Cryptanalysis of Full LowMC and LowMC-M with Algebraic Techniques

Fukang Liu[1,3], Takanori Isobe[2,3], Willi Meier[4]

[1] Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China
liufukangs@163.com
[2] National Institute of Information and Communications Technology, Tokyo, Japan
[3] University of Hyogo, Hyogo, Japan
takanori.isobe@ai.u-hyogo.ac.jp
[4] FHNW, Windisch, Switzerland
willimeier48@gmail.com

Abstract. In this paper, we revisit the difference enumeration techniques for LowMC and develop new algebraic techniques to achieve efficient key-recovery attacks with negligible memory complexity. Benefiting from our technique to reduce the memory complexity, we could significantly improve the attacks on LowMC when the block size is much larger than the key size and even break LowMC with such a kind of parameter. On the other hand, with our new key-recovery technique, we could significantly improve the time to retrieve the full key if given only a single pair of input and output messages together with the difference trail that they take, which was stated as an interesting question by Rechberger et al. in ToSC 2018. Combining both the techniques, with only 2 chosen plaintexts, we could break 4 rounds of LowMC adopting a full S-Box layer with block size of 129, 192 and 255 bits, respectively, which are the 3 recommended parameters for Picnic3, an alternative third-round candidate in NIST's Post-Quantum Cryptography competition. We have to emphasize that our attacks do not indicate that Picnic3 is broken as the Picnic use-case is very different and an attacker cannot even freely choose 2 plaintexts to encrypt for a LowMC instantiation. However, such parameters are deemed as secure in the latest LowMC. Moreover, much more rounds of seven instances of the backdoor ciphers LowMC-M as proposed by Peyrin and Wang in CRYPTO 2020 can be broken without finding the backdoor by making full use of the allowed 2^{64} data. The above mentioned attacks are all achieved with negligible memory.

Keywords: LowMC, LowMC-M, linearization, key recovery, negligible memory

1 Introduction

LowMC [5], a family of flexible Substitution-Permutation-Network (SPN) block ciphers aiming at achieving low multiplicative complexity, is a relatively new design in the literature and has been utilized as the underlying block cipher of the post-quantum signature scheme Picnic [3], which is an alternative third-round candidate in NIST's Post-Quantum Cryptography competition [1]. The feature of LowMC is that users can independently choose the parameters to instantiate it, from the number of S-boxes in each round to the linear layer, key schedule function and round constants.

To achieve low multiplicative complexity, the construction adopting a partial S-box layer (only partial state bits will go through the S-boxes and an identical mapping is applied for the remaining state bits) together with a random dense linear layer is most used. As such a construction is relatively new, novel cryptanalysis techniques are required. Soon after its publication, the high-order attack and interpolation attack on LowMC are proposed [13,11], which require a large number of plaintexts. To resist these attacks, LowMC v2 is proposed, i.e. new formulas are used to determine the secure number of rounds. To analyse one of the most useful settings, namely a few S-boxes in each round with low allowable data complexities, the so-called difference enumeration technique [26], which we call difference enumeration attack, is proposed, which makes LowMC v2 move to LowMC v3. The difference enumeration attack is a chosen-plaintext attack. The basic idea is to encrypt a pair (or more) of chosen plaintexts and then recover the difference evolutions between the plaintexts through each component in each round, i.e. to recover the differential characteristic. Finally, the secret key is derived from the recovered differential characteristic. As a result, the number of the required plaintexts can be as low as 4. For simplicity, LowMC represents LowMC v3 in the remaining part of this paper.

Recently, Picnic3 [17] has been proposed and alternative parameters have been chosen for LowMC. Specifically, different from Picnic2 where a partial S-box layer is adopted when instantiating LowMC, a full S-box layer is used when generating the three instances of LowMC in Picnic3. By choosing the number of rounds as 4, the designers found that the cost of signing time and verifying time can be reduced while the signature size is almost kept the same with that of Picnic2 [3]. By increasing the number of rounds to 5 for larger security margin, the cost is still lower than that of Picnic2. Consequently, 4-round LowMC is recommended and 5-round LowMC is treated as an alternative choice.

As can be found in the latest source code [2] to determine the secure number of rounds, the 3 instances of 4-round LowMC used in Picnic3 are deemed as secure. However, there is no thorough study for the constructions adopting a full S-box layer and low allowable data complexities (as low as 2 plaintexts). Therefore, it is meaningful to make an investigation in this direction.

Moreover, a family of tweakable block ciphers called LowMC-M [23] was proposed in CRYPTO 2020, which is built on LowMC and allows to imbed a backdoor in the instantiation. It is natural to ask whether the additional available degrees of freedom of the tweak can give more power to an attacker. Based on the current cryptanalysis [13,11,26], the designers claim that all the parameters of LowMC-M are secure even if the tweak is exploitable by an attacker.

Related Techniques. The algebraic technique seems to be a prominent tool to analyze designs that use low-degree S-boxes. The recent progress made in the cryptanalysis of Keccak is essentially based on algebraic techniques, including the preimage attacks [15,18,19,22], collision attacks [10,24,27,14] and cube attacks [12,16,20].

A pure algebraic attack is to construct a multivariate equation system to describe the target problem and then to solve this equation system efficiently. When the equation system is linear, the well-known Gauss elimination can be directly applied. However, when the equation system is nonlinear, solving such an equation system is NP-hard even if it is quadratic. For the design of block ciphers, there may exist undesirable algebraic properties inside the design which can simplify the equation system and can be further exploitable to accelerate the solving of equations. Such an example can be found in the recent cryptanalysis of the initial version of MARVELLOUS [6] using Gröbner basis attacks [4]. Indeed, there was once a trend to analyze the security of AES against algebraic attacks after Courtois and Pieprzyk demonstrated how to express AES as an overdefined system of multivariate polynomial equations over GF(2) where the equations are quadratic and very sparse [9]. In addition, they also showed how to solve such a quadratic equation system efficiently with the Extended Sparse Linearization (XSL) technique [9]. However, it is still unknown or even questionable whether the so-called XSL technique [9] can work as expected to find the secret key of AES. In the literature, the simple linearization technique and guess-and-determine technique are also common techniques to solve a nonlinear multivariate equation system.

Recently in CRYPTO 2020, a method is proposed to automatically verify a specified differential characteristic [21]. The core technique is to accurately capture the relations between the difference transitions and value transitions. We are inspired from such an idea and will further demonstrate that when the relations between the two transitions are special and when the difference transitions are special, under the difference enumeration attack framework [26], it is possible to utilize algebraic techniques to efficiently recover the differential characteristic for a single pair of (plaintext, ciphertext) and then to efficiently retrieve the full key from the recovered differential characteristic.

Our Contributions. This work is based on the difference enumeration attack framework and we developed several non-trivial techniques to significantly improve the cryptanalysis of LowMC. Our results are detailed as follows:

1. We developed an algebraic technique to reduce the memory complexity to recover the unknown differential characteristics, which allows us to break one parameter of LowMC where the block size is much larger than the key size and to significantly improve the key-recovery attacks on LowMC.
2. By studying the S-box of LowMC, we developed an efficient algebraic technique to retrieve the full key if given only a single pair of (plaintext, ciphertext) along with the corresponding differential characteristic that they take, which was stated as an interesting question by Rechberger et al. in ToSC 2018.

3. We further developed a new difference enumeration attack framework to analyze the constructions adopting a full S-box layer and low allowable data complexities.
4. Combining our techniques, we could break the 3 recommended parameters of 4-round LowMC used in Picnic3, which are treated as secure against the existing cryptanalysis techniques, though it cannot lead to an attack on Picnic3. In addition, much more rounds of 7 instances of LowMC-M can be broken without finding the backdoor, thus violating the security claim of the designers.

All our key-recovery attacks on LowMC only require 2 chosen plaintexts and negligible memory. It is easy to achieve more rounds once more data and huge (impractical) memory are allowed[5]. For our attack on LowMC-M, we will make full use of the allowed data to achieve more rounds. More details are displayed in Table 2, Table 3 and Table 4.

Organization. A brief introduction of LowMC and LowMC-M is given in section 2. We then revisit the difference enumeration attack framework in section 3. In section 4, we make a study on the S-box of LowMC. The techniques to reduce the memory complexity and to reduce the cost to retrieve the secret key from a differential characteristic are detailed in section 5 and section 6, respectively. The application of the two techniques to LowMC with a partial S-box layer and LowMC-M can be referred to section 7. The attack on LowMC with a full S-box layer is explained in section 8. Finally, we conclude the paper in section 9.

2 Preliminaries

2.1 Notation

As there are several instances for both LowMC and LowMC-M, we use the following notations to describe the parameters of LowMC [5] and LowMC-M [23].

1. n represents the block size.
2. k represents the size of the master key.
3. m represents the number of S-boxes in each round.
4. D represents $log_2(D')$, where D' is the number of the allowed data for each instantiation.
5. R represents the total number of rounds.

In addition, the following notations will also be used in this paper.

1. $Pr[\omega]$ represents the probability that the event ω happens.
2. $Pr[\omega|\chi]$ represents the conditional probability, i.e. the probability that ω happens under the condition that χ happens.
3. $x >> y$ represents that x is much greater than y.

[5] We omit this simple work as it is trivial.

2.2 Description of LowMC

LowMC [5] is family of SPN block ciphers proposed by Albrecht et al. in Eurocrypt 2015. Different from conventional block ciphers, the instantiation of LowMC is not fixed and each user can independently choose parameters to instantiate LowMC.

LowMC follows a common encryption procedure as most block ciphers. Specifically, it starts with a key whitening (**WK**) and then iterates a round function by R times. The round function at the $(i+1)$-th $(0 \leq i \leq R-1)$ round can be described below:

1. SBoxLayer (**SB**): A 3-bit S-box $S(x_0, x_1, x_2) = (x_0 \oplus x_1x_2, x_0 \oplus x_1 \oplus x_0x_2, x_0 \oplus x_1 \oplus x_2 \oplus x_0x_1)$ will be applied on the first $3m$ bits of the state in parallel, while an identical mapping is applied on the remaining $n-3m$ bits.
2. LinearLayer (**L**): A regular matrix $L_i \in F_2^{n \times n}$ is randomly generated and multiply the n-bit state with L_i.
3. ConstantAddition (**AC**): An n-bit constant $C_i \in F_2^n$ is randomly generated and is XORed to the n-bit state.
4. KeyAddition (**AK**): A full-rank $n \times k$ binary matrix M_{i+1} is randomly generated. The n-bit round key K_{i+1} is obtained by multiplying the k-bit master key with M_{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 $n \times k$ binary matrix M_0.

It has been studied that there is an equivalent representation of LowMC by placing (**AK**) between (**SB**) and (**L**). In this way, the size of the round key K_i $(i > 0)$ becomes $3m$, which is still linear in the k-bit master key and can be viewed as multiplying the master key with a $3m \times k$ random binary matrix. Notice that K_0 is still an n-bit value. We will use this equivalent representation throughout this paper for simplicity.

Moreover, for convenience, we denote the plaintext by p and the ciphertext by c. The state after **WK** is denoted by A^0. In the $(i+1)$-th round, the input state of (**SB**) is denoted by A_i and the output state of (**SB**) is denoted by A_i^S, as shown below:

$$p \xrightarrow{\mathbf{WK}} A_0 \xrightarrow{\mathbf{SB}} A_0^S \xrightarrow{\mathbf{AK}}\xrightarrow{\mathbf{L}}\xrightarrow{\mathbf{AC}} A_1 \to \cdots \to A_{R-1} \xrightarrow{\mathbf{SB}} A_{R-1}^S \xrightarrow{\mathbf{AK}}\xrightarrow{\mathbf{L}}\xrightarrow{\mathbf{AC}} A_R.$$

In addition, we also introduce the notations to represent the xor difference transitions, as specified below:

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

Specifically, in the $(i+1)$-th round, the difference of the input state of (**SB**) is denoted by Δ_i and the difference of the output state of (**SB**) is denoted by Δ_i^S. The difference of plaintexts is denoted by Δ_p, i.e. $\Delta_p = \Delta_0$.

