Higher-order cryptanalysis of LowMC

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel

IAIK, Graz University of Technology, Austria
maria.eichlseder@iaik.tugraz.at

Abstract. LowMC is a family of block ciphers developed particularly for use in multi-party computations and fully homomorphic encryption schemes, where the main performance penalty comes from non-linear operations. Thus, LowMC has been designed to minimize the total quantity of logical "and" operations, as well as the "and" depth. To achieve this, the LowMC designers opted for an incomplete S-box layer that does not cover the complete state, and compensate for it with a very dense, randomly chosen linear layer. In this work, we exploit this design strategy in a cube-like key-recovery attack. We are able to recover the secret key of a round-reduced variant of LowMC with PRESENT-like security, where the number of rounds is reduced from 11 to 9. Our attacks are independent of the actual instances of the used linear layers and therefore, do not exploit possible weak choices of them. From our results, we conclude that the resulting security margin of 2 rounds is smaller than expected.

Keywords: cryptanalysis, LowMC, higher-order cryptanalysis, key recovery, zero-sum distinguisher

1 Introduction

The recently proposed family of block ciphers LowMC [1] addresses the need for new block cipher structures suited for multi-party computation and fully homomorphic encryptions schemes, where the non-linear operations of a cipher contribute much more to the overall computational execution costs than the linear operations. Therefore, LowMC combines an incomplete S-box layer with a strong linear layer to reduce the multiplicative depth and size of the cipher. However, this is a quite uncommon approach and can be risky as shown for Zorro [3,8,13,16]. Therefore, LowMC comes with strong security arguments (bounds) against standard cryptanalytic attacks like differential and linear cryptanalysis. In more detail, the authors show that for the proposed instances of LowMC, no good differential and linear characteristics exist for more than 5 rounds. However, they do not provide such strong security arguments against other attack vectors including algebraic attacks.

In this work, we show that the security of LowMC against algebraic attacks is lower than expected. Our attacks are based on the ideas previously used in cube attacks [7], higher order differential cryptanalysis [12], AIDA [15], bit-pattern based integral attacks [17], or the square [6] and intergral [11] attacks. To be more specific, our attacks make use of the rather low algebraic degree of one round of LowMC to construct cube testers [2]. The fact that the S-box layers are incomplete can be exploited to efficiently construct vector spaces at the input which allow to create cube testers of low dimension covering a rather high number of rounds. The incomplete S-box layer also facilitates the existence of linear relations with probability 1, which allow to attack additional rounds. This leads to attacks on round-reduced variants of LowMC with PRESENT-like security, where the number of rounds is reduced from 11 to 9. Note that these attacks do not exploit any specific property of the linear layers and are thus applicable for randomly chosen linear layers.

Our results show that the security margin of LowMC with PRESENT-like security is smaller than expected, being only 2 rounds. Therefore, we conclude that the design of primitives with an incomplete S-box layer has not been fully understood yet. Therefore, it is recommendable to be more conservative when choosing the security margin in those designs.

2 Description of LowMC

LowMC [1] is a family of block ciphers, where each instance is characterized by the following parameters:

- n: block size,
- k: key size,
- m: number of S-boxes per substitution layer,
- d: logarithmic data complexity limit,
- r: number of rounds,
- the concrete instantiations of the linear layers f_L, and
- the key derivation function used in f_K.

The encryption of LowMC starts with a key whitening layer $f_K^{(0)}$, followed by r iterative applications of the round function

$$f^{(i)} = f_K^{(i)} \circ f_L^{(i)} \circ f_S,$$

consisting of the substitution layer f_S (identical for each round), the linear layer $f_L^{(i)}$, and the round-key addition $f_K^{(i)}$, as illustrated in Fig. 1.

The substitution layer is the parallel application of the same 3-bit S-box $S(a, b, c) = (a \oplus bc, a \oplus b \oplus ac, a \oplus b \oplus c \oplus ab)$ on the right-most (least significant) $3 \cdot m$ bits of the state. On the remaining $\ell = n - 3m$ bits, the identity mapping is applied. The used S-box has a maximum differential and linear probability of 2^{-2}, and algebraic degree 2 (the maximum possible degree for a 3-bit permutation).

