IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2018, No. 3, pp. 163–181. DOI:10.13154/tosc.v2018.i3.163-181

Cryptanalysis of Low-Data Instances of Full LowMCv2

Christian Rechberger[1], Hadi Soleimany[2] and Tyge Tiessen[3]

[1] Graz University of Technology, Graz, Austria
christian.rechberger@tugraz.at
[2] Cyberspace Research Institute, Shahid Beheshti University, Theran, Iran
h_soleimany@sbu.ac.ir
[3] Department of Applied Mathematics and Computer Science (DTU Compute), Technical University of Denmark, Kongens Lyngby, Denmark

Abstract. LowMC is a family of block ciphers designed for a low multiplicative complexity. The specification allows a large variety of instantiations, differing in block size, key size, number of S-boxes applied per round and allowed data complexity. The number of rounds deemed secure is determined by evaluating a number of attack vectors and taking the number of rounds still secure against the best of these.
In this paper, we demonstrate that the attacks considered by the designers of LowMC in the version 2 of the round-formular were not sufficient to fend off all possible attacks. In the case of instantiations of LowMC with one of the most useful settings, namely with few applied S-boxes per round and only low allowable data complexities, efficient attacks based on difference enumeration techniques can be constructed. We show that it is most effective to consider tuples of differences instead of simple differences, both to increase the range of the distinguishers and to enable key recovery attacks. All applications for LowMC we are aware of, including signature schemes like Picnic and more recent (ring/group) signature schemes have used version 3 of the round-formular for LowMC, which takes our attack already into account.

Keywords: Block cipher · Cryptanalysis · Low data · LowMC

1 Introduction

The security of block ciphers, one of the most versatile cryptographic primitives, is commonly believed to be well understood. Well-established block ciphers, such as the Advanced Encryption Standard (AES), have been designed to be both efficient and secure in a wide range of applications.

Novel use-cases however require new designs and also new cryptanalysis. Such use-cases include amongst others masking of block ciphers to thwart side-channel attacks, usage in secure multi-party computation (MPC) or fully homomorphic encryption (FHE), SNARKs, and very recently block ciphers designed for use in quantum-secure public-key signature schemes. Considering these use-cases has lead to a number of cipher designs tailored to the needs of those applications. Examples of such designs include Zorro [GGNPS13], LowMCv2 [ARS+15], Kreyvium [CCF+16], Flip [MJSC16], Rasta [DEG+18] and MiMC [AGR+16]. The main goal in the design of these ciphers is to minimize the number of multiplications in one way or another while retaining security.

One of the techniques that has been used to achieve this goal is to use partial non-linear layers, i.e., in one round a non-linear transformation is only applied to a part of the state. While this is an inherent part of the design of Feistel networks, it is still a relatively new technique in the design of substitution-permutation networks (SPN) and understanding

Received: 2017-11-23, Revised: 2018-06-01, Accepted: 2018-08-01, Published: 2018-09-04

the implications of its usage for the security of the block cipher is an interesting area of research.

LowMC

LowMC is a flexible block cipher family based on a substitution-permutation network where the block size, the key size, the number of S-boxes in the substitution layer and the allowed data complexity of attacks can independently be chosen. To reduce the multiplicative complexity, the number of S-boxes applied in parallel can be reduced, leaving part of the substitution layer as the identity mapping. The number of rounds needed to achieve these goals is then determined as a function of all these parameters. The way this is done is to consider and try to bound all known attack and choose the number of rounds so that the most effective attack for a particular set of parameters is just not able to violate the security expectation.[1]

The first version of this round 'formula' (henceforth LowMCv1) was introduced at Eurocrypt 2015 [ARS+15]. Soon after, optimized higher order and interpolation attacks [DLMW15, DEM16] were demonstrated. As a result, an updated round formula for LowMC (henceforth LowMCv2) was proposed by the designers [ARS+16] to take these new insights into account.

Our Contribution

In this paper, we provide new insight into the security of LowMC by demonstrating distinguishing and key-recovery attacks based on the enumeration of differences that are able to break full-round versions of LowMCv2. These attacks are based on finding collisions in the sets of reachable differences coming from both ends of the cipher. In contrast to differential cryptanalysis [BS90] this approach requires very little data — as little as 3 chosen plaintext-ciphertext pairs. If more data is available, this can be used to extend the number of covered rounds.

For some versions of LowMCv2, the distinguishing attack technique is hitting a boundary, namely when the number of enumerated differences begins to exceed the square root of the number of all possible differences, raising the collision probability close to 1. This occurs when the key size which limits the time complexity is larger than half of the block size. We demonstrate that the enumeration technique can be continued across this boundary by moving from differences to d-differences — a concept introduced in [Tie16].

We furthermore show that full key recovery is possible when using the distinguisher based on d-differences. To optimize the key recovery, we consider an equivalent representation of LowMCv2 were the round keys are only added to the part of the state that went through the non-linear part of the non-linear layer. While this not only allows us to simplify the key recovery, it also lead to a simplified and more efficient implementation of LowMCv2.

Our attack led to a version 3 of the LowMC round formula which takes our attack approach into account. This version was in turn used by the recent proposals for new signature schemes as we discuss in the following.

Impact on applications of LowMC: quantum-safe signature schemes

The attack we describe in this paper is not effective against every possible member of the LowMCv2 family of ciphers. More concretely, it applies well when the number of rounds r is chosen to only give security against an attacker with limited data complexity D but

[1]The number of rounds can vary widely depending on the parameters, especially with changing number of S-boxes per round: From around 10 to many 100s. A Python script is provided by the designers which derives that number of rounds: `https://github.com/LowMC/lowmc/determine_rounds.py`

high allowable time complexity, and most importantly when the number of S-boxes per round m is low.

Incidentally such a corner of the parameter space is especially relevant for newly proposed public-key signature schemes based on NIZK proofs that need as the only cryptographic assumption the security of a one-way function (OWF) or a pseudo-random function (PRF), rather than on other more structured mathematical assumptions.

In view of the NIST selection progress on new public-key primitives [2], and in order to be competitive, such a signature approach cannot rely on already standardized functions like AES or SHA-3, as the resulting signature sizes would be too large. Instead, a new function with fewer multiplications is needed. LowMC was recently proposed to be used as such a function in two independent works on this topic [DOR$^+$16, GCZ16], with the resulting merge [CDG$^+$17] being submitted to the NIST PQ-Crypto process.

LowMC turned out to be the most suitable option for the metric that determines the signature size (the most crucial property): the product of the number of multiplication gates contained in the circuit representation and the ring size in which the multiplications take place (minimal in the case of LowMC as multiplications are binary AND gates). Compared to standard functions, usage of LowMC allows to reduce the signature size by about one order of magnitude [CDG$^+$17]. Very recent follow-up and related work on improved representations and implemnetations [PPRR17], extended functionality like ring- and group signatures [BEF18, DRS18, KKW18] or using alternative zero-knowledge proof systems [KKW18] all use LowMC in the same setting, i.e. with few S-Boxes per round and low allowable data complexity.

In Section 4 we give examples of LowMCv2 parameters with low m and low D and show that a higher number of rounds is needed for security than what the round-formula for LowMCv2 suggests.