Definition 1. *A differential characteristic $\Delta_0 \to \Delta_{i+1} \to \cdots \to \Delta_r$ is called a r-round compact differential characteristic when all (Δ_t, Δ_t^S) $(0 \leq t \leq r-1)$ and Δ_r are known.*

2.3 Description of LowMC-M

LowMC-M [23] is a family of tweakable block ciphers built on LowMC, which is introduced by Peyrin and Wang in CRYPTO 2020. The feature of LowMC-M is that backdoors can be inserted in the instantiation. The only difference between LowMC and LowMC-M is that there is an addition operation AddSubTweak (**AT**) after **AK** and **WK**. In other words, the round function in the $(i+1)$-round $(0 \leq i \leq R-1)$ can be described as follows:

1. SBoxLayer (**SB**): Same with LowMC.
2. LinearLayer (**L**): Same with LowMC.
3. ConstantAddition (**AC**): Same with LowMC.
4. KeyAddition (**AK**): Same with LowMC.
5. AddSubTweak (**AT**): Add an n-bit sub-tweak TW_{i+1} to the n-bit state.

For the state after **WK**, it will also be XORed with an n-bit sub-tweak TW_0.

To strengthen the security of the backdoors, TW_i $(0 \leq i \leq R)$ are generated via an extendable-output-function (XOF) function. SHAKE-128 and SHAKE-256 are used as the XOF functions in LowMC-M for 128-bit and 256-bit security respectively. Specifically, the tweak TW is the input of the XOF function and the corresponding $n(R+1)$-bit output will be split into $(R+1)$ sub-tweaks TW_i, i.e. $(TW_0, TW_1, \cdots, TW_R) \leftarrow \text{XOF}(TW)$.

2.4 Standard Difference and d-Difference

The differential attack [8] was first proposed by Biham and Shamir with its application to DES. Since then, the resistance against the differential attack has become the criteria to design a new symmetric primitive. For the common differential attack, the XOR difference between two values will be considered, i.e. $\delta = u \oplus u'$.

However, when it comes to d-difference [28], d XOR differences will be simultaneously taken into account. Specifically, for a tuple of $(d+1)$ values $(u_0, u_2, \cdots, u_d)$, its d-difference is defined as $(\delta_0, \delta_1, \cdots, \delta_{d-1}) = (u_0 \oplus u_1, u_0 \oplus u_2, \cdots, u_0 \oplus u_d)$. If the ordered tuple $(u_0, u_2, \cdots, u_d)$ is transformed by a (linear or nonlinear) function F, supposing the ordered tuple after transformation is $(\nu_0, \nu_2, \cdots, \nu_d)$, i.e. $u_i \xrightarrow{F} \nu_i$ $(0 \leq i \leq d)$, $(u_0 \oplus u_1, u_0 \oplus u_2, \cdots, u_0 \oplus u_d)$ is defined as the input d-difference of F while $(\nu_0 \oplus \nu_1, \nu_0 \oplus \nu_2, \cdots, \nu_0 \oplus \nu_d)$ is defined as the output d-difference of F. Obviously, one could also define the differential characteristic for d-difference, which we call d-differential characteristic in this paper. Similarly, we could also define the compact d-differential characteristic for LowMC, i.e. all the input and output d-differences of each component in each round are fully known.

In this paper, we will use the terminology "d-difference" only when $d > 1$. Otherwise, we simply use "difference" or "standard difference" to represent the XOR difference between two values.

3 The Difference Enumeration Techniques

In this section, we briefly revisit the difference enumeration techniques in [25]. The overall procedure can be divided into three phases, as depicted in Figure 1.

Phase 1: Determine an input difference Δ_0 such that it will not activate any S-boxes in the first t_0 rounds, i.e. $Pr[\Delta_0 \to \Delta_{t_0}] = 1$.

Phase 2: Compute the corresponding Δ_{t_0} from Δ_0 obtained at Phase 1. Enumerate forwards all reachable values for $\Delta_{t_0+t_1}$ and store them in a table denoted by D_f.

Phase 3: Encrypt a pair of plaintexts whose difference equals Δ_0 and compute the difference Δ_r of the corresponding two ciphertexts. Enumerate backwards all reachable differences of $\Delta_{t_0+t_1}$ and check whether it is in D_f.

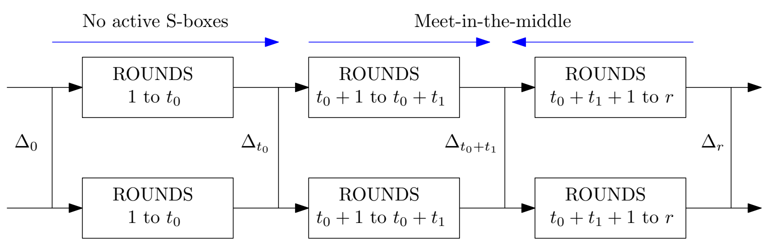

Fig. 1: The framework of the difference enumeration techniques

For convenience, suppose the reachable differences of $\Delta_{t_0+t_1}$ by computing backwards are stored in a table denoted by D_b, though there is no need to store them. To construct a distinguisher, one should expect that $|D_f| \times |D_b| < 2^n$. In this way, one could only expect at most one solution that can connect the difference transitions in both directions. Since there must be a solution, the solution found with the above difference enumeration techniques is the actual solution. After the compact differential characteristic is determined, i.e. the difference transitions in each round are fully recovered, the attacker launches the key-recovery phase.

To increase the number of rounds that can be attacked, the authors exploited the concept of d-difference, which can increase the upper bound for $|D_f| \times |D_b|$, i.e. $|D_f| \times |D_b| < min(2^k, 2^{nd})$. It should be noted that $|D_f| = \lambda_d^{mt_1}$ and $|D_b| = \lambda_d^{mt_2}$, where λ_d denotes the average number of reachable output d-differences over the S-box for a uniformly randomly chosen input d-difference. For the 3-bit S-box used in LowMC, $\lambda_1 \approx 3.62 \approx 2^{1.86}$ and $\lambda_2 \approx 6.58 \approx 2^{2.719}$. Therefore, more number of attacked rounds can be covered by using d-difference ($d > 1$) when $k \geq n$. As for $n > k$, it is thus more effective to use the conventional difference rather than d-difference ($d > 1$).

Another benefit from the d-difference ($d \geq 2$) is that the master key can be efficiently determined when the compact d-differential characteristic is known. For example, consider a non-zero 2-difference transition denoted by $(\delta_0, \delta_1) \rightarrow (\delta_2, \delta_3)$ through the 3-bit S-box of LowMC, where $(\delta_0, \delta_2) \neq (\delta_1, \delta_3)$. Then, the first element of the three inputs to the S-box can be uniquely determined. However, for the standard non-zero difference transition ($d = 1$), one could only derive the values for the two unordered inputs.

Since the candidates for the round key cannot be uniquely determined with only one pair of chosen plaintexts, two pairs of plaintexts are used in [25]. In addition, an interesting problem was proposed in the conclusion section [25], i.e. how to retrieve the full key if only given a single pair of inputs and outputs along with the compact differential characteristic that they take. An efficient method for this problem will bring several benefits. First of all, the data complexity can be further reduced. Most importantly, imagine the case when $|D_f| \times |D_b| > 2^n$, i.e. there are many solutions which can achieve the connection of difference transitions between the backward and forward directions. In this case, reducing the time complexity to recover the full key for each compact differential characteristic becomes quite meaningful as the attacker needs to launch a key-recovery phase after each solution achieving a connection is found.

3.1 The Extended Framework

It is stated in [25] that the above framework can be extended to more rounds if the allowed data are increased. Indeed, as can be found in the latest reference code [2] of LowMC (LowMC v3), the designers have already taken several extensions of the framework in [25] into account to determine a secure number of rounds with the parameters (n, k, D, m). The extended framework is depicted in Figure 2. On the whole, the procedure can be divided into the following five

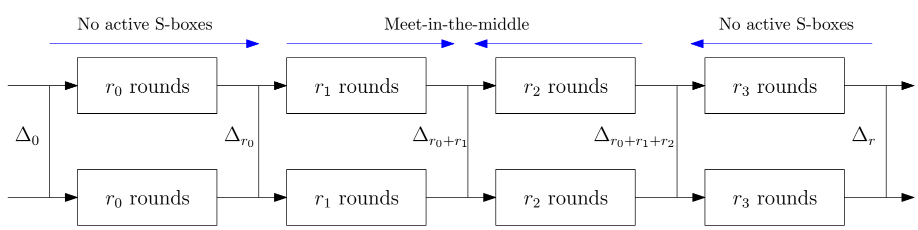

Fig. 2: The extended framework of the difference enumeration techniques

phases:

Phase 1: Iterate all possible d satisfying $1 \leq d \leq (2k/n+1)$ and $log_2(d+1) \leq D$. For each valid d, compute $r_0 + r_1 + r_2 + r_3$ as follows and choose its maximal value as the secure number of rounds.

Phase 2: Determine the maximal value of r_0 such that one could always construct a probability 1 d-differential characteristic in the first r_0 rounds.
Phase 3: Determine the maximal value of r_1 such that $\lambda_d^{mr_1} < min(2^k, 2^{nd})$.
Phase 4: Determine the maximal value of r_2 such that $\lambda_d^{mr_2} < min(2^k, 2^{nd})$.
Phase 5: Determine the maximal value of r_3 such that one could always construct a probability 1 d-differential characteristic in the last r_3 rounds with the allowed number of data.

For convenience, let D_f and D_b still denote the set of reachable d-differences of $\Delta_{r_0+r_1}$ backwards and forwards, respectively. Compared with the framework introduced in [25], there are three different points in the extended framework.

Point 1: $|D_f|$ is much larger when considering the standard difference ($d = 1$) for $n = k$.
Point 2: $|D_b|$ is much larger when considering the standard difference ($d = 1$) for $n = k$.
Point 3: There are extra r_3 rounds where the difference transitions can be determined by using the available plaintexts.

Specifically, when considering the standard difference for $n = k$, there is a constraint that $|D_f| \times |D_b| < 2^n$ in the original framework [25]. However, the constraint becomes $|D_f| < 2^n$ and $|D_b| < 2^n$ in the extended framework, thus significantly increasing the value of $r_1 + r_2$. In addition, after choosing a good starting input d-difference in the plaintexts, the attacker could construct $\lfloor \frac{2^D}{d+1} \rfloor$ different tuples of plaintexts satisfying the chosen input d-difference. For each tuple of plaintexts, the attacker can obtain the corresponding d-difference in the ciphertexts and check whether it will activate the S-boxes in the last r_3 rounds.

In addition, when $n = k$, d-difference ($d > 1$) is adopted in [25] to increase the maximal number of rounds that can be analyzed and the constraint is $\lambda_d^{m(r_1+r_2)} < 2^{2n}$. However, in the extended framework, we have $\lambda_d^{mr_1} < min(2^k, 2^{nd}) = 2^k = 2^n$ and $\lambda_1^{mr_2} < min(2^k, 2^{nd}) = 2^k = 2^n$, i.e. $\lambda_d^{m(r_1+r_2)} < 2^{2n}$. Since λ_d will increase as d increases, to maximize $r_1 + r_2$, d should be 1 and the maximal value is $r_1 = r_2 = \lfloor \frac{n}{m \cdot log_2 \lambda_1} \rfloor$. However, the maximal value is $r_1 = r_2 = \lfloor \frac{n}{m \cdot log_2 \lambda_d} \rfloor$ ($d > 1$) in [25]. Therefore, for $n = k$, we have that $\lfloor \frac{n}{m \cdot log_2 \lambda_1} \rfloor > \lfloor \frac{n}{m \cdot log_2 \lambda_d} \rfloor$ for $d > 1$. More specifically, for $n = k = 128$, we have $\lfloor \frac{128}{m \cdot log_2 \lambda_1} \rfloor \approx \lfloor \frac{86}{m} \rfloor$ and $\lfloor \frac{128}{m \cdot log_2 \lambda_2} \rfloor \approx \lfloor \frac{47}{m} \rfloor$.

It is not difficult to derive from the above explanation that the secure number of rounds are determined by only considering the standard difference for $n = k$. However, there will be about $\lambda_1^{m(r_1+r_2)} \times 2^{-n}$ possible solutions for the compact differential characteristics. When r_1+r_2 takes the maximal value, $\lambda_1^{m(r_1+r_2)} \times 2^{-n}$ is only slightly smaller than 2^n. However, costs for recovering the key which can conform one compact differential characteristic is relatively high. Based on the method mentioned in [25], without the aid of additional pairs of plaintexts, the cost to retrieve the full key is lower bounded by $2^{k/3}$ as each non-zero difference transition through the 3-bit S-box will suggest two solutions and the master key is a k-bit value. The reason why it is a lower bound is that there may exist inactive S-boxes in the differential characteristics and the attacker has to try all the 8 values. Thus, an efficient method to retrieve the full key will allow us to enlarge $\lambda_1^{m(r_1+r_2)} \times 2^{-n}$, thus increasing the number of rounds that can be attacked.