The linear layer f_L multiplies the n-bit state with a randomly chosen and invertible $n \times n$ matrix over $\mathbb{F}_2$. The matrix differs for each round.

In $f_K^{(i)}$, the whitening key K_0 and round keys $K_1, \ldots, K_r$ are added in the respective rounds. These round keys are generated by binary multiplications of

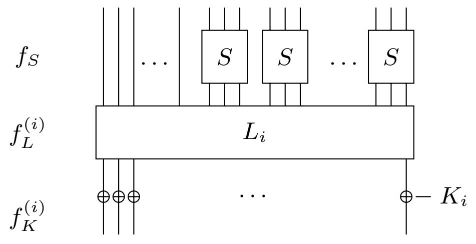

Fig. 1: The round function of LowMC: $f^{(i)} = f_K^{(i)} \circ f_L^{(i)} \circ f_S$.

randomly generated, full-rank $n \times k$ matrices with the master key K, followed by an addition with an randomly chosen n-bit round constant.

Albrecht et al. [1] propose two concrete instances, with the parameter sets shown in Table 2. The first set provides PRESENT-like security using an 80-bit key, while the second set provides AES-like security using a 128-bit key. To generate the used random matrices, the recommended instantiations use the Grain LSFR [9] as a source of random bits. Since our analysis does not depend on concrete instantiations of matrices, only on the parameters given in Table 2, we omit a description of the matrices.

3 Higher-order attacks in the known-key setting

In the known-key setting [10], we assume that the round keys have known values. The attack goal is to find non-random properties of the resulting permutation. More specifically, we will focus on (families of) zero-sum distinguishers: finding sets of inputs to the permutation such that both the sum (over $\mathbb{F}_2^n$) of the inputs, as well as the sum of their outputs, equal zero.

3.1 Basic zero-sum distinguisher

A well-known result from the theory of Boolean functions is that if the algebraic degree of a vectorial Boolean function (like a permutation) is d, then the sum over the outputs of the function applied to all elements of a vector space of dimension $\geq d+1$ is zero (as is the sum of all inputs, i.e., the elements of the vector space). The same property holds for affine vector spaces of the form $\{v + c \mid v \in V\}$ for some vector space V and constant c. Therefore, in the remaining text, we also refer to affine vector spaces as vector spaces for simplicity. This property allows to exploit a low algebraic degree of cryptographic functions to create zero-sum distinguishers and has been applied, for example, to the hash function Keccak [4,5].

For this reason, the designers of LowMC included bounds for the algebraic degree of multiple rounds of the permutation in their design paper [1]. Their bounds are based on the observation that if the degree after r rounds (with m

S-boxes per round of the n-bit permutation) is $d_r^{(n,m)}$, then the degree $d_{r+1}^{(n,m)}$ after $r+1$ rounds is bounded by

$$d_{r+1} \leq \min\left\{2 \cdot d_r, \quad m + d_r, \quad \frac{n}{2} + \frac{d_r}{2}\right\},$$

since the degree of one round is $d_1 = 2$. The resulting bounds for up to 15 rounds are given in Table 1.

Table 1: Upper bounds for the algebraic degree $d_r^{(n,m)}$ after r rounds of the LowMC permutation on $n = 256$ bits with $m \in \{49, 63\}$ S-boxes.

r	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$d_r^{(256,49)}$	2	4	8	16	32	64	113	162	209	232	244	250	253	254	255
$d_r^{(256,63)}$	2	4	8	16	32	64	127	190	223	239	247	251	253	254	255

Based on these numbers, the designers recommend that the number of rounds r satisfies

$$r \geq r_{\text{deg}} + r_{\text{outer}},$$

where r_{deg} is the number of rounds necessary for a sufficiently high degree ($d_{r_{\text{deg}}} \geq d - 1$ for the logarithmic data complexity limit d), and $r_{\text{outer}} = 5$ is a heuristic estimate for the number of rounds that can be "peeled off" in the beginning and end of the cipher, based on the bounds for linear and differential cryptanalysis. This leads to the round numbers stated in Table 2 for the recommended parameter sets.