Related Work

The idea of using differential trails in a meet-in-the-middle approach has previously been studied in the security analysis of DES [DSP07] and AES [DFJ13, DKS10]. More concretely on AES, it was shown in [GM00] that the mapping function from one active byte to one byte after three rounds depends on only 9 bytes. As a result, the set of mapping functions can be described by a table of size 2^{72} which can be utilized in a meet-in-the-middle attack. Demirci and Selçuk presented an improved cryptanalysis by applying the same idea to four rounds where the set of possible mapping functions can be described by 25 bytes [DS09]. In these attacks the size of the precomputed table is the bottleneck. Dunkelman et. al. presented a technique to decrease the number of variables for parametrizing the mapping functions to 16 bytes, increasing the number of required known plaintexts [DKS10].

In their attack, they consider a truncated differential characteristic for four rounds of AES, in which the input and output differences include a non-zero difference in exactly one byte. This characteristic is utilized to restrict the number of parameters in the mapping functions under the assumption that a pair of plaintexts and corresponding ciphertexts satisfies this differential path. Advanced optimizations and ideas in this line which depend on the linear layer of AES are not applicable to LowMC, though. In follow-up works, similar techniques were applied to improve the cryptanalysis on AES [DFJ13, LJW15] and similar block ciphers [LW15]. The above technique inherently requires a large amount of data since the probability of the truncated differential is usually extremely low (e.g for 4-round AES it is 2^{-120}).

In addition, all mentioned works rely on a specific structural truncated differential characteristic and consequently they highly depend on the inner properties of the cipher. In contrast to that method, our framework does not consider any specific differential

[2] https://csrc.nist.gov/projects/post-quantum-cryptography

characteristic and requires only a minimal amount of data. Our method does not depend on the particular properties of the S-boxes, key schedule, or linear layers.

Also of interest, and complementing our work, an automated search tool was presented in [BDD$^+$15] to find the best differential characteristic and linear approximation in SPN ciphers with partial non-linear layer ciphers.

Paper Organization.

This paper is structured as follows: Section 2 gives a short description of the structure of LowMCv2 and of the notation used in this paper. In Section 3 the technique for finding the distinguisher is described described. In Section 4, we show these distinguishers can be used to build attacks and demonstrate attacks on a number of LowMCv2 versions. To conclude we briefly discuss and compare the performance and limitations of our method with other existing cryptanalytic methods in Section 5. Furthermore we suggest possible future works.

2 Substitution-permutation networks with partial non-linear layers

The cryptanalytic techniques presented in this paper make no use of the specifics of the linear layer and are thus in principle applicable to substitution-permutation networks (SPNs) other than LowMCv2.

To address this, we give here the notation for a general SPN block cipher structure with a partial non-linear layer. After that we give a brief description of LowMCv2. We furthermore show that there is an equivalent description which reduces the size of the round keys allowing for more efficient key-guessing strategies. We will use this later when we mount a key-recovery attack.

2.1 Standard Description

A substitution-permutation network is constructed as a chaining of rounds each of which consists of two layers and a round key addition. The first layer of each round is called the substitution layer. In this layer, S-boxes are applied in parallel to the state to translate it to a new state. Here we consider the more general case where the S-boxes might only be applied to a part of the state — hence the name partial non-linear layers.

The other layer in each round is the permutation layer (or affine layer) which can be any affine transformation of the state. The round key addition adds a round key onto the state using exclusive-or addition. Finally there is an addition of a whitening key before the first round. The round keys and the whitening key are derived from the general key via a key schedule which is a linear function in the case of LowMCv2.

To make this notion a bit more precise, we use the following notation. The number of bits which the SPN operates on, the block size, is denoted as n. The number of rounds is denoted as r where the first round is round 1 and the last is round r. The round keys are denoted as $sk_1, \ldots, sk_r$ and the whitening key as sk_0. The general key is denoted as K and its size in bits is k.

In the substitution layer, we assume that all S-boxes are the same. We denote their width in bits by b and the number of S-boxes applied in parallel in the layer by m. We furthermore assume without loss of generality that the S-boxes are applied to the first mb bits of the state.

2.2 LowMC

LowMC is a family of block ciphers which is based on the SPN structure with partial non-linear layers with flexible parameters which can independently be chosen by users.

Encryption of LowMC starts with a key whitening, followed by rounds each of which consists of four operations in the following order:

1. `SboxLayer` applies a 3-bit S-box $S(a, b, c) = (a \oplus bc, a \oplus b \oplus ac, a \oplus b \oplus b \oplus ab)$ in parallel on the first $3m$ bits of the state while for the remaining $n - 3m$ bits, the identity mapping is applied.

2. `LinearLayer` multiplies the state with an invertible $n \times n$ matrix $\mathcal{L}_i$ which is chosen randomly.

3. `ConstantAddition` is the addition of an n-bit round constant RC_i to the state in $GF(2)$ which is chosen randomly.

4. `KeyAddition` is the addition of an n-bit round key sk_i to the state in $GF(2)$. The round key sk_i is generated by a randomly chosen multiplication of a full-rank $n \times k$ with the master key K in $GF(2)$.

The number of rounds needed to reach the security against several known attacks with reasonable security margins is then derived for any set of block size n, number of Sboxes m, key size k and (logarithmic) allowed data complexity D. As a result of optimized higher order and interpolation attacks [DLMW15, DEM16], the calculation formula for the number of required rounds proposed in the original proposal document is updated by the designers [ARS$^+$16] which is known as LowMCv2.

2.3 Equivalent representation with reduced round key material

The fact that the non-linear layer is partial can be used to reduce the size of the round keys required in each round. To this end, we describe here an equivalent representation of an SPN with partial non-linear layer with reduced round key material.

In the description of an SPN, it is possible to swap the order of the linear layer and the round key addition as both operations are linear. The round key then needs to be exchanged with an equivalent one. For round key sk_i, the equivalent one can be written as $sk'_i = \mathcal{L}_i^{-1}(sk_i)$ where $\mathcal{L}_i$ is the linear layer in the i-th round.

We can use this property now to move parts of the original round keys from the last round all the way through the cipher to the whitening key. To arrive at such a reduced variant, we apply a series of steps to the round keys starting with the last one (see also Section 2.3). First we find an equivalent key that is applied before the affine layer by moving the round key through the affine layer. Then we split the round key in two parts, one that applies to the S-box part of the non-linear layer and one that applies to the identity part of the non-linear layer. The key part that only applies to the non-linear layer part can now move further up where it is merged with the previous round key. If we apply this to all round keys, we finally end up with an equivalent representation in which round keys are only added to the output of the S-boxes apart from one whitening key which is initially applied to the entire state. Note that the round keys of this equivalent representation can still be calculated as linear functions of the master key, albeit using smaller matrices.

We will later use this representation to reduce the amount of key material that we need to guess in an attack. This simplified representation can in certain cases also reduce the implementation cost of an SPN block cipher with a partial non-linear layer. For instance, the standard representation of LowMCv2 requires key matrices of total size $kn(r + 1)$ where k is the key size, n is the block size and r is the number of rounds. The optimized

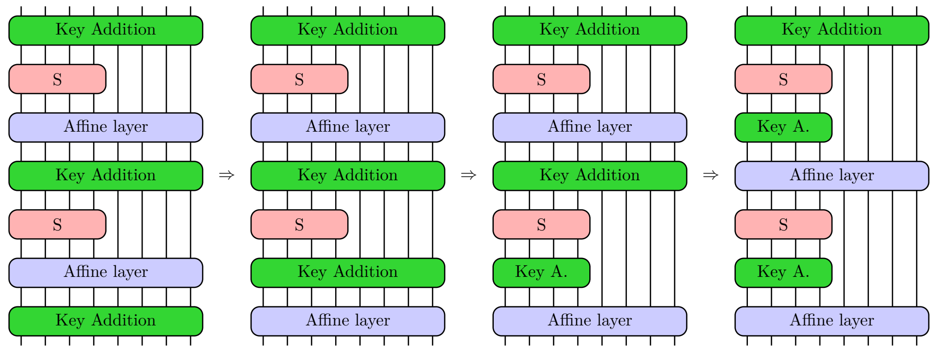

Figure 1: Simplified representation of an SPN block cipher with a partial non-linear layer.

representation only requires $kn + 3mkr$ where m is the number of S-boxes, thus potentially greatly reducing the amount of needed memory and calculation to produce the round keys.