Apart from the high cost of key recovery, it seems to be inevitable that the attacker needs to store D_f in advance for efficient checking. In other words, the memory complexity is rather high as $D_f = \lambda_1^{mr_1}$. We believe that attacks with negligible memory are more effective and meaningful if compared with a pure exhaustive key search.

4 Observations on the S-box

Before introducing our linearization-based techniques for LowMC, it is necessary to describe our observations on the 3-bit S-box used in LowMC. Supposing the 3-bit input and output of the S-box are (x_0, x_1, x_2) and (z_0, z_1, z_2), respectively. Based on the definition of the S-box, the following relations hold:

$$
\begin{aligned}
z_0 &= x_0 \oplus x_1 x_2 \\
z_1 &= x_0 \oplus x_1 \oplus x_0 x_2 \\
z_2 &= x_0 \oplus x_1 \oplus x_2 \oplus x_0 x_1
\end{aligned}
$$

Based on the specification of the S-box, we observed the following property of the S-box.

Observation 1 *For each valid non-zero difference transition $(\Delta x_0, \Delta x_1, \Delta x_2) \to (\Delta z_0, \Delta z_1, \Delta z_2)$, the inputs conforming 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 full list of the all valid non-zero difference transitions along with the corresponding conditions on (x_0, x_1, x_2) as well as the updated expressions for (z_0, z_1, z_2) is given in Table 1. For example, when $(\Delta x_0, \Delta x_1, \Delta x_2) = (0, 0, 1)$ and $(\Delta z_0, \Delta z_1, \Delta z_2) = (0, 0, 1)$, it can be derived that $x_0 = 0$ and $x_1 = 0$. Therefore, the expressions of (z_0, z_1, z_2) become $z_0 = 0$, $z_1 = 0$ and $z_2 = x_2$.

Indeed, the same property also hold for the inverse of the S-box. The inverse of the S-box can be written as follows:

$$
\begin{aligned}
x_0 &= z_0 \oplus z_1 \oplus z_1 z_2 \\
x_1 &= z_1 \oplus z_0 z_2 \\
x_2 &= z_0 \oplus z_1 \oplus z_2 \oplus z_0 z_1
\end{aligned}
$$

Similarly, based on Table 1, it can be derived that the following property also holds for the inverse of the S-box.

Observation 2 *For each valid non-zero difference transition $(\Delta z_0, \Delta z_1, \Delta z_2) \to (\Delta x_0, \Delta x_1, \Delta x_2)$, the inputs conforming such a difference transition will form an* *affine space of dimension 1. In addition, (x_0, x_1, x_2) becomes linear in (z_0, z_1, z_2), i.e. the inverse of the S-box is freely linearized for a valid non-zero difference transition.*

Table 1: The full list for all valid non-zero difference transitions

$(\Delta x_0, \Delta x_1, \Delta x_2)$	$(\Delta z_0, \Delta z_1, \Delta z_2)$	Conditions	z_0	z_1	z_2
(0,0,1)	(0,0,1)	$x_0 = 0, x_1 = 0$	0	0	x_2
	(0,1,1)	$x_0 = 1, x_1 = 0$	1	$1 \oplus x_2$	$1 \oplus x_2$
	(1,0,1)	$x_0 = 0, x_1 = 1$	x_2	1	$1 \oplus x_2$
	(1,1,1)	$x_0 = 1, x_1 = 1$	$1 \oplus x_2$	x_2	$1 \oplus x_2$
(0,1,0)	(0,1,0)	$x_0 = 1, x_2 = 0$	1	$x_1 + 1$	1
	(0,1,1)	$x_0 = 0, x_2 = 0$	0	x_1	x_1
	(1,1,0)	$x_0 = 1, x_2 = 1$	$1 \oplus x_1$	x_1	0
	(1,1,1)	$x_0 = 0, x_2 = 1$	x_1	x_1	$1 \oplus x_1$
(1,0,0)	(1,0,0)	$x_1 = 1, x_2 = 1$	$1 \oplus x_0$	1	0
	(1,0,1)	$x_1 = 0, x_2 = 1$	x_0	0	$1 \oplus x_0$
	(1,1,0)	$x_1 = 1, x_2 = 0$	x_0	$1 \oplus x_0$	1
	(1,1,1)	$x_1 = 0, x_2 = 0$	x_0	x_0	x_0
(0,1,1)	(0,0,1)	$x_1 = x_2 \oplus 1, x_0 = 1$	1	0	x_1
	(0,1,0)	$x_1 = x_2 \oplus 1, x_0 = 0$	0	x_1	1
	(1,0,1)	$x_1 = x_2, x_0 = 1$	$1 \oplus x_1$	1	$1 \oplus x_1$
	(1,1,0)	$x_1 = x_2, x_0 = 0$	x_1	x_1	0
(1,1,0)	(0,1,0)	$x_0 = x_1 \oplus 1, x_2 = 1$	1	x_1	0
	(0,1,1)	$x_0 = x_1, x_2 = 1$	0	x_1	$1 \oplus x_1$
	(1,0,0)	$x_0 = x_1 \oplus 1, x_2 = 0$	x_1	1	1
	(1,0,1)	$x_0 = x_1, x_2 = 0$	x_1	0	x_1
(1,0,1)	(0,0,1)	$x_1 = 1, x_0 = x_2$	0	1	$1 \oplus x_2$
	(1,0,0)	$x_1 = 0, x_0 = x_2$	x_2	0	0
	(0,1,1)	$x_1 = 1, x_0 = x_2 \oplus 1$	1	$1 \oplus x_2$	$1 \oplus x_2$
	(1,1,0)	$x_1 = 0, x_0 = x_2 \oplus 1$	$1 \oplus x_2$	$1 \oplus x_2$	1
(1,1,1)	(0,0,1)	$x_1 = x_2, x_0 = x_2 \oplus 1$	1	1	x_0
	(0,1,0)	$x_1 = x_2, x_0 = x_2$	0	x_0	0
	(1,0,0)	$x_1 = x_2 \oplus 1, x_0 = x_2 \oplus 1$	x_0	0	1
	(1,1,1)	$x_1 = x_2 \oplus 1, x_0 = x_2$	x_0	$1 \oplus x_0$	$1 \oplus x_0$

The above two observations will be used to recover the master key from a compact differential characteristic in an efficient way. Apart from the relations between the difference transitions and value transitions, we could also observe the relations inside the input difference and output difference from Table 1, as summarized below:

Observation 3 *For each non-zero input difference $(\Delta x_0, \Delta x_1, \Delta x_2)$, the corresponding valid output differences form an affine space of dimension 2.*

Observation 4 *For each non-zero output difference $(\Delta z_0, \Delta z_1, \Delta z_2)$, the corresponding valid input differences form an affine space of dimension 2.*

For instance, when the input difference is $(0, 1, 1)$, the corresponding valid output differences satisfy $\Delta z_1 \oplus \Delta z_2 = 1$. When the output difference is $(0, 1, 1)$, the corresponding valid input differences satisfy $\Delta x_1 \oplus \Delta x_2 = 1$.

In addition, for an inactive S-box, i.e. $(\Delta x_0, \Delta x_1, \Delta x_2) = (\Delta z_0, \Delta z_1, \Delta z_2) = (0, 0, 0)$, the S-box can be linearized by guessing two input bits or two output bits, which can be formalized as follows:

Observation 5 *For an inactive S-box, the input becomes linear in the output after guessing two output bits. If guessing two input bits, the output also becomes linear in the input.*

Such an observation can be easily proved by considering the specification of the 3-bit S-box and its inverse.

5 Reducing the Memory Complexity

As mentioned in the previous section, it seems to be inevitable to use a sufficiently large amount of memory to store some reachable differences to achieve efficient checking for the reachable differences computed backwards. It is commonly believed that attacks requiring too much memory indeed cannot compete with a pure exhaustive key search. Therefore, we aim to significantly reduce the memory complexity in both the original and extended frameworks. Specifically, for each reachable difference computed backwards, we try to construct an equation system whose solutions can correspond to the difference transitions in the forward direction.

As illustrated in Figure 3, after we determine the differential characteristic in the first r_0 rounds, Δ_{r_0} is known and there should be at least one active S-box when taking Δ_{r_0} as inputs, otherwise we could extend the deterministic differential characteristic for one more round.

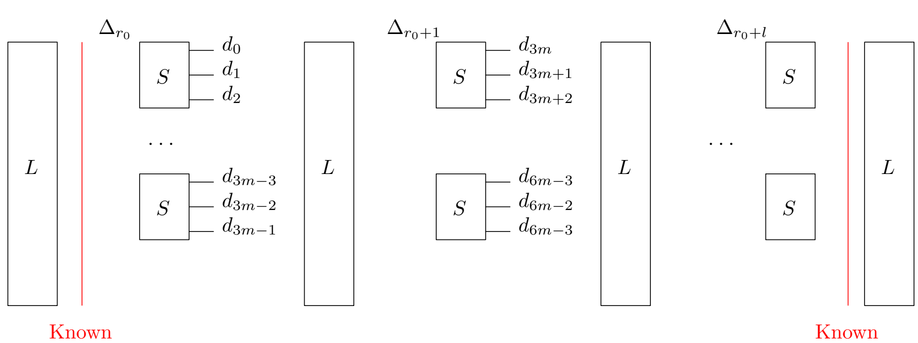

Fig. 3: Constructing the affine subspace of reachable differences

Let us introduce $3m$ variables $(d_0, \cdots, d_{3m-1})$ to denote the output difference of the m S-boxes for the input difference Δ_{r_0}. Then, there will be at least m linear relations inside $(d_0, \cdots, d_{3m-1})$. It can be found that when there is an inactive S-box, the output difference is $(0, 0, 0)$, i.e. three linear relations. When there is an active S-box, the valid output differences form an affine space of dimension 2 according to Observation 3, i.e. 1 linear relation. In other words, we only need to introduce at most $3m - m = 2m$ variables to denote the output differences for Δ_{r_0}. For the next $l - 1$ rounds, since the input difference of the S-box is uncertain due to the diffusion of a random linear layer, we directly introduce $3m(l-1)$ variables $(d_{3m}, \cdots, d_{3ml-1})$ to represent the output differences for each S-box. In this way, Δ_{r_0+l} is obviously linear in the introduced $3m(l-1)+2m = 3ml - m = m(3l-1)$ variables. In other words, Δ_{r_0+l} can be written as linear expressions in terms of the introduced $m(3l-1)$ variables.

Then, for the difference enumeration in the backward direction, after we obtain the output difference of the S-box for Δ_{r_0+l}, we start to construct the equation system to connect the output difference. Specifically, once the output difference of the m S-boxes becomes known, it will leak at least m linear relations for the input difference. Specifically, when the S-box is inactive, the input difference is 0, i.e. three linear relations. When the S-box is active, according to Observation 4, one linear relation inside the input difference can be derived. In other words, we could collect at least $m + (n - 3m) = n - 2m$ linear equations in terms of the introduced $m(3l-1)$ variables. When

$$m(3l-1) \leq n - 2m \to n \geq m(3l+1), \tag{1}$$

we can expect at most one solution of the equation system. Once a solution is found, all output differences of the S-box in the middle l rounds become known and we can easily check whether the difference transitions are valid by computing forwards. If the transitions are valid, a connection between the difference transitions in both directions are constructed. Otherwise, we need to consider another enumerated output difference of the S-box for Δ_{r_0+l} in the backward direction. We have to stress that when enumerating the differences backwards for r_2 rounds, there are indeed $l + 1 + r_2$ rounds in the middle, i.e. $r_1 = l + 1$ if following the extended framework as shown in Figure 2.

However, in some cases where m is large, there is no need to make such a strong constraint as in Equation 1. Even with $n < m(3l+1)$, at the cost of enumerating all the solutions of the constructed linear equation system, more rounds can be covered. In this way, the time complexity to enumerate differences become $2^{1.86mr_2+m(3l+1)-n}$. Thus, the constraint becomes

$$1.86mr_2 + m(3l+1) - n < k. \tag{2}$$

As $l = r_1 - 1$, it can be derived that

$$m(1.86r_2 + 3r_1 - 2) < n + k \tag{3}$$

In addition, the following constraint on r_2 should hold as well.

$$1.86mr_2 < k \tag{4}$$

Therefore, when r_1+r_2 is to be maximized, the above two inequalities should be taken into account. In this way, the time complexity of difference enumeration becomes

$$max(2^{1.86mr_2}, 2^{m(1.86r_2+3r_1-2)-n}). \quad (5)$$

6 Efficient Algebraic Techniques for Key Recovery

In this section, we describe how to retrieve the full key from a compact differential characteristic with an algebraic method. Following the extended framework, we assume that there is no active S-box in the last r_3 rounds. As illustrated in Figure 4, we could introduce $3mr_3$ variables to represent all the input bits of the S-boxes in the last r_3 rounds. Since A_r is the known ciphertext, we could know that A_{r-1} will be linear in $(v_0, \cdots, v_{3m-1})$. Similarly, it can be derived that A_{r-r_3} is linear in all the introduced $3mr_3$ variables $(v_0, \cdots, v_{3mr_3-1})$.