Table 2: Recommended number of rounds $r \geq r_{\text{deg}} + r_{\text{outer}}$ for recommended parameter sets [1].

Key size k	Block size n	S-boxes m	Data complexity d	r_{deg}	r_{outer}	Rounds r
80	256	49	64	6	5	11
128	256	63	128	7	5	12

The degree bounds from Table 1 clearly show that 11 or 12 rounds of the unkeyed round function are no ideal random permutation, although the complexity of a straightforward zero-sum distinguisher is far beyond the claimed security level: If we choose any subspace $V \leq \mathbb{F}_2^{256}$ with dimension ≥ 245 (resp. ≥ 252), we get

$$\sum_{v \in V} v = \sum_{v \in V} f^{11}(v) = 0$$

(resp. $\sum_{v\in V} f^{12}(v) = 0$) for LowMC-80 (resp. LowMC-128) with $m = 49$ (resp. $m = 63$) S-boxes per round, where f is the round with some fixed, known key. However, it is very easy to obtain distinguishers with a much lower complexity.

3.2 Initial structure, direct-sum construction, and partial zero-sums

First, since we are considering the known-key setting, we are not limited to starting computation before the first round. We can also define an initial structure as input for one of the middle rounds – say, round 7 – and compute backwards and forwards from there to again get zero-sums at input and output, with a much lower data complexity. Since LowMC uses 3-bit bijective S-boxes, the degree of the inverse S-box and thus of the inverse round function f^{-1} is also at most 2. Thus, for any subspace $V \le \mathbb{F}_2^{256}$ with dimension ≥ 65, we get

$$\sum_{v\in V} f^{-6}(v) = \sum_{v\in V} f^{5}(v) = 0$$

(resp. $\sum_{v\in V} f^{6} = 0$). The set of 2^{65} zero-sum input values $\{f^{-6}(v) \mid v \in V\}$ is below the data complexity limit d for LowMC-128, and only slightly above for LowMC-80.

Second, by choosing a vector space V of a particular structure as a starting point, we can add a free round in the middle. Assume that V is the direct sum of any subspace of $\mathbb{F}_2^{256-3\cdot m}$, and m trivial subspaces of $\mathbb{F}_2^3$ (i.e., each is either $\mathbb{F}_2^3$ or $\{(0,0,0)\}$). Since the bijective 3-bit S-box maps any trivial subspace of $\mathbb{F}_2^3$ to itself, applying the S-box layer to this vector space produces another vector space $V' \le \mathbb{F}_2^{256}$ (of the same form). This reduces the data complexity of the distinguisher below the data complexity limit for LowMC-80: For any V of dimension ≥ 33 of the above direct-sum format and the corresponding V',

$$\sum_{v\in V} f^{-5}(v) = \sum_{v'\in V'=f_S(V)} (f^5 \circ f_K \circ f_L)(v') = 0,$$

so the set $\{f^{-5}(v) \mid v \in V\}$ is a zero-sum distinguisher for LowMC-80 with known key, with a data complexity of 2^{33}. The attack strategy is illustrated in Fig. 2.

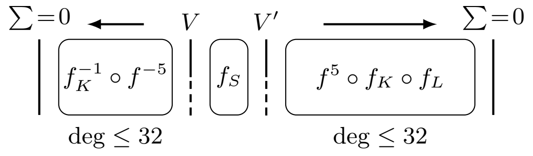

Fig. 2: Zero-sum distinguisher for $r = 11$ rounds of LowMC-80 with a data complexity of 2^{33} message blocks.