2.4 Notation

In this paper, we use the following notations: X_i denotes the input block of round i, while X_i^S, X_i^L and X_i^O denote the intermediate values after applying nonlinear layer, linear layer and round key addition operations of round i, respectively. Obviously, $X_{i-1}^O = X_i^I$ holds for $2 \leq i \leq r$. The round keys in the standard and simplified representations are denoted by sk_i and sk_i', respectively.

3 Building distinguishers based on difference enumeration

In this section, we describe the distinguishing techniques that we use in the attack and the time and data complexities of applying them. We will first describe a technique that uses simple differences and later extend this to describe a distinguisher that uses the relationships between larger tuples of texts. How to use these distinguishers in a key recovery attack will be described in the next section.

3.1 Enumeration of differences

3.1.1 Using difference enumeration as distinguisher

For a cipher to be secure, we should not be able to predict anything about the difference of two ciphertexts given the difference of the respective plaintexts. Ciphers that fail to accomplish this have successfully been broken using differential cryptanalysis. In this cryptanalytic technique, the attacker is able to find an input difference that yields a non-uniform distribution of output difference. He can then utilize this distinguishing feature to mount a key recovery attack. The downside of this technique is that it usually requires relatively large amounts of plaintext-ciphertext pairs to be able to create a statistically significant distinguisher.

Difference enumeration is a somewhat simpler concept. Here we find an input difference such that we can efficiently create a list of all reachable output differences. Such a list can be generated using the rules of difference propagation as known from standard differential cryptanalysis. If this list is significantly smaller than the set of all possible output differences, we can use this list as a distinguisher: Given an output difference that resulted from the specific input difference, we know it has to be in the list of possible output differences if the texts were generated by the attacked cipher. For a random permutation

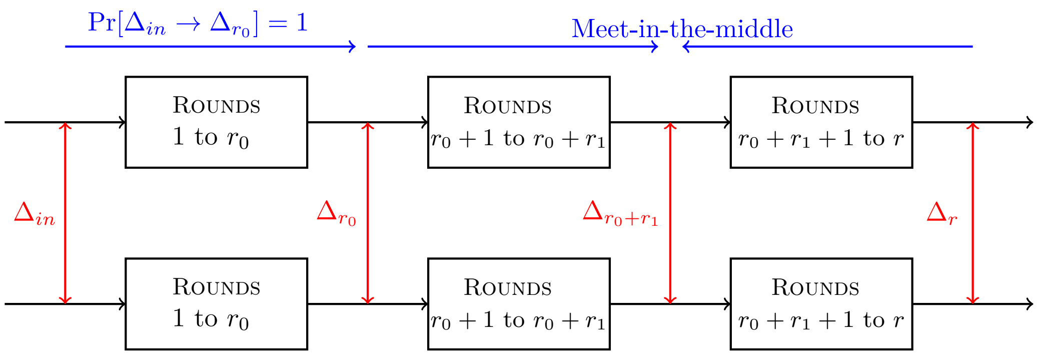
$\Pr[\Delta_{in} \to \Delta_{r_0}] = 1$
Meet-in-the-middle
ROUNDS 1 to r_0
ROUNDS r_0+1 to r_0+r_1
ROUNDS r_0+r_1+1 to r
Δ_{in}
Δ_{r_0}
$\Delta_{r_0+r_1}$
Δ_r
ROUNDS 1 to r_0
ROUNDS r_0+1 to r_0+r_1
ROUNDS r_0+r_1+1 to r

Figure 2: Overview of the technique

on the other hand, the output difference would only be in the list with a probability corresponding to the relative size of the list of all possible differences.

For the distinguisher to be better than an exhaustive search over the key space, the complexity of enumerating all differences must be smaller than the complexity of an exhaustive search.

3.1.2 Enhancing difference enumeration with meet-in-the-middle

Usually the number of reachable differences grows too fast to be efficiently enumerable and thereby usable in a distinguisher but we can significantly decrease the complexity of creating a distinguisher by using a meet-in-the-middle approach.

In contrast to conventional differential cryptanalysis, we do not aim to find differential paths with high probability. Instead, our model benefits from the fact that the number of reachable differences over a few rounds can be much smaller than all possible values. First, we investigate how a difference diffuses while it propagates through the rounds in order to count the number of reachable differences in the middle of the cipher for a specific input difference. Then we show how the internal differences taken by a pair of inputs can be retrieved by utilizing a meet-in-the-middle approach in cases where the number of reachable differences is restricted to an upper bound.

We divide the r rounds of the cipher into three consecutive parts: r_0, r_1, r_2 where $r = r_0 + r_1 + r_2$ (Figure 2). In the following we denote the output difference of the i-th round by Δ_i. We select an input difference $\Delta_0 \in \mathbb{F}_2^n$ so that the output difference after r_0 rounds can be determined with a probability of one. In other words, Δ_0 is selected in a way that does not activate any of the $r_0 m$ Sboxes in the first r_0 rounds. For a successful attack this property should be satisfied for maximum number of rounds.

To this end, we enumerate for a given input difference $\Delta_{r_0} \in \mathbb{F}_2^n$ the number of reachable differences after $r_0 + r_1$ rounds and store them in a list $\mathcal{D}_f$. When we receive an output difference, we likewise enumerate the reachable differences after the $(r_0 + r_1)$th round and store them in a list $\mathcal{D}_b$, only this time going backwards through the cipher over r_2 rounds, starting from the received output difference. Now we expect to see a collision in those enumerated differences and the generated list.

To be useable as a distinguisher, we need to require that the complexity of enumerating the differences and finding the collision is more efficient than an exhaustive search. Furthermore, the probability of finding a collision if we were given instead a random output difference should be less than one.

3.1.3 Estimating the number of reachable differences

A simple upper bound on the number of reachable differences after r rounds given a block cipher structure as described in Section 2 can be found as follows. We assume every S-box is activated in every round and that every S-box creates as many new differences as possible. Let γ be the differential uniformity of the S-box, i.e., the maximal number of distinct differences to which an input difference of the S-box can be mapped. Then the number of differences that a single difference can be mapped to over one round is at most γ^m where m is the number of S-boxes per layer. The number of differences that can be reached from a single difference after r rounds is hence upper bounded by γ^{mr}.

While this is an accurate upper bound, a more precise estimate would take into account that usually not all S-boxes are activated in each round. We can achieve this by working with the average number of reachable output differences. Let λ be the average number of reachable output differences over the S-box for a uniformly randomly chosen input difference (for LowMC $\lambda \approx 29/8 = 3.62$).