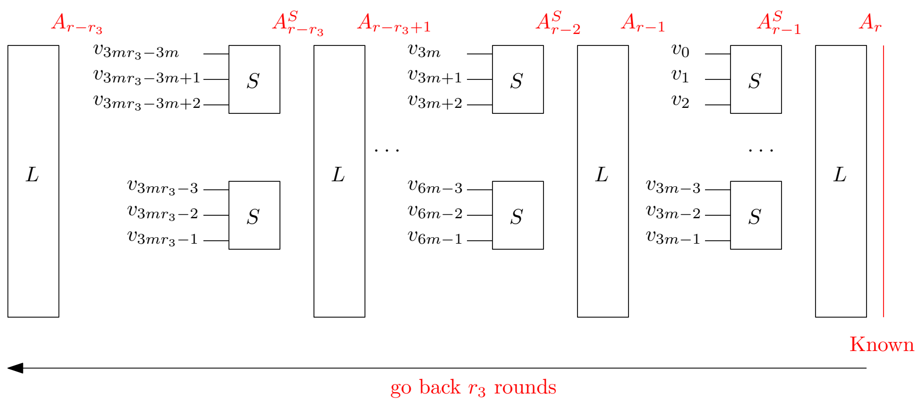

Fig. 4: Linearizing the last r_3 rounds

6.1 Exploiting the Leaked Linear Relations

Since all the S-boxes in the last r_3 rounds are inactive, we have to introduce $3mr_3$ variables to achieve linearization. However, we have not yet obtained any linear equations in terms of these variables. Therefore, we will focus on how to construct a sufficient number of linear equations such that there will be a unique solution of these introduced variables.

It should be noticed that the difference enumeration starts from Δ_{r-r_3} in the backward direction. For a valid r_2-round differential propagation $(\Delta_{r-r_3} \rightarrow \Delta_{r-r_3-1} \rightarrow \cdots \rightarrow \Delta_{r-r_3-r_2})$ enumerated in the backward direction, there should be one valid r_1-round differential propagation $(\Delta_{r_0} \rightarrow \Delta_{r_0+1} \rightarrow \cdots \rightarrow \Delta_{r_0+r_1})$

enumerated in the forward direction such that $\Delta_{r_0+r_1} = \Delta_{r-r_3-r_2}$. Once such a sequence is identified, i.e. $(\Delta_{r_0} \to \cdots \to \Delta_{r-r_3})$ is fully known, we start extracting linear equations from the difference transitions inside the S-boxes in the middle $r_1 + r_2$ rounds.

Specifically, for each active S-box, there will be two linear equations inside the 3-bit output according to Observation 2. In addition, the 3-bit S-box is freely linearized once it is active according to Observation 2, i.e. the 3-bit input can be written as linear expressions in terms of the 3-bit output. Note that A_{r-r_3} is linear in $(v_0, \cdots, v_{3mr_3-1})$.

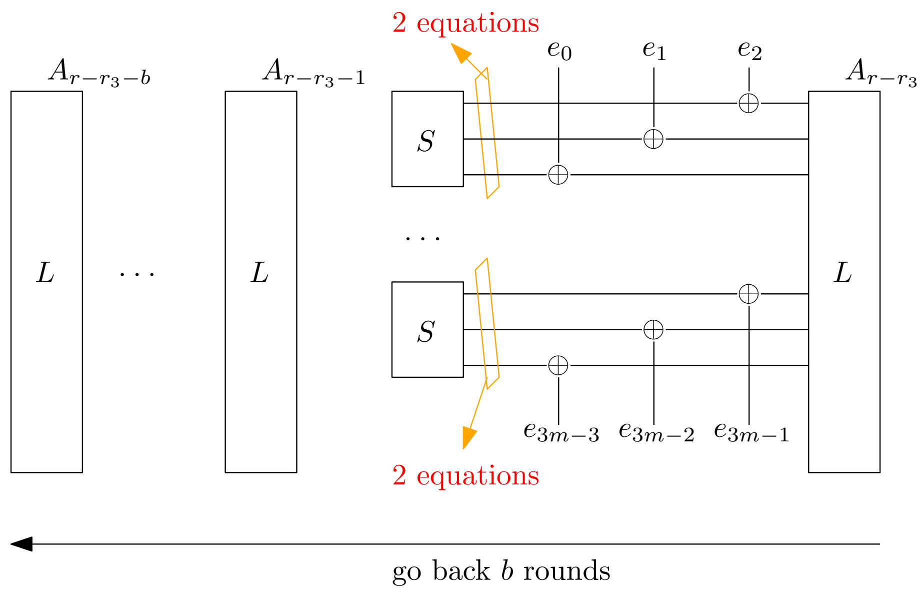

Fig. 5: Extract linear equations from the inactive S-boxes

As depicted in Figure 5, denote the equivalent round key bits used in the $(r - r_3)$-th round by $(e_0, \cdots, e_{3m-1})$. For simplicity, assume that all the S-boxes are active when going back b rounds starting from A_{r-r_3}. The case when there are inactive S-boxes will be discussed later. Under such an assumption, we could derive $2m$ linear equations in terms of $(v_0, \cdots, v_{3mr_3-1}, e_0, \cdots, e_{3m-1})$ based on Observation 2. In addition, since the input becomes linear in the output for each active S-box, A_{r-r_3-1} becomes linear in $(v_0, \cdots, v_{3mr_3-1}, e_0, \cdots, e_{3m-1})$. Similarly, denote the equivalent round key bits used in the $(r-r_3-i)$-th round by $(e_{3mi}, \cdots, e_{3mi+3m-1})$ $(0 \leq i \leq b-1)$. Then, one could derive $2m$ linear equations in terms of $(v_0, \cdots, v_{3mr_3-1}, e_0, \cdots, e_{3mi+3m-1})$ in the $(r - r_3 - i)$-th round and A_{r-r_3-i-1} will be linear in $(v_0, \cdots, v_{3mr_3-1}, e_0, \cdots, e_{3mi+3m-1})$. Repeating such a procedure for b rounds backwards, we could collect in total $2mb$ linear equations in terms of $3mr_3 + 3mb$ variables $(v_0, \cdots, v_{3mr_3-1}, e_0, \cdots, e_{3mb-1})$. Since each equivalent round key bit is linear in the k-bit master key according to the linear key schedule function, we indeed succeed in constructing $2mb$ linear equations in terms of $(v_0, \cdots, v_{3mr_3-1})$ and the k-bit master key. To ensure that there is a unique solution to the equation system, the following constraint should holds:

$$2mb \geq k + 3mr_3. \tag{6}$$

As $2m$ linear equations will be leaked when going back 1 round, there may exist redundant linear equations, i.e. $2mb > k + 3mr_3$. Indeed, only

$$h = \lceil \frac{(k + 3mr_3) - 2m(b-1)}{2} \rceil \tag{7}$$

active S-boxes are needed in the $(r - r_3 - b)$-th round. In this way, we only need in total

$$H = h + m(b-1) \tag{8}$$

S-boxes to ensure that there exists a unique solution of the constructed equation system.

6.2 Linearizing the Inactive S-boxes

After discussing the case when all the S-boxes are active when going back b rounds starting from A_{r-r_3}, let us consider the case when there are q inactive S-boxes among the required H S-boxes in these b rounds ($0 \leq q \leq H$). Specifically, we want to compute the probability as well as the time complexity to recover the full key for such a case.

For an inactive S-box, it can be linearized by guessing two bits of its input or output according to Observation 5. In other words, even for an inactive S-box, we could guess two linear relations for its output. Therefore, the number of equations remain the same as in the case when all the S-boxes are active. The only cost is that we need to iterate 2^{2q} times of guessing. If Equation 6 satisfies, for each time of guessing, one could only expect 1 unique solution for the k-bit master key. Therefore, the expected time to recover the full key from one random compact differential characteristic can be evaluated as follows:

$$\begin{aligned}
T_0 &= \sum_{q=0}^{H} (\frac{7}{8})^{H-q} \times (\frac{1}{8})^{q} \times \binom{H}{q} \times 2^{2q} \\
&= \sum_{q=0}^{H} (\frac{7}{8})^{H-q} \times (\frac{1}{2})^{q} \times \binom{H}{q} \\
&= (\frac{7}{8} + \frac{1}{2})^{H} = 1.375^{H},
\end{aligned}$$

where $H = h + m(b-1)$.

The above formula shows the expected time to recover the full key from one compact differential characteristic. It can not be ensured that the recovered key is the actual one and one has to further verify it by checking the plaintext-ciphertext pair. If there are N valid compact differential characteristics left in the extended framework, the total time complexity to recover the correct master key is therefore

$$T_1 = N \times 1.375^H. \tag{9}$$

Similar to the above method, we could also give a formula to compute the expected time to recover the correct key if following the simple method as discussed in [25]. It should be noted that there is no extra strategy used in the key-recovery phase in [25]. Specifically, when the S-box is active, the attacker needs to try the two possible values. When the S-box is inactive, the attacker needs to try all the 8 possible values. However, since the attacker could always derive 3-bit information of the master key from one S-box in this way, he only needs to go back $b' = \lceil \frac{k-mr_3}{3m} \rceil$ rounds and the needed number of S-boxes is $H' = \lceil \frac{k}{3} \rceil - mr_3$ in these b' rounds. Thus, the expected time T' can be formalized as follows:

$$\begin{aligned} T_2 &= N \times 8^{mr_3} \times \sum_{q=0}^{H'} (\frac{7}{8})^{H'-q} \times (\frac{1}{8})^q \times \binom{H'}{q} \times 8^q \times 2^{H'-q} \\ &= N \times 2^{3mr_3} \times \sum_{q=0}^{mb} (\frac{7}{8} \times 2)^{H'-q} \times (\frac{1}{8} \times 8)^q \times \binom{H'}{q} \\ &= N \times 2^{3mr_3} \times (\frac{7}{4} + 1)^{H'}. \end{aligned}$$

To explain the significant improvement achieved by our linearization techniques to recover the matser key, we make a comparison between T_1 and T_2 as shown below:

$$\frac{T_2}{T_1} = \frac{2^{3mr_3}(\frac{7}{4}+1)^{H'}}{1.375^H}.$$

Since $H = \lceil \frac{k+3mr_3}{2} \rceil$ and $H' = \lceil \frac{k}{3} \rceil - mr_3$, we have

$$\frac{T_2}{T_1} = \frac{2^{3mr_3}(\frac{7}{4}+1)^{H'}}{1.375^H} \approx \frac{2^{3mr_3+1.46(\frac{k}{3}-mr_3)}}{2^{0.46(0.5k+1.5mr_3)}} \approx 2^{0.256k+0.85mr_3}.$$

Obviously, our new key-recovery technique is much faster if compared with the method as in [25].

6.3 Further Improvement

Indeed, one could further reduce the cost to retrieve the full key from a compact differential characteristic. Specifically, we first upper bound b as in Equation 6. Then, when going back $r_3 + b - 1$ rounds from the ciphertext, there will be

$2m(b-1)$ leaked equations and the last r_3+b-1 rounds are fully linearized. Since only $k+3mr_3$ equations are needed and each active S-box will leak 2 equations, we only need to use

$$h = \lceil \frac{(k+3mr_3)-2m(b-1)}{2} \rceil$$

active S-boxes in the $(r-r_3-b)$-th round.