Third, we can take advantage of the special structure of the S-box layer, which only applies S-boxes to part of the state, while the rest is left unchanged by an identity map. In the inverse round f^{-1}, the linear layer is applied before the S-box layer. This means that (the leftmost / most significant) $\ell = n - 3 \cdot m$ bits of the output of each inverse round f^{-1} only depend linearly on the round input bits ($\ell = 109$ bits for LowMC-80, $\ell = 67$ bits for LowMC-128). If we add such an inverse round to a function, the degree of ℓ output bits is bounded by the same limit as the original function output, so the inverse round (corresponding to a round in the beginning of the cipher) is essentially "for free" on these bits. A similar idea can be used to also add a free forward round at the end. To compensate for the additional linear layer after the last S-box layer, however, we need to generalize the partial zero-sum property further: Instead of a zero-sum property on some of the output bits, we get a zero-sum property on some (linearly independent) linear combinations of the cipher's output bits. Note that the final linear transformation $f_K \circ f_L$ can be swapped to $f_L \circ f_{K'}$ with some equivalent key K'. Since the addition of K' does not change the partial zero-sum property, the linear combination of output bits that sums to zero does not depend on the key. Since ℓ is relatively large ($\ell > k$ for LowMC-80), even a zero-sum distinguisher only for ℓ bits of the input and (linearly combined) output gives us a detectable distinguishing property. With the above approach, we get ℓ-bit partial zero-sums with 2 more rounds for free, so the dimension of V can be reduced to 17 (for LowMC-80) and 33 (for LowMC-128) to cover the recommended full round sizes, as illustrated in Fig. 3. Table 3 summarizes the best attacks for the known-key setting.

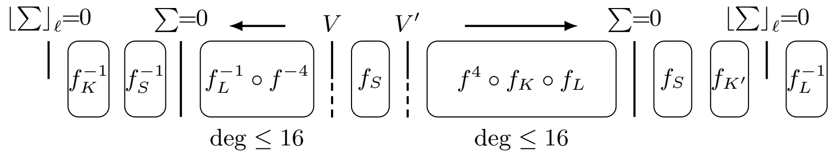

Fig. 3: Partial 109-bit zero-sum distinguisher for $r = 11$ rounds of LowMC-80 with a data complexity of 2^{17} message blocks.

Table 3: Distinguishers for LowMC in the known-key setting.

	Zero-sum		Partial zero-sum		
Target	Rounds	Complexity	Zero-sum bit size	Rounds	Complexity
LowMC-80	11/11	2^{33}	109/256	11/11	2^{17}
LowMC-128	12/12	2^{65}	67/256	12/12	2^{33}

4 Higher-order attacks in the secret-key setting

In this section, we investigate the applicability of higher-order attacks in the secret-key setting. While the basic observations about the degree of the round functions and the zero-sum property still hold in this setting, it is no longer easily possible to define an initial structure in the middle of the cipher as in Sect. 3.2 and compute in both directions. Attack goals in the secret-key setting include output distinguishers and key recovery. For simplicity, we demonstrate our attacks for the LowMC variant LowMC-80 ("PRESENT-like security"), and only briefly discuss the applicability and complexity for LowMC-128 and more general LowMC configurations at the end of the paper.

4.1 Basic zero-sum key recovery

Based on the basic observations from Sect. 3.1, an attacker can trivially distinguish the output of up to 3, 4, or 5 rounds of LowMC-80 from random values with a data complexity of 2^9, 2^{17}, or 2^{33}, respectively (all below the logarithmic data complexity of $d = 64$): He requests the ciphertexts for all values of an (affine) vector space V of the given size, and verifies the zero-sum property of the corresponding outputs.

By choosing a vector space of the same direct-sum form as in Sect. 3.2 for V, we can add an additional free round in the beginning: The output values after applying the initial key whitening f_K and the first S-box layer f_S to all vectors in V is another (affine) vector space V' of the same direct-sum form, and the remaining added round functions f_L, f_K, both of degree ≤ 1, can be added to the following 3, 4, or 5 rounds for free, without increasing the necessary dimension of V.