We now want to calculate the number of reachable differences over one round given a uniformly randomly chosen input difference. Over one round the number of reachable output differences over each S-box are independent of another. We furthermore know that the expected value of the product of independent random variables is the product of their expected values. Thus the average number of reachable differences over one round given a uniformly randomly chosen input difference can the be calculated as

$$\delta = \lambda^m \tag{1}$$

We can thus estimate the average number of reachable differences over r rounds as

$$\delta^r = \lambda^{mr}. \tag{2}$$

3.1.4 Choosing a good starting difference

To minimize the computational complexity of enumerating or to maximize the number of rounds that can be covered with the distinguisher, the starting difference should be chosen to minimize the increase in the number of differences.

To accomplish this, the attacker can make good use of the partial non-linear layer: a non-linear layer that is not full allows us to cover one or more rounds without activating any S-boxes. The number of rounds that can be covered this way depends on the ratio of the linear part to the non-linear part in the non-linear layer. There are 2^{n-mb} differences that do no activate any S-boxes in the first round. Of the resulting differences after the first round, there are still 2^{n-2mb} differences that do not activate any S-boxes in the second round (assuming $n \leq 2mb$). Continuing this, it is straightforward to see that the maximal number of rounds that a difference can go without activating any S-boxes is

$$r_0 = \left\lceil \frac{n}{bm} \right\rceil - 1. \tag{3}$$

Once the maximum number of rounds has been covered for free, the number of S-boxes that are activated in the following round can be minimized by utilizing the remaining freedom in the differences that can be reached for free up to this point (they form a linear subspace). If r_0 are the number of rounds that were covered for free, at least $\left\lfloor \frac{n-r_0bm}{b} \right\rfloor$ S-boxes can be avoided in the next round.

3.1.5 Complexity of the distinguisher

As it is mentioned in Sec 3.1.2, we separate the number of rounds that we can cover into three parts: r_0, r_1, and r_2. r_0 is the number of rounds that can be covered for free, i.e., the rounds that the input difference is mapped deterministically. After $r_0 + r_1$ rounds we create the list of reachable differences $\mathcal{D}_f$ while we go back the last r_2 rounds from the ciphertext difference to check for a match in this list $\mathcal{D}_b$.

The number of free rounds r_0 should be set to the maximum value as given in Eq. (3). The complexity of creating the list is proportional to the number of reachable differences and can be estimated using the average diffusion δ per round as given in Eq. (2) as δ^{r_1}. As mentioned above, the additional freedom possibly left in the choice of the input difference after maximizing the number of rounds which are passed deterministically by the difference can be used to reduce the diffusion in round r_0+1. Using this we can reduce the complexity of creating the list of differences to

$$|\mathcal{D}_f| = \lambda^{m-\lfloor\frac{n-r_0bm}{b}\rfloor} \cdot \delta^{r_1-1} \tag{4}$$

where λ is again the average number of differences reachable over the S-box.

In case no additional freedom left after maximizing the number of rounds, the complexity of creating the list $\mathcal{D}_f$ equals to

$$|\mathcal{D}_f| = \delta^{r_1} \tag{5}$$

For enumerating the differences after round r_0+r_1 when going back from the ciphertext difference, the complexity again corresponds to the number of reachable differences which can be estimated as follows:

$$|\mathcal{D}_b| = \delta^{r_2} \tag{6}$$

Checking for a collision in the list of differences can be done in constant time. Consequently, the total time complexity is dominated by creating the lists and can be computed as $|\mathcal{D}_f| + |\mathcal{D}_b|$.

3.2 Enumeration of d-differences

One of the limitations of the difference enumeration technique is that the probability of finding a collision in the enumerated differences should be lower than 1 to give a good distinguisher. In other words the following condition should hold to avoid any wrong collision:

$$|\mathcal{D}_f| \times |\mathcal{D}_b| = \delta^{(r_1+r_2)} < 2^n \quad \rightarrow \quad r_1 + r_2 < \frac{n}{\log_2(\delta)} \tag{7}$$

In cases where the key size is larger than half of the block size, this implies that the number of rounds that can be attacked is bounded by the blocksize, not by the time constraint given by the key size.

To circumvent this restriction, it is possible to increase the size of the space where we look for collisions by considering several differences simultaneously. This technique has been named polytopic cryptanalysis [Tie16] and we briefly summarize it here.

3.2.1 About d-differences

In a d-difference instead of looking at the difference of a pair of texts x_0 and x_1, we consider the d differences formed between a base text x_0 and d other texts $x_1, \ldots, x_d$. A d-difference is then the ordered tuple of the respective differences, i.e., $(x_1 \oplus x_0, \ldots, x_d \oplus x_0)$. Just as with a single difference, we can study how these difference tuples propagate through the steps of the cipher.

While the rate of diffusion is generally higher for d-differences, it is nonetheless limited by the size of the S-boxes that are used in the construction. A b-bit S-box can map an input d-difference to at most 2^b possible output d-differences.

3.2.2 Enumerating d-differences

Just as with standard differences, we can enumerate the reachable d-differences that an input d-difference can reach over a given number of rounds. We can thus transfer the distinguisher that we used with differences to a distinguisher based on d-differences. The only change is now that we will be looking for collisions in the enumerations of the d-differences instead of simple differences. We will thus be looking for collisions on dn bits instead of n bits. By increasing the number of differences d that we use, we can thus make sure that the bottleneck is never the block size but always the key size (under the assumption that the data complexity allows that).

Since the number of reachable d-differences over the S-box for a non-zero input d-difference is *at most* 2^b, a simple upper bound on the number of reachable d-differences after r rounds given a block cipher structure as described in Section 2 can be found as $2^{b \cdot m \cdot r}$. However, to calculate the average number of d-difference reachable over r rounds more precisely, we can use the same formula as we used for calculating the diffusion for standard differences but we have to use a value of λ_d that corresponds to the average number of reachable d-differences over one S-box for a uniformly randomly chosen input d-difference. For $d = 2$, this is for example $\lambda_2 \approx 421/64 = 6.58$ in LowMC. λ_2 gets close to the upper bound (i.e 8) by increasing the value of d.

Similarly the average number of reachable d-differences over one round given a uniformly randomly chosen input difference can the be calculated as

$$\delta_d = \lambda_d^m \tag{8}$$

We can thus estimate the average number of reachable differences over r rounds as

$$\delta_d^r = \lambda_d^{mr}. \tag{9}$$

3.2.3 Selection of parameters

To have a d-differences characteristic of probability one for the first r_0 rounds, it is sufficient that $2^{n-bmr_0} > d$ holds. We can select r_0 as the largest possible value:

$$r_0 = \left\lfloor \frac{n - \log_2 d}{b \cdot m} \right\rfloor \tag{10}$$

The data required is only $d+1$ chosen plaintexts for the distinguisher. To make sure that the key-recovery attack succeeds in practice, we run the attack prcedure for two d-differences which requires $2(d+1)$ chosen plaintexts.