Therefore, in the $(r-r_3-b)$-th round, when there are more than h active S-boxes, there is no need to guess extra equations but we still need to construct the equation system. However, when there are i $(i<h)$ active S-boxes, it is necessary to guess $2h-2i$ extra equations. Therefore, the expected time complexity can be refined as:

$$\begin{aligned}
T_3 = {} & N \times T_4 \times \sum_{i=0}^{h} \binom{m}{i} \times (\frac{7}{8})^i \times (\frac{1}{8})^{m-i} \times 2^{2h-2i} \\
& + N \times T_4 \times \sum_{i=h+1}^{m} \binom{m}{i} \times (\frac{7}{8})^i \times (\frac{1}{8})^{m-i} \\
& \approx N \times T_4 \times 2^{2h} \times \sum_{i=0}^{h} \binom{m}{i} \times (\frac{7}{32})^i \times (\frac{1}{8})^{m-i} \\
& + N \times T_4 \times (1 - \sum_{i=0}^{h} \binom{m}{i} \times (\frac{7}{8})^i \times (\frac{1}{8})^{m-i}) \\
& < N \times T_4 \times (1 + 2^{2h} \times \sum_{i=0}^{h} \binom{m}{i} \times (\frac{7}{32})^i \times (\frac{1}{8})^{m-i})
\end{aligned}$$

where

$$\begin{aligned}
T_4 &= \sum_{q=0}^{m(b-1)} (\frac{7}{8})^{m(b-1)-q} \times (\frac{1}{8})^q \times \binom{m(b-1)}{q} \times 2^{2q} \\
&= \sum_{q=0}^{m(b-1)} (\frac{7}{8})^{m(b-1)-q} \times (\frac{1}{2})^q \times \binom{m(b-1)}{q} \\
&= (\frac{7}{8} + \frac{1}{2})^{m(b-1)} = 1.375^{m(b-1)}.
\end{aligned}$$

There is no simple approximation for T_3 and we therefore provide a loose upper bound which can be easily calculated, as specified below:

$$T_3 < N \times T_4 \times (1 + 2^{2h} \times \sum_{i=0}^{m} \binom{m}{i} \times (\frac{7}{32})^i \times (\frac{1}{8})^{m-i}) = N \times T_4 \times (1 + 2^{2h-1.54m}).$$

Hence, in general, we can use the following formula Equation 10 to calculate the time complexity to retrieve the full key from N compact differential characteristics.

$$T_3 \approx N \times 1.375^{m(b-1)} \times (1 + 2^{2h-1.54m}). \tag{10}$$

It is not surprising that one could go back more than $b+r_3$ rounds to obtain more leaked linear equations if $b \leq r_1 + r_2$. However, the cost of linearization cannot be neglected, i.e. it is necessary to introduce more variables to represent the 3 input bits of an inactive S-box. In other words, although more linear equations can be derived, more variables are involved into the equation system. Note that we need to introduce 3 extra variables to linearize an inactive S-box and only 2 linear equations can be derived from an active S-box. For such a case, it is difficult to give a simple formula describing the expected time complexity to retrieve the full key. Thus, the formula Equation 10 can be viewed as an upper bound.

7 Applications

The above two algebraic techniques can be utilized to further understand the security of LowMC as well as LowMC-M. LowMC is the underlying block cipher used in Picnic, which is an alternative third-round candidate in NIST's post-quantum cryptography competition. For LowMC-M, it is a family of block ciphers based on LowMC which allows to insert a backdoor.

7.1 Applications to LowMC with a Partial S-Box Layer

In this section, we describe how to apply our techniques to instantiations with a partial S-box layer. The result can be summarized in Table 2. All these attacks only require 2 chosen plaintexts and negligible memory. For better understanding, we take $(n, k, m, D, R) = (128, 128, 10, 1, 20)$, $(n, k, m, D, R) = (256, 256, 1, 1, 380)$ and $(n, k, m, D, R) = (1024, 128, 1, 1, 776)$ as examples.

Attack on $(128, 128, 10, 1)$***.*** When $(n, k, m, D) = (128, 128, 10, 1)$, as explained in the extended framework, we have $r_3 = 0$ as there are only two allowed plaintexts for each instantiation and $r_0 = \lfloor \frac{128}{30} \rfloor = 4$. According to Equation 6, $b = 7$. Therefore, the time complexity to retrieve the master key becomes $T_3 \approx 2^{1.86m(r_1+r_2)-128} \times 2^{0.46m(b-1)} = 2^{18.6(r_1+r_2)-81.8} < 2^{128}$ based on Equation 10. The time complexity to enumerate differences is $max(1.86mr_2, m(1.86r_2+3r_1-2)-n) = max(18.6r_2, 18.6r_2+30r_1-148) < 2^{128}$ based on Equation 5 while $18.6r_2 < 128$ (Equation 4) and $18.6r_2 + 30r_1 < 276$ (Equation 3) should hold. Therefore, we have $r_1 + r_2 \leq 11$, $r_2 \leq 6$, $18.6r_2 + 30r_1 \leq 276$. To maximize $r_1 + r_2$ and minimize the total time complexity, we can choose $r_1 = 5$ and $r_2 = 6$. In this way, the time complexity to recover the master key is $2^{122.8}$ while the time complexity to enumerate differences is $max(2^{111.6}, 2^{111.8}) = 2^{111.8}$. Therefore, we could break 15 (out of 20) rounds of LowMC taking the parameter $(n, k, m, D) = (128, 128, 10, 1)$ with time complexity $2^{122.8}$ and only 2 chosen plaintexts.

Attack on $(256, 256, 1, 1)$***.*** When $(n, k, m, D) = (256, 256, 1, 1)$, we have $r_0 = \lfloor \frac{256}{3} \rfloor = 85$, $r_3 = 0$. According to Equation 6, Equation 7, $b = 128$ and $h = 1$. In addition, based on Equation 10, the time complexity to retrieve the master key is $T_3 \approx 2^{1.86(r_1+r_2)-197.12} < 2^{256}$. Based on Equation 5, the time complexity to enumerate differences is $max(1.86r_2, 1.86r_2 + 3r_1 - 258) < 2^{256}$. In addition, we have $1.86r_2 < 256$ and $1.86r_2 + 3r_1 < 514$. Therefore, we have $r_1 + r_2 \leq 243$, $r_2 \leq, 137$ and $1.86r_2 + 3r_1 \leq 514$. Therefore, we choose $r_2 = 137$ and $r_1 = 86$ to maximize $r_1 + r_2$ while minimizing the total time complexity. In this way, we could break $85 + 86 + 137 = 308$ (out of 363) rounds of LowMC taking the parameter $(n, k, m, D) = (256, 256, 1, 1)$ with time complexity $2^{254.82}$.

***Attack on* $(1024, 128, 1, 1)$.** Similarly, when $(n, k, m, D) = (1024, 128, 1, 1)$, we have $r_3 = 0$, $r_0 = \lfloor \frac{1024}{3} \rfloor = 341$, $b = 64$ and $h = 1$. In addition, $1.86r_2 < 128$, $1.86r_2 + 3r_1 < 1154$ and we want to maximize $r_1 + r_2$ while minimizing $max(1.86r_2, 1.86r_2 + 3r_1 - 1026) < 2^{128}$. Therefore, to mount an effective attack, $r_2 = 66$ and $r_1 = 342$ is chosen and the time complexity of difference enumeration is $2^{122.76}$. As $1.86(r_1 + r_2) - 1028 < 0$, we can expect only one solution of the compact differential characteristic, thus resulting in $T_3 \approx 2^{29.44}$. Therefore, we could break $341 + 342 + 66 = 749$ (out of 776) rounds of LowMC taking the parameter $(n, k, m, D) = (1024, 128, 1, 1)$ with time complexity $2^{122.76}$. Indeed, as can be found in latest code to determine a secure number of rounds of LowMC [2], 481 rounds is deemed as secure against the difference enumeration attack, while we could break 749 rounds. As an extreme example, we could break $(n, k, m, D) = (1024, 256, 1, 1)$, as displayed in Table 2.

Remark. It is not surprising to further extend r_1 by using a huge amount of memory when $n = k$. However, such attacks are indeed less effective compared with a pure exhaustive search. Therefore, we omit the simple extension of how to attack more rounds using huge memory. It can be found that we could attack more rounds when $n = k$ even with only 2 chosen plaintexts and negligible memory compared with the results in [25], which most benefits from our efficient key-recovery technique. It also shows that the standard difference can outperform the d-difference ($d > 1$) for $n = k$, which is different from that statement in [25] that d-difference ($d > 1$) outperforms the standard difference for $n = k$.

On the other hand, when $n >> k$, we could significantly improve r_1 as the constraint becomes $3r_1 < n$ when using our efficient technique to reduce the memory complexity, while the constraint is $\lambda_1^{r_1} < min(2^{nd}, 2^k)$ in the extended framework. For example, when attacking $(n, k, m, D) = (1024, 128, 1, 1)$, r_1 cannot reach 342 without our technique to reduce the memory complexity since $2^{1.86r_1} < 2^{128}$ has to be satisfied if simply enumerating the reachable differences.

7.2 Applications to LowMC-M

The only difference between LowMC and LowMC-M is that there is an additional operation after the key addition, i.e. the sub-tweak addition. Since the sub-tweaks are generated with an XOF function, the attacker loses the capability to directly control the difference of sub-tweaks. However, the additional degree of freedom provided by the tweak can still be utilized to further extend r_0.

Table 2: The results for LowMC with a partial S-box layer, where the success probability is recorded in the last column

n	k	m	D	R	r_0	r_1	r_2	r_3	r	Data	Time	Memory	Pro.
128	128	1	1	182	42	43	67	0	152	2	$2^{124.62}$	negligible	1
128	128	10	1	20	4	5	6	0	15	2	$2^{122.8}$	negligible	1
192	192	1	1	273	64	64	101	0	229	2	$2^{187.86}$	negligible	1
192	192	10	1	30	6	7	10	0	23	2	2^{186}	negligible	1
256	256	1	1	363	85	86	137	0	306	2	$2^{254.82}$	negligible	1
256	256	10	1	38	8	9	13	0	30	2	$2^{241.8}$	negligible	1
1024	128	1	1	776	341	342	66	0	749	2	$2^{122.76}$	negligible	1
1024	256	1	1	819	341	342	136	0	819	2	2^{253}	negligible	1

Maximizing r_0 based on [7]. A very recent work [7] shows how to compute the maximal value of r_0 with a birthday search method. In brief, when there is no active S-box in the first r_0 rounds, an attacker can construct a linear equation system of size $3mr_0$ and in terms of Δ_0 as well as the difference of sub-tweaks $(\Delta TW_0, \cdots, \Delta TW_{r_0-1})$. When the sub-tweaks are fixed, the equation system is thus only in terms of Δ_0, i.e. n variables. Therefore, when $3mr_0 > n$, the equation system is consistent with probability 2^{n-3mr_0}. Thus, the attacker needs to find an assignment for $(\Delta TW_0, \cdots, \Delta TW_{r_0-1})$ such that the constructed equation system is consistent.

To achieve this goal, the equation system will be first re-organized by placing $(\Delta TW_0, \cdots, \Delta TW_{r_0-1})$ on the right-hand of the equation system and placing Δ_0 on the left-hand of the equation system. In other words, the equation system becomes

$$A \cdot \Delta_0 = B \cdot (\Delta TW_0, \cdots, \Delta TW_{r_0-1}),$$

where A is a binary matrix of size $3mr_0 \times n$ and B is a binary matrix of size $3mr_0 \times nr_0$. To ensure that there is a solution to Δ_0, one can derive an equation system of size $3mr_0 - n$ and only in terms of $(\Delta TW_0, \cdots, \Delta TW_{r_0-1})$. Specifically, apply a transform $A'_{3mr_0 \times 3mr_0}$ to both A and B such that the first n rows of $A' \cdot A$ is an identity matrix and the remaining $(3mr_0 - n)$ rows of $A' \cdot A$ are all zero. In this way, we only need to focus on the last $(3mr_0 - n)$ rows of $A' \cdot B$, i.e. a linear equation system of size $3mr_0 - n$ and in terms of $(\Delta TW_0, \cdots, \Delta TW_{r_0-1})$ can be derived to ensure that there is always a solution to Δ_0. Thus, with a parallel collision search [29], it is expected to find $(\Delta TW_0, \cdots, \Delta TW_{r_0-1})$ with time complexity $2^{\frac{3mr_0-n}{2}}$ and negligible memory satisfying such an equation system. Therefore, the constraint for r_0 becomes

$$\frac{3mr_0 - n}{2} < k. \tag{11}$$

In this way, one could find the desirable pair of tweaks as well as the plaintext difference Δ_0 with time complexity $2^{\frac{3mr_0-n}{2}}$. This is the method given in [7] to maximize r_0 and we will use it in our attacks.

Since the allowed data complexity is 2^{64} for all instances of LowMC-M, we can also construct a differential characteristic in the last r_3 rounds where no active S-boxes exist with 2^{3mr_3+1} attempts, i.e. $3mr_3 \leq 63$. Similar to the cryptanalysis of LowMC, we could compute (r_0, r_1, r_2, r_3) and the corresponding total time complexity, as summarized in Table 3.