Additionally, we can add another final round without increasing the data complexity, by turning the distinguishing attack into a key recovery attack: To recover the key from $r \in \{5, 6, 7\}$ rounds of LowMC-80, the attacker chooses a vector space V of the previous direct-sum form with $|V| = 2^{2^{r-2}+1}$ elements as inputs (corresponding to 2^9, 2^{17}, or 2^{33} plaintexts, respectively). As previously, the corresponding outputs after $r-1$ rounds will sum to zero. This property can be used to recover the final round key K_r in 3-bit chunks, which in turn allows to easily recover the original key K. Let $S_i^{(r-1)}$ denote the state after $r-1$ rounds applied to input $P_i \in V$, and $S_i^{(r)} = C_i$ the corresponding ciphertext obtained by the attacker, so

$$C_i = (f_K^{(r)} \circ f_L^{(r)} \circ f_S)(S_i^{(r-1)}) = (f_L^{(r)} \circ f_S)(S_i^{(r-1)}) + K_r.$$

Since the key addition $f_K^{(r)}$ and linear layer $f_L^{(r)}$ can be swapped (replacing the original K_r with a transformed K'_r), the zero-sum property translates to

$$\sum_i S_i^{(r-1)} = \sum_i f_S^{-1} \left(K'_r + {f_L^{(r)}}^{-1}(C_i) \right) = 0.$$

For the 109 bits of the identity part of f_S, this property holds irrespective of the value of the corresponding bits of K'_r. For each of the $m = 49$ S-boxes, however, the property can be checked independently for each possible value of the corresponding 3 bits of K'_r. Since we expect to require about 80 bits of K'_r to recover K_r and consequently K, we guess the keys for 27 S-boxes, which leads to an overall computational complexity of $2^{2^{r-2}+1}$ queries plus a comparably negligible $2^{2^{r-2}+1} \cdot 8 \cdot 27$ xor operations.

The approach is illustrated in Fig. 4. The complexity is 2^9 encryption queries for 5 rounds, 2^{17} for 6 rounds, and 2^{33} for 7 rounds. Note that this attack is relatively generic for SPNs with degree 2, without using any particular properties of LowMC (in particular, independent of the number m of S-boxes per layer).

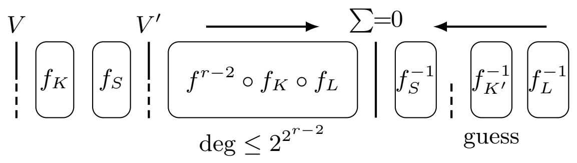

Fig. 4: Key recovery attack on $r = 5$, 6, or 7 rounds of LowMC, with a data complexity of $2^{2^{r-2}+1} = 2^9$, 2^{17}, or 2^{33} plaintexts, respectively.

4.2 Adding rounds: Initial key-guessing

In addition to the generic attack strategy applied so far, we can take advantage of the special structure of LowMC's substitution layer. So far, we were able to add one free initial round by using a direct-sum construction to obtain a vector space again after applying one round f. We will now try to define an initial structure that yields a vector space after 2 rounds f^2.

Consider the first 2 rounds $f^2 = f_K \circ f_L \circ f_S \circ f_K \circ f_L \circ f_S$ of LowMC-80. The final linear components $f_K \circ f_L$ pose no constraints. Let $V = \{(0, \ldots, 0)\} \times \mathbb{F}_2^{109} \leq \mathbb{F}_2^{256}$ be the vector space of elements that are zero except for the bits processed by the identity part of the substitution layer f_S. Thus, for any subspace $W \leq V$, $f_S(W) = W$. We want to find a suitable subspace $W \leq V$ that will yield another vector space W'' even after applying the second substitution layer f_S. The input space W' to the second f_S is an affine transformation of W, $W' = (f_K \circ f_L)(W)$. Ideally, this space W' would be of the same structure as W: Zero (or constant) in all bits except the identity part. However, this requirement would impose 147 linear constraints (one for each S-box input bit) on the 109-dimensional space V, so we cannot expect any suitable (nontrivial) solution space W.