We want to consider cases in which merging the lists for obtaining the d-differences in the middle of cipher leads to a unique candidate. Since the number of reachable d-differences is *at most* $\delta_d^{m \cdot (r_1+r_2)}$, the following condition should hold for dimension d to avoid any wrong collision in Algorithm 1:

$$|\mathcal{D}_f| \times |\mathcal{D}_b| = \lambda_d^{m \cdot (r_1+r_2)} = \delta_d^{(r_1+r_2)} < 2^{n \cdot d} \quad \rightarrow \quad d > \frac{\log_2(\delta_d)(r_1+r_2)}{n}. \tag{11}$$

The time complexity is dominated by finding the d-differences collision in the middle of cipher which equals to $\delta_d^{r_1} + \delta_d^{r_2}$ where δ_d is the average number of reachable differences over one round as given in Eq 8. So it makes sense to consider an equal values for r_1 and r_2. The time complexity should be less than the exhaustive search. As a result the following condition should hold:

$$\max\left(\delta_d^{r_1}, \delta_d^{r_2}\right) < 2^k \quad \rightarrow \quad \max(r_1, r_2) < \frac{k}{\log_2(\delta_d)}. \tag{12}$$

To maximize the number of attacked rounds while retaining at least one expected d-differences collision in the middle, we can select r_1 and r_2 to be the largest possible values:

$$r_1 = r_2 = \left\lfloor \frac{k}{\log_2(\delta_d)} \right\rfloor . \tag{13}$$

4 Key-recovery attacks

We start by introducing an algorithm which can be used to obtain the internal d-differences for a specific d-tuple of plaintexts and the corresponding ciphertexts. After that, we discuss how the known internal conventional differences or d-differences can be used to mount a key recovery attack. We will then present the attack results on LowMCv2.

4.1 Recovering the d-differences trail

We begin the key recovery attack with the distinguisher that we constructed in Section 3. In this distinguisher, we computed two lists of reachable d-differences in the middle of the cipher and tested for a collision. As a random permutation would generate such a collision only with a small probability, the occurrence of one could be used to distinguish the cipher from a random permutation. Now in the attack case, we already know that we are dealing with the cipher. But as the collision occurs randomly only with a low probability, with high likelihood only a single collision will be detected.

The first step in the attack is now to determine the d-differences trail that the messages have taken. We already know the input d-differences, the output d-differences and from the collision, we know the d-differences in the middle. Indeed it is straightforward to determine the entire d-differences trail with little addition computational cost. This can for example be done by storing with each d-differences in the lists the d-difference trails in the upper or lower half part of the cipher that was taken to reach it.

To describe this process in more detail, let us assume that there exists an input d-difference $(\Delta_0^1, \cdots \Delta_0^d)$, such that $(\Delta_0^1, \cdots \Delta_0^d) \to (\Delta_{r_0}^1, \cdots \Delta_{r_0}^d)$ holds over the first r_0 rounds with probability one where $\Delta_{r_0}^i = \mathcal{L}_r \circ \cdots \circ \mathcal{L}_1(\Delta_0^i)$ for $1 \le i \le d$. In the following we show how we can retrieve the values of internal d-difference for an arbitrary $(d+1)$-tuple $(P_0, P_1 = P_0 \oplus \Delta_0^1, \cdots, P_d = P_0 \oplus \Delta_0^d)$ and their corresponding ciphertexts:

Step 1: Ask the encryption oracle to provide the encryption of $(P_0, P_1 = P_0 \oplus \Delta_0^1, \cdots, P_d = P_0 \oplus \Delta_0^d)$ and save the corresponding ciphertexts respectively as $(C_0, C_1, \cdots, C_d)$. We denote the output d-difference as $(\Delta_r^1, \cdots, \Delta_r^d)$ where $r = r_0 + r_1 + r_2$.

Step 2: Compute all possible d-differences which can be reached in the output of the $(r_0 + r_1)$-th round from the d-differences $(\Delta_{r_0}^1, \cdots, \Delta_{r_0}^d)$ in the forward direction over the r_1 rounds and save them in the set $\mathcal{D}_f$. Note that we know the values of $(\Delta_{r_0}^1, \cdot, \Delta_{r_0}^d)$ because of the deterministic differential characteristic for the first r_0 rounds.

Step 3: Similarly compute all possible d-differences that can be reached in the output of $(r_0 + r_1)$th round from the difference $(\Delta_r^1, \cdots, \Delta_r^d)$ in the backward direction over the last r_2 rounds and save them in the set $\mathcal{D}_b$.

Step 4: Retrieve the d-difference $(\Delta_{r_0+r_1}^1, \cdots, \Delta_{r_0+r_1}^d)$ by looking for a collision between the sets $\mathcal{D}_f$ and $\mathcal{D}_b$.

If the collision probability is sufficiently low, the retrieved difference in the middle will be uniquely determined. Now if we have a collision, we can connect the paths belonging to the differences in the list to determine the entire difference trail. For this purpose, alternatively Algorithm 1 can be used to obtain all of the internal difference which can be exploitable for key recovery.

Algorithm 1 FIND.MIDDLE.d-DIFFERENCE

```
Require: $R$ and $R'$ where $R < R'$. $(\Delta^d_R, \cdots, \Delta^d_R)$ and $(\Delta^d_{R'}, \cdots, \Delta^d_{R'})$ which are internal
    $d$-differences in rounds $R$ and $R'$, respectively.
Ensure: $d$-difference in the round $\left\lfloor \frac{R+R'}{2} \right\rfloor$
1: Compute all possible $d$-differences that can be reached in the output of the $\left\lfloor \frac{R+R'}{2} \right\rfloor$-th
   round from $d$-difference $(\Delta^d_R, \cdots, \Delta^d_R)$ in the $R$-th round and save them in the set $\mathcal{D}_f$.
2: Compute all possible $d$-differences that can be reached in the output of $\left\lfloor \frac{R+R'}{2} \right\rfloor$-th
   round from the $d$-difference $(\Delta^d_{R'}, \cdots, \Delta^d_{R'})$ over the last round and save them in the
   set $\mathcal{D}_b$.
3: Match the sets $\mathcal{D}_f$ and $\mathcal{D}_b$ and return the collision.
```

In order to find all internal d-differences, one should apply meet-in-the-middle approach for around r_1+r_2 times. In each iteration, the time complexity is dominated by constructing the lists which is proportional to the size of the corresponding created lists $|\mathcal{D}_f| + |\mathcal{D}_b|$, since finding a collision in the lists can be done in constant time. The number of reachable differences grows exponentially by increasing the number of rounds. Consequently the total time complexity of finding internal d-differences is dominated by finding the first internal d-differences collision in the middle of cipher which equals to

$$\delta_d^{r_1} + \delta_d^{r_2} \tag{14}$$

Similarly the memory complexity is dominated for saving the possible d-differences for the first call of MITM approach which is $(\delta_d^{r_1} + \delta_d^{r_2}) \cdot d \cdot (n/8)$ bytes to allocate $\delta_d^{r_1} + \delta_d^{r_2}$ memories for saving d values.

4.2 Constructing a full key recovery attack

4.2.1 Retrieving all equivalent subkeys by utilizing difference trail

In what follows we describe the method to retrieve the key based on the knowledge of internal differences. We denote by 2^x the maximum number of solutions for the equation $\beta = S(X) \oplus S(X \oplus \alpha)$ where $\alpha, \beta \in \mathbb{F}_2^b$. In other words we assume that the b-bit Sbox is 2^x-uniform (for LowMC $2^x = 2$). On the basis of the solutions for the equation $\beta = S(X) \oplus S(X \oplus \alpha)$, we present a method to obtain round keys by considering two consecutive differences.