Table 3: The results for LowMC-M, where the success probability is recorded in the last column

n	k	m	D	R	r_0	r_1	r_2	r_3	r	Data	Time	Memory	Pro.
128	128	1	64	208	122	43	64	21	250	2^{64}	2^{120}	negligible	1
128	128	2	64	104	61	22	32	10	125	2^{61}	2^{120}	negligible	1
128	128	3	64	70	40	15	21	7	83	2^{64}	$2^{118.18}$	negligible	1
128	128	10	64	23	12	5	6	2	25	2^{61}	2^{118}	negligible	1
256	256	1	64	384	253	86	136	21	496	2^{64}	$2^{252.96}$	negligible	1
256	256	3	64	129	83	29	45	7	164	2^{64}	$2^{250.1}$	negligible	1
256	256	20	64	21	12	5	6	1	24	2^{61}	2^{232}	negligible	1

Attack on $(256, 256, 1, 64, 384)$***.*** For this example, we explain how to maximize the number of rounds that can be attacked. We first assume the total time complexity is about 2^{252}. Based on Equation 11, we choose the maximal value $r_0 \approx 253$. In this way, the time complexity of the offline phase is $2^{251.5}$. Then, we have $r_3 = \lfloor \frac{63}{3} \rfloor = 21$. Therefore, $b = \lceil \frac{256+3\times 21}{2} \rceil = 160$ and $h = 1$. Therefore, based on Equation 10, we have $T_3 = 2^{1.86(r_1+r_2)-256+0.46\times 160} = 2^{1.86(r_1+r_2)-182.4}$. Then, we try to maximize $r_1 + r_2$. Based on Equation 3, Equation 4 and Equation 5, we can choose $r_1 = 86$ and $r_2 = 136$. In this way, the time complexity to enumerate differences is $max(2^{136\times 1.86}, 2^{136\times 1.86+3\times 86-2-256}) = 2^{252.96}$ and the time complexity to retrieve the key is $T_3 = 2^{230.52}$. Therefore, we could break $253 + 86 + 136 + 21 = 496$ rounds with time complexity $2^{252.96}$.

Remark. Based on our new key-recovery attacks, it seems that LowMC-M should add extra $2 \times \lceil \frac{k}{3m} \rceil$ rounds to have the same security margin as that of LowMC. In addition, compared with the attack in [7] under the difference enumeration attack framework, our attacks require negligible memory and therefore are much more meaningful compared with a pure exhaustive key search. In addition, much more rounds can be attacked due to our developed algebraic techniques.

8 A Refined Attack Framework for the Full S-Box Layer

The above two techniques are quite general and therefore they can be applied to arbitrary instantiations of LowMC. However, when it comes to a full S-Box layer, we need to make extra efforts to improve the extended attack framework developed by the designers of LowMC. Specifically, it is impossible to construct a probability-1 differential characteristic anymore in the first few rounds. On the other hand, the cost of difference enumeration becomes rather high as a full S-box layer is applied.

To overcome the obstacle that there is no probability-1 differential characteristic, we turn to consider how to choose a desirable input difference such that it will activate a small number of S-boxes as possible in the first two rounds. However, since the linear layer is randomly generated, it is difficult to provide an accurate answer. Thus, similar to the method to calculate the time complexity to retrieve the full key, the general case is taken into account and we calculate the expectation of the number of inactive S-boxes in the first two rounds and verify it via experiments.

To reduce the cost of difference enumeration, we still use the linearization technique. However, as will be explained in the following, a different setting needs to be taken into account where the number of variables is larger than the number of equations. In other words, instead of enumerating the reachable differences for each S-boxes in each round, we construct a linear equation system and enumerate the solutions of this equation system, which can significantly reduce the time complexity as some incompatible difference transitions can be avoided in advance.

8.1 Maximizing the Number of Inactive S-boxes

To maximize the number of inactive S-boxes in the first two rounds, we consider the case when there is only one active S-box in the first round, which can obviously reduce the total number of reachable differences after two rounds.

First of all, consider a simple related problem, as stated below:

Suppose there are two vectors $\mu = (\mu_0, \mu_1, \mu_2) \in F_2^3$ *and* $\gamma = (\gamma_0, \gamma_1, \gamma_2) \in F_2^3$. *For a random binary matrix* M *of size* 3×3 *satisfying*

$$\gamma = M \times \mu,$$

calculate the probability that such a matrix M *is used such that* $\gamma = (0, 0, 0)$ *for* $(\mu_0, \mu_1, \mu_2) = (0, 0, 1), (0, 1, 0), \cdots, (1, 1, 1)$, *respectively.*

For $(\mu_0, \mu_1, \mu_2) = (0, 0, 1)$, when $M[i][j] = 0$ $(j = 2)$, $\gamma = (0, 0, 0)$ can always hold. Thus,

$$Pr[(\gamma_0, \gamma_1, \gamma_2) = (0, 0, 0)|(\mu_0, \mu_1, \mu_2) = (0, 0, 1)] = 2^{-3}.$$

Similarly, we have $Pr[(\gamma_0, \gamma_1, \gamma_2) = (0, 0, 0)|(\mu_0, \mu_1, \mu_2) \neq (0, 0, 0)] = 2^{-3}$.

Note that $\Delta_1 = L_0 \times \Delta_0^S$, where Δ_1 and Δ_0^S are two Boolean vectors of size n and L_0 is a $n \times n$ invertible binary matrix. When there is only one active S-box in the first round, we can know that there is only one non-zero triple $(\Delta_0^S[3i], \Delta_0^S[3i + 1], \Delta_0^S[3i + 2])$ $(0 \leq i < \frac{n}{3})$.

Consider a randomly generated L_0 and a fixed value of Δ_0^S with only one non-zero triple $(\Delta_0^S[3i], \Delta_0^S[3i + 1], \Delta_0^S[3i + 2])$. Denote the event by α that $(\Delta_0^S[3i], \Delta_0^S[3i + 1], \Delta_0^S[3i + 2]) \neq (0, 0, 0)$. Denote by IA the number of inactive

S-boxes in the second round. In this way, we could calculate the conditional probability that there are q inactive S-boxes under α happens, as specified below:

$$Pr[IA = q|\alpha] = \binom{\frac{n}{3}}{q} \times 2^{-3q} \times (\frac{7}{8})^{\frac{n}{3}-q},$$

Since that there are 7 assignments for a non-zero triple $(\Delta_0^S[3i], \Delta_0^S[3i+1], \Delta_0^S[3i+2])$ and there are $\frac{n}{3}$ such triples, there are in total $7 \times \frac{n}{3}$ assignments for Δ_0^S satisfying that there is only one active S-box in the first round. Hence, we can expect to find

$$V(n, q) = \frac{n}{3} \times 7Pr[IA = q|\alpha]. \tag{12}$$

required assignments for Δ_0^S which can ensure q inactive S-boxes in the second round. In other words, when $V(n, q) > 1$, it is expected to always find more than 1 assignments for Δ_0^S such that there are q inactive S-boxes in the second round. For instance, $V(129, 11) > 1$. In other words, we could always expect to find an assignment for Δ_0^S such that there is 1 active S-box in the first round and 11 inactive S-boxes in the second round, which has been confirmed via experiments.

8.2 Enumerating Differences Via Solving Equations

For the instantiations adopting a full S-box layer, even enumerating the difference backwards for 1 round from the ciphertext, it costs at most $4^{\frac{n}{3}}$ time when all the $\frac{n}{3}$ S-boxes are active. In the following, we will show how to reduce the time complexity from $4^{\frac{n}{3}}$ to less than $2^{\frac{n}{3}}$ under the worst case, i.e. all the S-boxes are active. Obviously, when there are fewer active S-boxes, the time can be further reduced. Therefore, we also provide a formula to calculate the expected time to enumerate all the differences via solving equations.

As depicted in Figure 6, we aim to enumerate the differences for two consecutive rounds. Specifically, assuming Δ_i and Δ_{i+1}^S are fixed and known, our aim is to enumerate all the solutions for Δ_i^S such that they can reach Δ_{i+1}^S.

First of all, consider the case where all the S-boxes in the $(i + 1)$-th round and $(i+2)$-th round are active, i.e. all the $2 \times \frac{n}{3}$ S-boxes are active in Figure 6. In this case, there are $4^{\frac{n}{3}}$ possible reachable differences for Δ_{i+1} and each reachable difference of Δ_{i+1} can reach Δ_{i+1}^S with probability $2^{-\frac{n}{3}}$ as each output difference can correspond to 4 different input differences through the 3-bit S-box of LowMC. Thus, it is expected to find the valid $2^{\frac{n}{3}}$ solutions of Δ_{i+1} in $4^{\frac{n}{3}}$ time using the simple difference enumeration.

However, similar to our technique to reduce the memory complexity, based on Observation 3, we could introduce $2 \times \frac{n}{3}$ variables to represent the possible values of Δ_i^S. In this way, Δ_{i+1} will be linear in these variables. Furthermore, based on Observation 4, there will be $\frac{n}{3}$ linear constraints on Δ_{i+1}. Therefore, an equation system of size $\frac{n}{3}$ in terms of $2 \times \frac{n}{3}$ variables is constructed and each solution of the equation system will correspond to a valid connection between Δ_i and Δ_{i+1}^S. Thus, we could find the valid $2^{\frac{n}{3}}$ solutions in only $2^{\frac{n}{3}}$ time.

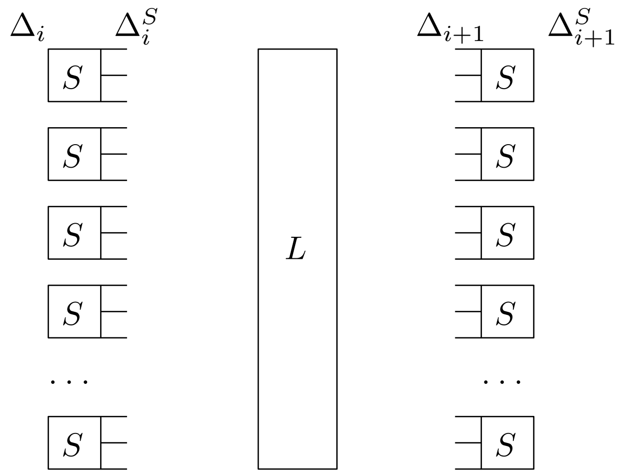

Fig. 6: Enumerating differences via solving equations

After discussing the case where all the S-boxes are active, we consider the general case. Specifically, assume there are w random pairs $(\Delta_i, \Delta_{i+1}^S)$. The expected time complexity to enumerate all the valid difference transitions $\Delta_i \to \Delta_{i+1}^S$ for these w random pairs using our techniques can be formalized as follows.

$$\begin{aligned}
T_5 &= \Big(\sum_{t=0}^{\lfloor 0.5m \rfloor} \binom{m}{t} \times (\frac{1}{8})^t \times (\frac{7}{8})^{m-t} \times \sum_{j=0}^{\lfloor 0.5m \rfloor - t} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j} \times 2^{m-2j-2t}\Big)w \\
&+ \Big(1 - \sum_{t=0}^{\lfloor 0.5m \rfloor} \binom{m}{t} \times (\frac{1}{8})^t \times (\frac{7}{8})^{m-t} \times \sum_{j=0}^{\lfloor 0.5m \rfloor - t} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j}\Big)w \\
&\approx \Big(\sum_{t=0}^{\lfloor 0.5m \rfloor} \binom{m}{t} \times (\frac{1}{8})^t \times (\frac{7}{8})^{m-t} \times \sum_{j=0}^{\lfloor 0.5m \rfloor - t} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j} \times 2^{m-2j-2t}\Big)w + w.
\end{aligned}$$

Specifically, when there are t and j inactive S-boxes in the $(i+2)$-th round and $(i+1)$-th round, respectively, the equation system is of size $3t+(m-t) = m+2t$ and in terms of $2(m-j)$ variables. Thus, for the case $2(m-j)-(m+2t) = m-2j-2t < 0 \to 2j+2t > m$, there is no need to enumerate the solutions and we only need to construct the equation system with time 1. However, for the case $2j+2t \leq m$, we need to construct the equation system as well as enumerate the $2^{m-2j-2t}$ solutions.

As $m > 1$, a loose upper bound for T_5 can be as follows:

$$T_5 < w + w \times 2^m \times (\frac{29}{32})^m \times (\frac{29}{32})^m \approx w \times 2^{0.716m} \tag{13}$$

A fixed random Δ_{i+1}^S**.** We also feel interested in that Δ_{i+1}^S takes a fixed random value while Δ_i takes w random values, which is exactly the case in our attack on 4-round LowMC with a full S-box layer.