However, it is not actually necessary to require all S-box input bits to be zero or constant. Consider an input set where two of the input bits to an S-box are fixed, but one bit is toggled. The two different input values will produce two different output values, i.e., toggling one input bit will toggle some of the output bits. This is essentially "linear" behaviour, so the S-box is linear with respect to one input bit. Thus, an input space that allows up to one bit per S-box to be toggled (non-constant) will produce another affine space as the output of the S-box layer. Note that due to the key addition f_K just before the S-box layer, we cannot know or control the constant values of the constant bits, and thus do not know the linear behaviour of the S-box. However, we can be sure that if W is such that after the linear layer f_L, two of the three input bits to each S-box of f_S are constant, then the input space W will be mapped to another vector space by 2 rounds f^2 of LowMC-80. This corresponds to $2 \cdot 49 = 98$ linear constraints on the 109-dimensional V, so we will get a solution space W of dimension at least $109 - 98 = 11$. W can be precomputed and depends only on the matrix of the linear layer of the first round of LowMC-80. The dimension of 11 is sufficient for our previous attack on 5 rounds, which required an input space of 2^9 elements, and allows us to extend this attack to 6 rounds at no additional cost. The attack is illustrated in Fig. 5.

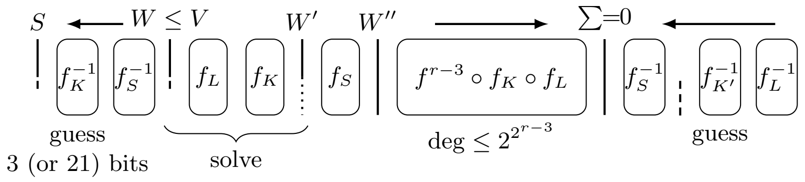

Fig. 5: Key recovery attack on $r = 6$ rounds of LowMC-80, with complexity 2^9.

Unfortunately, the dimension is too small to extend the previous attacks on 6 and 7 rounds, which required input spaces of 2^{17} and 2^{33} elements, respectively. To increase the dimension of W to 17 or higher, we need to allow for more freedom either in the first or in the second substitution layer f_S. First consider the first substitution layer. If we want to choose our inputs so as to ensure a specific vector space structure after the first substitution layer, we can achieve this trivially if the target vectors are non-constant only in the identity part. If we want specific values at the output of an S-box, we need to guess the corresponding 3 bits of the first whitening key, which is added right before the substitution layer. By guessing these 3 bits, we can increase the dimension of W by 3. Note that if we "activate" an S-box this way, the required message input set S to produce W is no longer necessarily a vector space. In particular, its elements no longer necessarily sum to zero. However, this is not required for our key recovery attacks, so the loss of the input zero-sum property is no problem. To apply the technique to extend the previous 6-round attack of Sect. 4.1 and increase W to dimension 17, we need to activate 2 S-boxes and thus guess 6 key bits. This increases the attack complexity for the extended 7-round attack to $2^{17} \cdot 2^6 = 2^{23}$. For the previous 7-round attack of Sect. 4.1, we need dimension 33 and thus need to activate 8 S-boxes with 24 guessed key bits, and the complexity for the extended 8 rounds is $2^{33} \cdot 2^{24} = 2^{57}$.

We can, however, also consider additional freedom in the second substitution layer in order to decrease the necessary number of activated S-boxes in the first layer. We previously chose a fixed bit per S-box of the second f_S which was allowed to toggle, while the other two bits needed to remain constant. But this is not actually necessary: we have the freedom to choose any of the 3 bit positions of each S-box as the toggle-bit, so we have a total of $3^{49} \approx 2^{77.7}$ options to choose the 98 (out of the total 147) constraints imposed by the second layer. The 147 available constraints are specified by the (roughly uniformly randomly generated) rows of the linear layer matrix. In addition, we have the freedom to select the activated S-boxes of the first layer. For each option, we have a very small chance that the selection of 98 constraints is redundant (with respect to the $109 + 3s$-dimensional V, if we guess s S-box keys in the first substitution layer), and the remaining solution space has a dimension larger than $11 + 3s$.