To describe this process in more detail, we consider a pair of plaintexts $(P, P' = P \oplus \Delta_{in})$ and the corresponding ciphertexts (C, C') where $\Delta_{in} \in \mathbb{F}_2^n$. There exists a unique differential path from plaintexts to ciphertexs over r rounds of the cipher that correspond to this pair. This path directly depends on the values of rounds keys and can be found by the method described in Sec 4.1. We denote the output difference of the i-th round by Δ_i where $\Delta_i \in \mathbb{F}_2^n$ and $1 \leq i \leq r$. Obviously $\Delta_r = C \oplus C' = \Delta_{out}$. In addition we denote the internal states in the i-th round which correspond to the pairs (P, C) and (P', C') by X_i and X'_i, respectively.

Let us assume that the difference of the semi-final round, i.e. Δ_{r-1}, is known. In addition, the transition difference from Δ_{r-1} to $C \oplus C' = \Delta_{out}$ is not deterministic, i.e. $\Pr[\Delta_{r-1} \to \Delta_{out}] < 1$. Usually the linear operation is omitted in the last round of the cipher. Nevertheless we assume the last round includes the linear layer $\mathcal{L}_r$ which can simply be considered as an identity function in the case of nonexistence. We expect to have at most $2^{m.x}$ solutions for the quadratic $(X_r^I, X_r^{'I}, X_r^S, , X_r^{'S})$, since each Sbox is differentially 2^x-uniform. Each solution uniquely suggests a candidate for the equivalent round key sk'_r as follows:

$$C \oplus sk'_r = X_r^L = \mathcal{L}_r(X_r^S) \to sk'_r = C \oplus \mathcal{L}_r(X_r^S)$$

So totally $2^{m.x}$ values are obtained as candidates for the equivalent round key sk'_r which is significantly less than all $2^{m.b}$ possible values. In other words we get $m \cdot (b-x)$-bit information about the last equivalent round key sk'_r.

Now let us assume that ℓ different pairs of plaintexts $(P_i, P_i \oplus \Delta_{in})$ with corresponding ciphertexts C_i, C'_i are given. In addition, we assume that the internal differences for each of the pairs can be retrieved uniquely with the method described in Section 3. With the method described above, $m \cdot (b-x)$-bit information about the last equivalent round key can be retrieved from each pair. Consequently, the number of pairs needs to retrieve sk'_r can be estimated as follows:

$$\ell = \lceil \frac{m \cdot b}{m \cdot (b-x)} \rceil = \lceil \frac{b}{b-x} \rceil. \tag{15}$$

Our key-recovery attack takes advantage of the fact that for any arbitrary differences $(\alpha, \beta) \in (\mathbb{F}_2^b \times \mathbb{F}_2^b)$ the number of solutions for the equation $\beta = S(x) \oplus S(x \oplus \alpha)$ is significantly smaller than 2^b. This property is an obvious design criterion from the point of view of cryptographers. To guarantee a strong resistance against differential-type cryptanalysis, Sboxes are built upon functions with low differential uniformity. Interestingly the data required for retrieving the equivalent subkey in our attack decreases when the S-box utilized in the cipher is stronger against differential attack as it can be observed in Eq. (15). Since LowMC S-box is 2-uniform, the attack on LowMC requires around $\lceil \frac{3}{3-1} \rceil = 2$ pairs of chosen plaintexts. However, we can use a few more pairs of chosen plaintexts to make sure we can find different differences over S-boxes in the key-recovery part. To illustrate this fact, let us consider the following simple example:

Example 4.1. For the sake of simplicity we consider one Sbox in the last round of LowMC excluding the linear layer. The following relation holds:

$$S(y) + k = c$$

where k is a 3-bit fixed but unknown, y is a 3-bit input of the last round and c is the corresponding 3-bit in the ciphertext. Let us assume that for a given pair $((P_1, C_1), (P'_1, C'_1))$, the difference of the semi-final round is found as 1. In addition, we assume $c_1 = 0$ and $c'_1 = 5$ which means the input and output differences over the Sbox in the last round are respectively 1 and 5. The internal value $S(y_1)$ is either 3 or 6, since $S(2) \oplus S(3) = 3 \oplus 6 = 5$. Consequently, the key $k = c_1 \oplus S(y_1)$ is either $0 \oplus 3 = 3$ or $0 \oplus 6 = 6$. Similarly assume for another given pair $((P_2, C_2), (P'_2, C'_2))$, the input and output differences over the Sbox in the last round are 1 and 1, respectively. In addition, assume $c_2 = 3$ and $c'_2 = 2$. The corresponding internal values for $S(y_2)$ is either 0 or 1, since $S(0) \oplus S(1) = 0 \oplus 1 = 1$. Consequently, the key $k = c_2 \oplus S(y_2)$ is either $3 \oplus 0 = 3$ or $3 \oplus 1 = 2$. The key can be obtained uniquely by considering the intersection between two sets $\{3, 6\} \cap \{3, 2\}$=3.

For the obtained equivalent last round key sk'_r, all ciphertexts C_i and C'_i can be decrypted over the last round. Then the sk'_{r-1} can be obtained with the similar method by considering the differences Δ_{r-1} and Δ_{r-2}. The same arguments suggest that sk'_{r-1} can be determined by uniquely. We can simply continue this procedure over $r_1 + r_2$ rounds to obtain the all equivalent subkeys uniquely.

The time complexity of the key-recovery attack is $(r_1 + r_2) \cdot 2^\ell \cdot 2^{m \cdot x}$ memory accesses and simple operations. Obliviously the time complexity of the key-recovery attack is much smaller than the time complexity of the process of finding internal differences described in Section 4.1 which is equal to $\delta^{r_1} + \delta^{r_2}$ for each pairs as it is give in Eq. (14). So the total complexity of the attack can be estimated as follows:

$$\ell \cdot (\delta^{r_1} + \delta^{r_2}) \tag{16}$$

where ℓ is the number of required pairs.

4.2.2 Retrieving all equivalent subkeys by utilizing d-differences

Because of the existence of symmetric solutions, the lowest number of solutions for a pair of input and output differences over an S-box is 2. However, the situation is different for d-differences. While a b-bit S-box can map an input d-difference to at most 2^b possible output d-differences, the number of possible output d-differences increases exponentially with the dimension d, i.e. $2^{d \cdot b}$. It is easy to verify that the d-difference distribution is sparse for $d > 1$ and for most pairs of input and output d-differences over an S-box there exist a unique solution. For instance, the number of possible input and output 2-differences pairs over the Sbox of LowMC is 421 out of $2^{2.3} \cdot 2^{2.3} = 2^{12}$ total values. For $336/421 = 79\%$ possible input and output 2-differences there exist only one solution. By increasing the dimension d, this ratio becomes higher. For instance, there exist a unique solution for $3696/3893 = 94\%$ possible input and output 3-differences over the Sbox of LowMC. Consequently, if we move from differences to d-differences where $d > 1$, the problem becomes a lot easier. If we know the input d-difference and the output d-difference over a LowMC S-box and if there are at least two unique non-zero differences among the d differences, the values of the input and output messages are uniquely determined.

If we are given the plaintext and ciphertext messages and their corresponding d-difference trail, we can thus determine for any active S-box in the last round, the value of the corresponding part of the last round key uniquely. By running the same procedure two or a few more times to activate different S-boxes in the last round, we can retrieve the last round key completely (in our equivalent representation of LowMC). We can thus peel of the last round and use the same data to retrieve the second to last equivalent round key, and so forth.