When there are $t \leq \lfloor 0.5m \rfloor$ inactive S-boxes in the $(i+2)$-th round, the time complexity T_5 to enumerate all the valid difference transitions can be refined as below:

$$\begin{aligned} T_5 = (&\sum_{j=0}^{\lfloor 0.5m \rfloor - t} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j} \times 2^{m-2j-2t})w \\ &+ (1 - \sum_{j=0}^{\lfloor 0.5m \rfloor - t} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j})w \\ &= (\sum_{j=0}^{\lfloor 0.5m \rfloor - t} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j} \times 2^{m-2j-2t})w + w. \end{aligned}$$

Similarly, a bound for T_5 can be as follows:

$$T_5 < w + w \times 2^{m-2t} \times (\frac{29}{32})^m \approx w + w \times 2^{0.858m-2t}. \tag{14}$$

When there are $t > \lfloor 0.5m \rfloor$ inactive S-boxes in the $(i+2)$-th round, the time complexity T_5 to enumerate all the valid difference transitions can be refined as below:

$$T_5 = (\sum_{j=0}^{m} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j})w = w \tag{15}$$

Combining Equation 14 and Equation 15, we can know that whatever value t takes, the following bound for T_5 holds

$$T_5 < w + w \times 2^{0.858m-2t}. \tag{16}$$

8.3 Applications to 4-Round LowMC with a Full S-box Layer

As can be found in the latest released Picnic3 document, three recommended parameters $(n, k, m, D, R) \in \{(129, 128, 43, 1, 4), (192, 192, 64, 1, 4), (255, 256, 85, 1, 4)\}$ are adopted to achieve the required security. By increasing the number of rounds by 1, i.e. $R = 5$, the designers claim that Picnic3 will provide stronger security. Anyway, 4-round LowMC with a full S-box layer is the recommended instantiation and such three parameters are deemed as secure against the difference enumeration attack [2]. In the following, we explain how to break such 3 parameters with our linearization techniques under the difference enumeration attack framework. Our attacks only apply to LowMC and the security of Picnic3 will never be influenced.

As depicted in Figure 7, our attack procedure can be divided into 4 steps as follows:

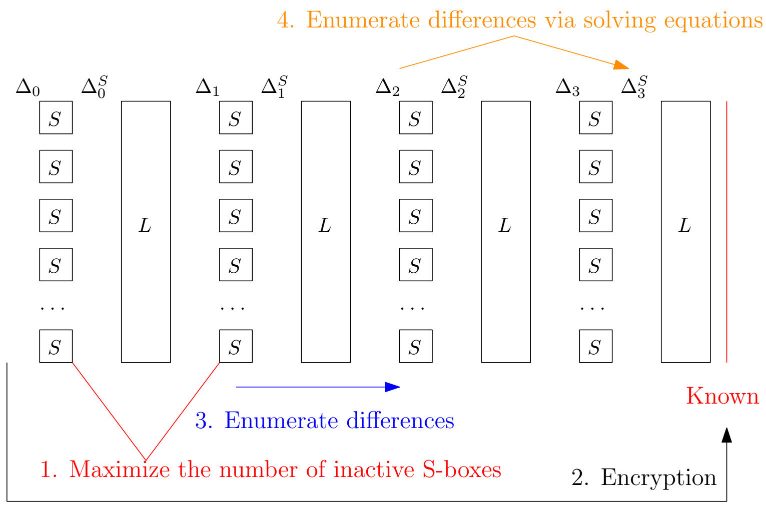

Fig. 7: The attack framework for 4-round LowMC with a full S-box layer

Step 1: According to Equation 12, we find a suitable assignment for Δ_0^S such that the number of inactive S-boxes in the 2nd round can be maximized and there is only one active S-box in the first round. Denote the number of inactive S-boxes in the 2nd round by q.

Step 2: Choose a value for Δ_0 such that it can reach Δ_0^S and encrypt two arbitrary plaintexts whose difference equals Δ_0. Collect the corresponding ciphertexts and compute Δ_3^S.

Step 3: Enumerate 4^{m-q} possible difference transitions from Δ_1 to Δ_2. For each possible difference transition, move to Step 4.

Step 4: For each obtained Δ_2, we enumerate the possible difference transitions from Δ_2 to Δ_3^S via solving a linear equation system, as detailed above. For each solution of the equation system, a compact differential characteristic is obtained and we retrieve the full key from it using our linearization techniques.

Although the formula to calculate the time complexity to retrieve the full key has been given, we should refine it for the attack on 4-round LowMC with a full S-box layer. As can be observed in our attack procedure, once guessing Δ_0^S from its 4 possible values, we already collect two linear equations in terms of the master key and the plaintexts which can ensure that $\Delta_0 \rightarrow \Delta_0^S$ is deterministic based on Observation 1.

On the other hand, due to a sufficiently large number of S-boxes in each round, for the last round, we can introduce extra variables to represent the output bits of the inactive S-boxes. In this way, it is required to extract more than $k-2$ linear equations when a compact differential characteristic is confirmed. Specifically, assuming that there are t inactive S-boxes in the 4th round, the required number of equations become $3t+k-2$. Therefore, we try to extract linear equations from the active S-boxes in the 3rd round and 2nd round, which requires that all the S-boxes in the 3rd are linearized. Therefore, the following formula can be used to estimate the expected time complexity to retrieve the full key from all compatible differential characteristics:

$$\begin{aligned}
T_6 = 4^{m-q} \times (&\sum_{t=0}^{\lfloor \frac{6m-k+2-2q}{5} \rfloor} \binom{m}{t} \times (\frac{1}{8})^t \times (\frac{7}{8})^{m-t} \\
&\times \sum_{j=0}^{m} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j} \times 2^{2j} \times 2^{m-2j-2t} \\
&+ \sum_{t=\lfloor \frac{6m-k+2-2q}{5} \rfloor+1}^{m} \binom{m}{t} \times (\frac{1}{8})^t \times (\frac{7}{8})^{m-t} \\
&\times \sum_{j=0}^{m} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j} \times 2^{2j} \\
&\times 2^{(3t+k-2)-(2(m-t)+2m+2(m-q))} \times 2^{m-2j-2t})
\end{aligned}$$

Specifically, when there are t and j inactive S-boxes in the 4th and 3rd round, respectively, the equation system used to retrieve the master key will be of size $2+2(m-t)+2m+2(m-q)$ and in terms of $3t+k$ variables. More specifically, from the assumed difference transition $\Delta_0 \rightarrow \Delta_0^S$, two linear equations in terms of the master key and the plaintext can be obtained. From the 4th round, as there are $(m-t)$ active S-boxes, $2(m-t)$ equations are obtained. For the 3rd round, we linearize all the j inactive S-boxes by guessing two extra equations based on Observation 5, i.e. guessing two output bits of each inactive S-box. In this way, there will always be $2m$ equations derived from the 3rd round. For the 2nd round, as the 4th round and 3rd round are fully linearized and there are $(m-q)$ active S-boxes, we can obtain $2(m-q)$ linear equations in the 2nd round. Thus, if $3t+k-(2+2(m-t)+2m+2(m-q)) < 0 \rightarrow 5t < 6m-k+2$, the cost is to establish the equation system. When $5t \geq 6m-k+2$, it is necessary to enumerate all the $2^{(3t+k-2)-(2(m-t)+2m+2(m-q))}$ solutions and check them via the plaintext-ciphertext pair.

There is no simple approximation for T_6, we thus provide a loose upper bound:

$$\begin{aligned}
T_6 &< 4^{m-q} \times 2^m \times ((\frac{29}{32})^m + 2^{-6m+k-2+2q} \times (\frac{29}{32})^m) \\
&\approx 2^{3m-2q} \times (2^{-0.142m} + 2^{-6m+k-2+2q} \times 2^{-0.142m}) \\
&= 2^{3m-2q} \times (2^{-0.142m} + 2^{k-2+2q-6.142m}) \\
&= 2^{2.858m-2q} + 2^{k-2-3.142m}
\end{aligned}$$

When the key size is almost equal to the block size, i.e. the parameters ($k \approx n = 3m$) recommended for Picnic3, we have

$$T_6 < 2^{0.9527k-2q}$$

To ensure $2^{2.858m-2q} < 2^{k-2}$, it is required that

$$2.858m - 2q < k - 2 \Rightarrow 2q > 2.858m - k + 2 \quad (17)$$

Δ_3^S is a fixed random value. One may observe that in our attack using only two chosen plaintexts, Δ_3^S is a random fixed value while Δ_2^S behaves randomly. Similar to computing the upper bound for the time complexity to enumerate differences for this case, i.e. Equation 14 and Equation 15, we also try to deal with the time complexity T_6 to retrieve the master key for this case. Similarly, we assume that there are t inactive S-boxes in the 4th round.

When $t \leq \lfloor \frac{6m-k+2-2q}{5} \rfloor$, we have

$$T_6 = 4^{m-q} \times \sum_{j=0}^{m} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j} \times 2^{2j} \times 2^{m-2j-2t} = 2^{3m-2q-2t} \quad (18)$$

When $t > \lfloor \frac{6m-k+2-2q}{5} \rfloor$, we have

$$T_6 = 4^{m-q} \times \sum_{j=0}^{m} \binom{m}{j} \times (\frac{1}{8})^j \times (\frac{7}{8})^{m-j} \times 2^{2j} \quad (19)$$

$$\times 2^{-6m+k-2+2q+5t} \times 2^{m-2j-2t} = 2^{-3m+3t+k-2} \quad (20)$$

Attacks on* $(129, 128, 43, 1, 4)$*. For $(n, k, m, D, R) = (129, 128, 43, 1, 4)$, we have $V(129, 11) > 1$ based on Equation 12, i.e. we can expect to always find an assignment to Δ_0^S such that there will be $q = 11$ inactive S-boxes in the 2nd round. After such a Δ_0^S is chosen, we randomly choose Δ_0 such that $\Delta_0 \to \Delta_0^S$ is valid. There are 4 different values of Δ_0^S for such a Δ_0 and one of Δ_0^S is expected to inactive 11 S-boxes.

The time complexity to retrieve the master key from all valid 4-round compact differential characteristics is related to the value of (t, q). As $t \sim B(m, \frac{1}{8})$, we can expect $t = 5$. In this way, we have $5t = 25 < 6m - k + 2 - 2q = 132 - 2q$ whatever value q $(0 \leq q \leq m)$ takes. In other words, for the expected case $q = 11$, the time complexity to retrieve the master key is $2^{3m-2q-2t} = 2^{97}$ based on Equation 18. By taking the remaining 3 different possible values of Δ_0^S into account, even for the worst case $(q = 0)$, the total time complexity to retrieve the master key for all 4 possible values of Δ_0^S will not exceed $3 \times 2^{3m-2t} = 2^{120.6}$, i.e. less than exhaustive key search.

For the time complexity to enumerate the difference, for the expected case $q = 11$, we have $T_5 < 2^{2m-2q} \times (1 + 2^{0.858m-2t}) = 2^{2.858m-2q-2t} + 2^{2m-2q} = 2^{90.9}$ based on Equation 16. For the worst case $q = 0$, we have $T_5 < 2^{2.858m-2t} = 2^{112.9}$. Therefore, the total time complexity to enumerate the difference will not exceed $3 \times 2^{112.9} \approx 2^{114.5}$. i.e. less than exhaustive key search.

As t increases, T_5 will become smaller. However, when $5t \geq 6m - k + 2 - 2q = 132 - 2q$, we need to use another formula to calculate the time complexity to retrieve the master key, i.e. $T_6 = 2^{-3m+3t+k-2} = 2^{3t-3}$ as shown in Equation 19. Thus, only when $t = m = 43$ which holds with probability 2^{-129},

$T_6 = 2^{128}$. Otherwise our key-recovery attack is always exponentially faster than an exhaustive key search.

As $Pr[t \geq 2] \approx 0.97$ and $Pr[t = 43] \approx 0$, we conclude that with success probability 0.97, the total time complexity to retrieve the master key will be $max(3 \times 2^{3m-2t}, 4 \times 2^{3\times 42-3}) = 2^{126.6}$ and the total time complexity to enumerate differences will not exceed $3 \times 2^{2.858m-2t} < 2^{120.5}$. Thus, we can break the parameter $(n, k, m, D, R) = (129, 128, 43, 1, 4)$ with time complexity less than $2^{126.6}$ and success probability 0.97.

As $Pr[t \geq 4] \approx 0.62$ and $Pr[42 \leq t \leq 43] \approx 0$, we conclude that with success probability 0.62, the total time complexity to retrieve the master key will be $max(3 \times 2^{3m-2t}, 4 \times 2^{3\times 41-3}) = 2^{122.6}$ and the total time complexity to enumerate differences will not exceed $3 \times 2^{2.858m-2t} < 2^{117.5}$. Thus, we can break the parameter $(n, k, m, D, R) = (129, 128, 43, 1, 4)$ with time complexity less than $2^{122.6}$ and success probability 0.62[6].

