdot 2^t = (n - 3h + t) \cdot 2^t$$

bit operations. Finally, since all the values of N_1, N_2 are already stored in table I_a, I_b , then to create I_4 can be done by simply adding N_1, N_2 for $\forall(V_2', V_2'') \in I_0$. The time complexity is about $(n - 3h) \cdot 2^t$ bit operations.

Therefore, the total time complexity of this attack is estimated as

$$\begin{aligned} T_{all} &= T_0 + 2^{\frac{n}{3}+h} \cdot \left((3h)^3 + T_1 + T_2 + T_3 + (n - 3h) \cdot 2^t\right) \\ &\approx 2^{\frac{n}{3}+h} \cdot \left((3h)^3 + T_1 + T_2 + T_3 + (n - 3h) \cdot 2^t\right) \end{aligned}$$

bit operations, while the total memory complexity of this attack is estimated as

$$M_{all} = 3h \cdot n + (2n - 6h - 2t) \cdot (2^t + 2^{n-3h-2t}) + (3n - 9h - t) \cdot 2^t + M_0$$

bits to store the system of linear equations and above ten tables.

The Success Probability and Expansibility. Note that the entire attack algorithm is based on the assumption that the coefficient matrix A' is of full-rank. However, in previous work, the relationship between variables was always assumed to be linearly independent during formal Gaussian elimination, so our attack assumption can be established by default. On the other hand, we can also estimate the success probability by using the well-known result of discrete mathematics. Specifically, we denote $\lambda = rank(A')$, then

$$\Pr[\lambda = d] = \prod_{i=0}^{d-1} \left(1 - \frac{2^i}{2^n}\right)$$

for $\forall d \leq n$. When $d = n/n-1/n-2$, the probability $\Pr[\lambda \geq d] \approx 0.29/0.58/0.77$ as n increases. Hence, in the case of a correct guess, we expect the matrix A' to be full-rank with a success probability of 0.29, which is useful in practice. Even if the condition is not met, we can significantly increase the success probability by guessing no more than two key bits, with a slight sacrifice in time complexity.

For expansibility, our optimal attacks can be extended to the 3-round LowMC instances with full S-box layers. This can be achieved simply by linearizing the 3rd S-box layer, where we guess one output bit for each of s S-boxes to transform them into 2-round instances that can be efficiently solved, thereby increasing the time complexity by a factor of 2^s. However, as shown in Table 1.2, our results for 3-round LowMC are still better than those of Banik et al [6], especially in terms of memory complexity.

5 Conclusion

In this paper, we first present a new GnD attack framework designed for 2-round LowMC instances with full S-box layers. When we perform efficient algorithms to solve the transformed system of Boolean equations, the size of the problem has been significantly reduced. Hence, The potential of the fast exhaustive search algorithm and Dinur's algorithm can be maximized for the cryptanalysis of 2-round LowMC. Following that, we present a new MITM attack framework in a more fine-grained manner that combines the GnD method, which can gradually reduce the number of variables and narrow down the range of candidate keys in stages. Most importantly, our 3-stage MITM attacks currently stand as the most effective 2-round attacks and can be implemented with negligible memory, which set a new record in the LowMC cryptanalysis competition.

Acknowledgments. We thank the anonymous reviewers of SAC 2024 for their valuable and constructive comments to improve the quality of this paper. This work is supported by the National Natural Science Foundation of China (Grant No. 12371525).

References

1. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: 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. pp. 430–454. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_17
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. IACR Cryptol. ePrint Arch. p. 687 (2016), http://eprint.iacr.org/2016/687
3. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. IACR Transactions on Symmetric Cryptology pp. 1–45 (2020)
4. Banik, S., Barooti, K., Caforio, A., Vaudenay, S.: Memory-efficient single data-complexity attacks on lowmc using partial sets. Cryptology ePrint Archive (2022), https://eprint.iacr.org/2022/688
5. Banik, S., Barooti, K., Durak, F.B., Vaudenay, S.: Cryptanalysis of lowmc instances using single plaintext/ciphertext pair. IACR Transactions on Symmetric Cryptology **2020**(ARTICLE), 130–146 (2020). https://doi.org/10.46586/tosc.v2020.i4.130-146
6. Banik, S., Barooti, K., Vaudenay, S., Yan, H.: 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. pp. 303–331. Springer (2021). https://doi.org/10.1007/978-3-030-92062-3_11
7. Bouillaguet, C., Chen, H.C., Cheng, C.M., Chou, T., Niederhagen, R., Shamir, A., Yang, B.Y.: Fast exhaustive search for polynomial systems in. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 203–218. Springer (2010). https://doi.org/10.1007/978-3-642-15031-9_14

8. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. Journal of Cryptology **31**(3), 885–916 (2018)
9. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Proceedings of the 2017 acm sigsac conference on computer and communications security. pp. 1825–1842 (2017). https://doi.org/10.1145/3133956.3133997
10. Dinur, I.: 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. pp. 374–403. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_14
11. Dinur, I., Liu, Y., Meier, W., Wang, Q.: Optimized interpolation attacks on lowmc. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 535–560. Springer (2015). https://doi.org/10.1007/978-3-662-48800-3_22
12. Dobraunig, C., Eichlseder, M., Grassi, L., Lallemand, V., Leander, G., List, E., Mendel, F., Rechberger, C.: 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. pp. 662–692. Springer (2018)
13. Dobraunig, C., Eichlseder, M., Mendel, F.: 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. pp. 87–101. Springer (2016). https://doi.org/10.1007/978-3-319-30840-1_6
14. Dobraunig, C., Grassi, L., Guinet, A., Kuijsters, D.: Ciminion: symmetric encryption based on toffoli-gates over large finite fields. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 3–34. Springer (2021)
15. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for {Zero-Knowledge} proof systems. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 519–535 (2021)
16. Kales, D., Zaverucha, G.: Improving the performance of the picnic signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 154–188 (2020). https://doi.org/10.13154/tches.v2020.i4.154-188
17. Liu, F., Isobe, T., Meier, W.: 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. pp. 368–401. Springer (2021). https://doi.org/10.1007/978-3-030-84252-9_13
18. Liu, F., Meier, W., Sarkar, S., Isobe, T.: New low-memory algebraic attacks on lowmc in the picnic setting. IACR Transactions on Symmetric Cryptology pp. 102–122 (2022). https://doi.org/10.46586/tosc.v2022.i3.102-122
19. Liu, F., Sarkar, S., Wang, G., Meier, W., Isobe, T.: Algebraic meet-in-the-middle attack on lowmc. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 225–255. Springer (2022). https://doi.org/10.1007/978-3-031-22963-3_8
20. Lokshtanov, D., Paturi, R., Tamaki, S., Williams, R., Yu, H.: Beating brute force for systems of polynomial equations over finite fields. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 2190–2202. SIAM (2017). https://doi.org/10.1137/1.9781611974782.143

21. Méaux, P., Journault, A., Standaert, F.X., Carlet, C.: 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. pp. 311–343. Springer (2016)
22. Qiao, W., Yan, H., Sun, S., Hu, L., Jing, J.: New cryptanalysis of lowmc with algebraic techniques. Designs, Codes and Cryptography **91**(5), 2057–2075 (2023). https://doi.org/10.1007/s10623-022-01178-1
23. Rechberger, C., Soleimany, H., Tiessen, T.: Cryptanalysis of low-data instances of full lowmcv2. IACR Transactions on Symmetric Cryptology pp. 163–181 (2018). https://doi.org/10.13154/tosc.v2018.i3.163-181
24. Sun, Y., Cui, J., Wang, M.: Improved attacks on lowmc with algebraic techniques. IACR Trans. Symmetric Cryptol. **2023**(4), 143–165 (2023). https://doi.org/10.46586/TOSC.V2023.I4.143-165
25. Zhang, L., Liu, M., Lin, D.: A three-stage mitm attack on lowmc from a single plaintext-ciphertext pair. In: International Conference on Selected Areas in Cryptography. pp. 306–327. Springer (2022)