Similarly the time complexity of retrieving the equivalent subkeys is negligible in comparison with the time complexity of finding internal differences described in Section 4.1 which is equal to $\delta_d^{r_1} + \delta_d^{r_2}$ for each pair as it is give in Eq. (14). So the total complexity of the attack is dominated by the process of finding internal differences for two different pairs and can estimated as:

$$2 \cdot (\delta_d^{r_1} + \delta_d^{r_2}) \tag{17}$$

4.2.3 Full key from equivalent round keys

In general, the exact amount of information extracted about the master key depends on the key schedule of the cipher. However, as the key schedule of LowMC is linear, we only need to determine enough round key material to ensure that the full key can be determined uniquely. Since the key schedule is generated pseudo-randomly, this should be the case as soon as the collected round key material exceeds the size of the full key.

4.3 Results on LowMCv2

To estimate the security of LowMCv2 against the described attack, we can take two different approaches. The first is to compare the time complexity of the proposed attack on the full-round cipher with the given threshold 2^k which is the time complexity of exhaustive search over all key candidates. In Table 1 we list resulting attacks on a few different instances of LowMCv2 with low allowable data complexity (enough to allow our attack vector to succeed with high probability) and a very small number of S-Boxes per round. The time complexity of the attack is proportional only to the values of r_1 and

Table 1: Full-round attacks on different versions of LowMCv2. Data is given in number of chosen plaintexts. Block and key size are given in bit.

Cipher Specification					Attack Details					
Block	S-boxes	Data	Key	Rounds	Dimension	r_0	r_1	r_2	Time Complexity	Data
n	m	D	k	r	d	$\lfloor \frac{n-\log_2 d}{3\cdot m} \rfloor$	$\lfloor \frac{r-r_0}{2} \rfloor$	$\lceil \frac{r-r_0}{2} \rceil$	$2\cdot(\delta_d^{r_1}+\delta_d^{r_2})$	$2(d+1)$
128	1	16	256	158	4	41	58	58	$2^{164.9}$	10
128	5	16	256	37	4	8	14	15	$2^{212.75}$	10
256	1	8	256	243	2	85	79	79	2^{223}	6
256	5	8	256	53	2	17	18	18	$2^{254.9}$	6
512	1	8	256	413	1	170	121	121	$2^{226.6}$	4
1024	1	8	512	758	1	341	208	209	$2^{389.9}$	4

Table 2: Maximum number of attacked rounds for different versions of LowMCv2. Time complexity is in all cases just below what the key size allows. Data is given in number of chosen plaintexts. Block and key size are given in bit.

Cipher Specification					Attack Details					
Block	S-boxes	Data	Key	Rounds	Dimension	r_0	r_1, r_2	Time	Data	Max. Rounds
n	m	D	k	r	d	$\left\lfloor \frac{n-\log_2 d}{3\cdot m} \right\rfloor$	$\left\lfloor \frac{k}{\log_2(\delta_d)} \right\rfloor$	$2(\delta_d^{r_1}+\delta_d^{r_2})$	$2(d+1)$	$r_0+r_1+r_2$
128	1	16	256	158	4	41	84	2^{254}	10	209
128	5	16	256	37	4	8	16	2^{254}	10	40
256	1	8	256	243	2	85	89	$2^{252.3}$	6	263
256	5	8	256	53	2	17	17	$2^{254.9}$	6	53
512	1	8	256	413	1	170	136	$2^{254.41}$	4	442
1024	1	8	512	758	1	341	274	$2^{510.54}$	4	889

r_2. Consequently, we choose r_0 as the largest possible value presented in Eq. (10). We cover the remaining rounds by selecting $r_1 = \lfloor \frac{r-r_0}{2} \rfloor$ and $r_2 = \lceil \frac{r-r_0}{2} \rceil$ (almost) equally to decrease the time complexity of the attack presented in Eq. (14). As can be seen from the Table 1, several low-data instances of LowMCv2 can be broken significantly faster than exhaustive search.

A second approach is to focus on determining the maximal number of rounds which are still attackable with a complexity marginally below exhaustive search. The gap between this number and the number of rounds deemed secure in LowMCv2 is thus indicative of the instantiation's vulnerability. We apply the round formula given in Sec 3.2.3 to derive the maximized number of cipher rounds vulnerable to the attack. To maximize the number of attacked rounds, we select r_0 and r_1, r_2 to be the largest possible values as proposed in Eq. (10) and Eq. (13), respectively. In Table 2 we list resulting attacks on LowMCv2 as can be seen in the 'Max. rounds' column.

We exemplify the numbers for the attack on the first example in the table where we have a 128-bit state, one S-box per layer, 16 allowed chosen plaintext/ciphertext pairs and a 256-bit key. In the best attack, we use 4-differences such that a single distinguisher requires 5 chosen messages. To ensure that we have enough active S-boxes for the round key recovery, we double this number to allow for a second independent distinguisher. The number of attacked rounds is determined using the results of Section 3. First we can cover 41 rounds for free. Then after 84 additional rounds we construct the first list of 4-differences. Coming from the ciphertext end, we can cover 84 rounds where we then search for the collision in the list. This gives in total 209 attacked rounds.

By increasing the dimension d, δ_d becomes higher which leads to the growth of the time complexity. On the other hand the probability of false collisions in MITM step becomes very low by increasing d. Consequently, as it can be seen in Table 1 we select different scenario based on the block size (n) and the key size (k). For cases which $n > k$, we use standard differential. For cases which $n = k$, we use 2-differences. Finally for cases which $n < k$, we choose $d > 3$.

4.4 Experimental verification

To verify the theoretical model of attacks proposed in sec 4.2, we implement the described attack on a small variant of LowMC(20,1,3,20). As suggested by the designers, we use the Grain LSFR as self-shrinking generator to pseudorandomly binary matrices for linear layers, key schedule and values of round constants. As it is shown in Eq. (10), the maximal number of rounds that a difference can go without activating any S-boxes is estimated as $r_0 = \left\lceil \frac{n-log_2(d)}{bm} \right\rceil - 1 = \lceil \frac{19}{3} \rceil - 1 = 6$. In our experience there exist 3 deterministic differential characteristic over 6 rounds: $(\texttt{0x0C3AA} \rightarrow \texttt{0xE69AB})$, $(\texttt{0x1CAD9} \rightarrow \texttt{0x973E0})$ and $(\texttt{0x10973} \rightarrow \texttt{0x71A4B})$. We take two input 2-differences $(\alpha_1, \alpha_2) = \texttt{(0x0C3AA,0x1CAD9)}$ to have a deterministic 2-differences $(\alpha_1, \alpha_2) \rightarrow (\beta_1, \beta_2)$ over the first 6 rounds where $(\beta_1, \beta_2) = \texttt{(0xE69AB,0x973E0)}$. For a random anchor P_0 we encrypt the plaintexts $(P_0, P_1 = P \oplus \alpha_1, P_2 = P \oplus \alpha_2)$ over 18 rounds and save their corresponding ciphertexts as $C_0 = \texttt{79E52}, C_1 = \texttt{0xC596B}$ and $C_2 = \texttt{0x3AE8C}$, respectively.