As $Pr[36 \leq t \leq 43] \approx 0$, if further reducing the success probability to $0.97 \times 0.25 = 0.24$, i.e. $\Delta_0 \to \Delta_0^S$ is assumed to be deterministic and we expect $q = 11$, the time complexity to enumerate the difference will not exceed $2^{2m-2q} + 2^{2.858m-2q-2t} \approx 2^{96.9}$ and the time complexity to retrieve the master key be $max(2^{3m-2q-2t}, 2^{3t-3}) < 2^{103}$.

Attacks on $(192, 192, 64, 1, 4)$***.*** Similar to the above analysis, we first confirm q. As $V(192, 16) > 1$ based on Equation 12, we can expect to always find an assignment to Δ_0^S such that there will be $q = 16$ inactive S-boxes in the 2nd round.

As $Pr[t \geq 3] \approx 0.99$ and $Pr[62 \leq t \leq 64] \approx 0$, based on Equation 18 and Equation 19, the time complexity to retrieve the master key will be $max(3 \times 2^{3m-2t}, 4 \times 2^{-3m+3t+k-2} = 4 \times 2^{3t-2}) < 2^{187.6}$. Based on Equation 16, the time complexity to enumerate the difference is less than $3 \times (2^{2m} + 2^{2m-2t+0.858m}) = 3 \times (2^{2m} + 2^{2.858m-2t}) < 2^{178.5}$. Therefore, we could break $(n, k, m, D, R) = (192, 192, 64, 1, 4)$ with time complexity less than $2^{187.6}$ and success probability 0.99.

As $Pr[t \geq 6] = 0.82$ and $Pr[61 \leq t \leq 64] \approx 0$, the time complexity to retrieve the master key will be $max(3 \times 2^{3m-2t}, 4 \times 2^{3t-2}) = 2^{180}$, while the time complexity to enumerate the differences will not exceed $3 \times (2^{2m} + 2^{2.858m-2t}) < 2^{170.9}$. Therefore, we could break $(n, k, m, D, R) = (192, 192, 64, 1, 4)$ with time complexity less than 2^{180} and success probability 0.82.

To further reduce the success probability, we focus on the expected case $q = 16$ and $3 \leq t \leq 52$. As $Pr[t \geq 3] \approx 0.99$ and $Pr[53 \leq t \leq 64] \approx 0$, we have $Pr[3 \leq t \leq 52] \approx 0.99$. The time complexity to retrieve the master key becomes $max(2^{3m-2t-2q}, 2^{3t-2}) < 2^{154}$. The time complexity to enumerate the difference is less than $2^{2m-2q} + 2^{2.858m-2t-2q} < 2^{144.9}$. Therefore, we could

[6] Obviously, we can achieve lower time complexity by further reducing the success probability. We omit this simple work as our aim is to violate the 128-bit security claim.

break $(n,k,m,D,R) = (192,192,64,1,4)$ with time complexity less than 2^{154} and success probability $0.99 \times 0.25 = 0.247$.

Attacks on $(255,256,85,1,4)$**.** For $(n,k,m,D,R) = (255,256,85,1,4)$, we have $V(256,20) > 1$ based on Equation 12, i.e. we can expect to always find an assignment to Δ_0^S such that there will be $q = 20$ inactive S-boxes in the 2nd round.

As $Pr[t \geq 5] \approx 0.986$ and $Pr[79 \leq t \leq 85] \approx 0$, based on Equation 18 and Equation 19, the time complexity to retrieve the master key will be $max(3 \times 2^{3m-2t}, 4 \times 2^{-3m+3t+k-2} = 4 \times 2^{3t-1}) < 2^{246.6}$. Based on Equation 16, the time complexity to enumerate the difference is less than $3 \times (2^{2m} + 2^{2m-2t+0.858m}) = 3 \times (2^{2m} + 2^{2.858m-2t}) < 2^{234.53}$. Therefore, we could break $(n,k,m,D,R) = (255,256,85,1,4)$ with time complexity less than $2^{246.6}$ and success probability 0.986.

As $Pr[t \geq 8] = 0.848$ and $Pr[79 \leq t \leq 85] \approx 0$, the time complexity to retrieve the master key will be $max(3 \times 2^{3m-2t}, 4 \times 2^{3t-1}) < 2^{240.6}$, while the time complexity to enumerate the differences will not exceed $3 \times (2^{2m} + 2^{2.858m-2t}) < 2^{228.53}$. Therefore, we could break $(n,k,m,D,R) = (255,256,85,1,4)$ with time complexity less than $2^{240.6}$ and success probability 0.848.

To further reduce the success probability, we focus on the expected case $q = 20$ and $5 \leq t \leq 85$. As $Pr[t \geq 5] \approx 0.986$ and $Pr[69 \leq t \leq 85] \approx 0$, we have $Pr[5 \leq t \leq 68] \approx 0.986$. The time complexity to retrieve the master key becomes $max(2^{3m-2t-2q}, 2^{3t-1}) < 2^{205}$. The time complexity to enumerate the difference is less than $2^{2m-2q} + 2^{2.858m-2t-2q} < 2^{192.93}$. Therefore, we could break $(n,k,m,D,R) = (255,256,85,1,4)$ with time complexity less than 2^{205} and success probability $0.986 \times 0.25 = 0.2465$.

Remark. In the above, attacks with success probability of about 0.25 but with a rather low time complexity are introduced when we assume that the transition $\Delta_0 \to \Delta_0^S$ is deterministic. Such attacks can be applied to a multi-target setting. Specifically, when there are 5 users with 5 different master keys, the attacker can recover the master key for one of the 5 users with time complexity significantly less than an exhaustive search. All the above attacks only require 2 chosen plaintexts and negligible memory, which are summarized in Table 4.

9 Conclusion

Benefiting from the low-degree S-box and the linear key schedule function of LowMC, we developed an efficient algebraic technique to solve a general problem of how to retrieve the key if given a single pair of (plaintext, ciphertext) along with its compact differential characteristic. Such a technique is quite meaningful as much more differential characteristic candidates are allowed to exist under the difference enumeration attack framework. As a result, we could significantly extend the number of rounds that can be attacked even with only two allowed plaintexts.

Table 4: The results for 4-round LowMC with a full S-box layer, where the success probability is recorded in the last column

n	k	m	D	R	Data	Time	Memory	Pro.
129	128	43	1	4	2	$2^{126.6}$	negligible	0.97
129	128	43	1	4	2	$2^{122.6}$	negligible	0.62
129	128	43	1	4	2	2^{103}	negligible	0.24
192	192	64	1	4	2	$2^{187.6}$	negligible	0.99
192	192	64	1	4	2	2^{180}	negligible	0.82
192	192	64	1	4	2	2^{154}	negligible	0.247
255	256	85	1	4	2	$2^{246.6}$	negligible	0.986
255	256	85	1	4	2	$2^{236.6}$	negligible	0.848
255	256	85	1	4	2	2^{205}	negligible	0.2465

On the other hand, by exploiting the fact that only a few S-boxes are applied to the state, we could simulate the difference transitions over rounds by constructing a linear equation system, thus requiring negligible memory complexity. However, such a technique cannot be simply used for the construction adopting a full S-box layer. Therefore, we turn to use 2 special properties of the S-box, i.e. both the output differences of a fixed input difference and the input differences of a fixed output difference form an affine space. In this way, even for constructions with a full S-box layer, it is still feasible to constrain all the valid difference transitions over rounds with a linear equation system, thus significantly reducing the time to enumerate all valid difference transitions compared with a simple difference enumeration where invalid difference transitions cannot be filtered in advance.

Combining all our techniques, we violate the security claim for some instances of LowMC. Especially, the 3 recommended parameters of LowMC used in Picnic3 are shown to be insecure against our attacks, though it cannot threaten the security claim for Picnic3. As the backdoor cipher LowMC-M is built on LowMC, making progress in the cryptanalysis of LowMC directly threatens the security claim for 7 instances of LowMC-M even without finding the backdoor.

References

1. `https://csrc.nist.gov/projects/post-quantum-cryptography`.
2. Reference code, 2017. `https://github.com/LowMC/lowmc/blob/master/determine_rounds.py`.
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 S. D. Galbraith and S. Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 371–397. Springer, 2019.

5. M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In E. Oswald and M. Fischlin, editors, *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*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.
6. A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Report 2019/426, 2019. `https://eprint.iacr.org/2019/426`.
7. T. Beyne and C. Li. Cryptanalysis of the malicious framework. Cryptology ePrint Archive, Report 2020/1032, 2020. `https://eprint.iacr.org/2020/1032`.
8. E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, pages 2–21, 1990.
9. N. T. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Y. Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.
10. I. Dinur, O. Dunkelman, and A. Shamir. New attacks on Keccak-224 and Keccak-256. In *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, pages 442–461, 2012.
11. I. Dinur, Y. Liu, W. Meier, and Q. Wang. Optimized interpolation attacks on lowmc. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 535–560. Springer, 2015.
12. I. Dinur, P. Morawiecki, J. Pieprzyk, M. Srebrny, and M. Straus. Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. 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*, pages 733–761, 2015.
13. C. Dobraunig, M. Eichlseder, and F. Mendel. Higher-order cryptanalysis of lowmc. In S. Kwon and A. Yun, editors, *Information Security and Cryptology - ICISC 2015 - 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, volume 9558 of *Lecture Notes in Computer Science*, pages 87–101. Springer, 2015.
14. J. Guo, G. Liao, G. Liu, M. Liu, K. Qiao, and L. Song. Practical collision attacks against round-reduced SHA-3. *IACR Cryptology ePrint Archive*, 2019:147, 2019.
15. J. Guo, M. Liu, and L. Song. Linear structures: Applications to cryptanalysis of round-reduced keccak. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 249–274, 2016.
16. S. Huang, X. Wang, G. Xu, M. Wang, and J. Zhao. Conditional cube attack on reduced-round keccak sponge function. In *Advances in Cryptology - EUROCRYPT*

2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II, pages 259–288, 2017.
17. D. Kales and G. Zaverucha. Improving the performance of the picnic signature scheme. Cryptology ePrint Archive, Report 2020/427, 2020. `https://eprint.iacr.org/2020/427`.
18. T. Li and Y. Sun. Preimage attacks on round-reduced keccak-224/256 via an allocating approach. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, pages 556–584, 2019.
19. T. Li, Y. Sun, M. Liao, and D. Wang. Preimage attacks on the round-reduced keccak with cross-linear structures. *IACR Trans. Symmetric Cryptol.*, 2017(4):39–57, 2017.
20. Z. Li, X. Dong, W. Bi, K. Jia, X. Wang, and W. Meier. New conditional cube attack on keccak keyed modes. *IACR Trans. Symmetric Cryptol.*, 2019(2):94–124, 2019.
21. F. Liu, T. Isobe, and W. Meier. Automatic verification of differential characteristics: Application to reduced gimli (full version). Cryptology ePrint Archive, Report 2020/591, 2020. `https://eprint.iacr.org/2020/591`.
22. F. Liu, T. Isobe, W. Meier, and Z. Yang. Algebraic attacks on round-reduced keccak/xoodoo. Cryptology ePrint Archive, Report 2020/346, 2020. `https://eprint.iacr.org/2020/346`.
23. T. Peyrin and H. Wang. The malicious framework: Embedding backdoors into tweakable block ciphers. Cryptology ePrint Archive, Report 2020/986, 2020. `https://eprint.iacr.org/2020/986`.
24. K. Qiao, L. Song, M. Liu, and J. Guo. New collision attacks on round-reduced keccak. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 216–243, 2017.
25. C. Rechberger, H. Soleimany, and T. Tiessen. Cryptanalysis of low-data instances of full LowMC v2. *IACR Trans. Symmetric Cryptol.*, 2018(3):163–181, 2018.
26. C. Rechberger, H. Soleimany, and T. Tiessen. Cryptanalysis of low-data instances of full lowmcv2. *IACR Trans. Symmetric Cryptol.*, 2018(3):163–181, 2018.
27. L. Song, G. Liao, and J. Guo. Non-full sbox linearization: Applications to collision attacks on round-reduced keccak. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 428–451, 2017.
28. T. Tiessen. Polytopic cryptanalysis. In M. Fischlin and J. Coron, editors, *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*, volume 9665 of *Lecture Notes in Computer Science*, pages 214–239. Springer, 2016.
29. P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12(1):1–28, 1999.