After that we compute all possible 2-differences which can be reached in the output of the 12-th round from the 2-difference (β_1, β_2) in the forward direction over 6 rounds and save them in a set $\mathcal{D}_f$. Our Experience shows the number of reachable 2-differences in forward direction is $|\mathcal{D}_f| = 46863 \simeq 2^{15.51}$. We also compute all possible differences that can be reached in the output of 12'th round from the 2-difference $\Delta_{18} = (C_0 \oplus C_1, C_0 \oplus C_2) = \texttt{(0xBC739,430DE)}$ in the backward direction over the last 6 rounds and save them in another set $\mathcal{D}_b$. Similarly our experience shows the number of reachable 2-differences is $|\mathcal{D}_b| = 60183 = 2^{15.87}$. We repeat our experience for different random matrices in the linear layers for both forward and backward directions. We always reach less than $2^{16.1}$ candidates for the 2-differences over 6 rounds which is less than the estimation $\delta_2^6 = \lambda_2^6 = 2^{16.8}$ given in Eq. (9). This fact can facilitate the attack procedure in practice. We finally retrieve the 2-differences $\Delta_{12} = \texttt{(0x3B203,0xFEFF7)}$ uniquely by looking for a collision between the sets $\mathcal{D}_f$ and $\mathcal{D}_b$.

We similarly obtain other 2-differences of the last rounds by using the same method described in Algorithm 1 which leads to retrieve the equivalent subkeys. In particular, the 2-difference in the semi-last round obtained as $\Delta_{17} = (\texttt{0xC5023}, \texttt{0xEDACA})$ which equals to the input 2-difference of the non-linear layer of last round. The output 2-difference of the non-linear layer of the last round is $\mathcal{L}_{18}^{-1}(\Delta_{18}) = \mathcal{L}_{18}^{-1}(\texttt{0xBC739,430DE}) = (\texttt{0x65023,0x4DACA})$. Both differences in Δ_{17} and $\mathcal{L}_{18}^{-1}(\Delta_{18})$ are equal in the last 17 bits as we expected, since non-linear layer includes only one S-box and covers the first 3 bits. The corresponding input and output 2-differences over the S-box in the last round is $(\texttt{0x3}, \texttt{0x2})$ and $(\texttt{0x6}, \texttt{0x7})$, respectively. There exist only one solution for this transmission over the Sbox: $(S(\texttt{3}) \oplus S(\texttt{0}), S(\texttt{3}) \oplus S(\texttt{1})) = (\texttt{6} \oplus \texttt{0}, \texttt{6} \oplus \texttt{1}) = (\texttt{6,7})$. So we obtain the first 3-bits of the state after the S-box corresponding to the pair (P_0, C_0) uniquely as 3. By considering the first 3-bits of $\mathcal{L}_{18}^{-1}(C_0) = \texttt{A9F75}$ the equivalent subkey in the last round can be obtained as $sk'_{18} = 3 \oplus 5 = 6$. In our experience, other equivalent subkeys can be found similarly by utilizing at most two different 2-differences.

5 Conclusion

In this paper we provided new insight into the security of LowMCv2. We demonstrated that some versions of LowMCv2 with sufficiently sparse non-linear layers and low allowed data-complexity are vulnerable to attacks based on difference enumeration. We further demonstrated how these attacks could be made more generic by considering tuples of differences — d-differences.

Indeed that is exactly the parameter space relevant for recently important for LowMC's use-case in post-quantum signature schemes [CDG+17, KKW18, BEF18, DRS18, PPRR17]. This is a result of the fact that the overall number of multiplications is minimized by decreasing the number of S-boxes per layer and the fact that only low-data security is required in this class of application. Thereby, our cryptanalysis turns out to be applicable on an important category of the LowMCv2 family that are utilized in real-world applications. All the above mentioned applications of LowMC do take our attacks into account already in their parameterization of LowMC as they used from their start version 3 of the round formular of LowMC.

While the impact on LowMCv2 is clear, it is an open question whether the attack can be effective on other designs with partial non-linear layers as well. It is furthermore an interesting question in itself how to retrieve the full key if we are given only a single pair of input and output messages together with the difference trail that they took.

Acknowledgments

The work of Hadi Soleimany is partly supported by grants from Shahid Beheshti University and by the Iranian National Science Foundation (grant n°95835673). Christian Rechberger has partly been supported by H2020 project PRISMACLOUD, grant agreement n°644962.

References

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10031, pages 191–219. Springer, 2016.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.

[ARS+16] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. *IACR Cryptology ePrint Archive*, 2016:687, 2016.

[BDD+15] Achiya Bar-On, Itai Dinur, Orr Dunkelman, Virginie Lallemand, Nathan Keller, and Boaz Tsaban. Cryptanalysis of SP Networks with Partial Non-Linear Layers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 315–342. Springer, 2015.

[BEF18] Dan Boneh, Saba Eskandarian, and Ben Fisch. Post-Quantum EPID Group Signatures from Symmetric Primitives. Cryptology ePrint Archive, Report 2018/261, 2018. https://eprint.iacr.org/2018/261.

[BS90] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[CCF+16] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 313–333. Springer, 2016.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1825–1842. ACM, 2017.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 662–692. Springer, 2018.

[DEM16] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Higher-Order Cryptanalysis of LowMC. In Soonhak Kwon and Aaram Yun, editors, *ICISC 2015*, volume 9558 of *LNCS*, pages 87–101. Springer, 2016.

[DFJ13] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 371–387. Springer, 2013.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 158–176. Springer, 2010.

[DLMW15] Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized Interpolation Attacks on LowMC. In Soonhak Kwon and Aaram Yun, editors, *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 535–560. Springer, 2015.

[DOR+16] David Derler, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, and Daniel Slamanig. Digital signatures from symmetric-key primitives. Cryptology ePrint Archive, Report 2016/1085, 2016. http://eprint.iacr.org/2016/1085.

[DRS18] David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, volume 10786 of *Lecture Notes in Computer Science*, pages 419–440. Springer, 2018.

[DS09] Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2009.

[DSP07] Orr Dunkelman, Gautham Sekar, and Bart Preneel. Improved meet-in-the-middle attacks on reduced-round DES. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, volume 4859, pages 86–100. Springer, 2007.

[GCZ16] Steven Goldfeder, Melissa Chase, and Greg Zaverucha. Efficient Post-Quantum Zero-Knowledge and Signatures. Cryptology ePrint Archive, Report 2016/1110, 2016. http://eprint.iacr.org/2016/1110.

[GGNPS13] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block Ciphers That Are Easier to Mask: How Far Can We Go? In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2013.

[GM00] Henri Gilbert and Marine Minier. A collision attack on 7 rounds of rijndael. In *AES Candidate Conference*, pages 230–241, 2000.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. Cryptology ePrint Archive, Report 2018/475, 2018. https://eprint.iacr.org/2018/475.

[LJW15] Leibo Li, Keting Jia, and Xiaoyun Wang. Improved single-key attacks on 9-round AES-192/256. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 127–146. Springer, 2015.

[LW15] Li Lin and Wenling Wu. Meet-in-the-middle attacks on reduced-round midori-64. Cryptology ePrint Archive, Report 2015/1165, 2015. http://eprint.iacr.org/2015/1165.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016*, volume 9665 of *LNCS*, pages 311–343. Springer, 2016.

[PPRR17] Léo Perrin, Angela Promitzer, Sebastian Ramacher, and Christian Rechberger. Improvements to the linear layer of lowmc: A faster picnic. Cryptology ePrint Archive, Report 2017/1148, 2017. https://eprint.iacr.org/2017/1148.

[Tie16] Tyge Tiessen. Polytopic cryptanalysis. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016*, volume 9665 of *Lecture Notes in Computer Science*, pages 214–239. Springer, 2016.