Consider again the 8-round attack, with its required input space of $2^{33} = 2^{11} \cdot 2^{22}$ elements. To increase the dimension by 22, we had to activate $s = 8$ S-boxes. We only needed 1 bit of freedom from the last of the 8 S-boxes, but still had to guess all the corresponding 3 key bits. There is a reasonable chance that if we activate only $s = 7$ S-boxes (and start with V of dimension $109 + 7 \cdot 3 = 130$) and add the 98 constraints of the second layer, the remaining solution space has the required dimension 33 instead of the expected $130 - 98 = 32$. This is equivalent to the event that 98 randomly selected vectors from $\mathbb{F}_2^{130}$ span a subspace of dimension 97, or that a randomly selected 130×98 matrix over $\mathbb{F}_2$ has rank 97. The probability of picking a rank-r matrix uniformly random from $\mathbb{F}_2^{n \times m}$, $n \geq m$ [14] is given by

$$P(n, m, r) = \frac{\prod_{i=0}^{r-1}(2^m - 2^i) \cdot \prod_{i=0}^{r-1}(2^n - 2^i)}{\prod_{i=0}^{r-1}(2^r - 2^i) \cdot 2^{n \cdot m}},$$

so our success chance for one selection of constraints is about

$$P(130, 98, 97) \approx 2^{-32.0}.$$

Even though the available selections of constraints are not independent, we verified experimentally that the measured distribution of the rank of random selections closely matches the theoretic expectations. Thus, it is reasonable to expect that a suitable selection exists among the available choices, and that it can be efficiently found (e.g., after trying about 2^{32} random selections). Since the selection depends only on the corresponding matrix of the linear layer, it can be precomputed in advance.

The same strategy can also be applied to the 7-round attack, although at a higher precomputation cost: We activate $s = 1$ instead of $s = 2$ S-boxes in the first layer, and compensate by reducing the rank of the 98 selected constraint vectors by 3 to 95. The success chance for one selection is about

$$P(112, 98, 95) \approx 2^{-49.4}.$$

The modified attack for 7 and 8 rounds is illustrated in Fig. 6. The final attack complexities are 2^9 for 6 rounds, 2^{20} (with $2^{49.4}$ precomputation) or 2^{23} (without precomputation) for 7 rounds, and 2^{54} for 8 rounds.

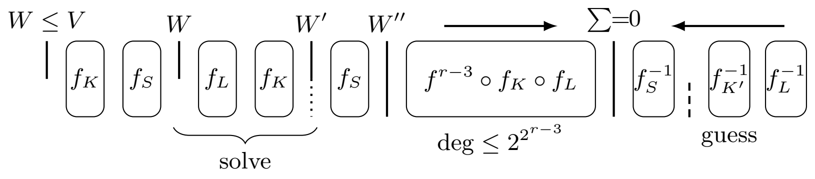

Fig. 6: Key recovery attack on $r = 7$ (or 8) rounds of LowMC-80, with complexities 2^{20} and 2^{54}, respectively.

4.3 Adding rounds: Final key-guessing

We can not only guess the round keys of the first round to increase the attacked round number, but also the last round keys. We want to combine a linear mask for the linear layer of the second-to-last round with key guesses for some S-boxes of the last round. This combination will allow us to derive 1 bit of key information per input set, and can be repeated to learn more.

For an attack on r rounds, assume we have constructed a zero-sum attack for $r-2$ rounds, that is, we can generate sets of inputs such that their corresponding outputs after $r-2$ rounds sum to zero. If we denote the intermediate states and rearrange the key addition layer as in Sect. 4.1, we get

$$C_i = S_i^{(r)} = \left(f_L^{(r)} \circ f_{K'}^{(r)} \circ f_S \circ f_L^{(r-1)} \circ f_{K'}^{(r-1)} \circ f_S\right)\left(S_i^{(r-2)}\right).$$

Since $\sum_i S_i^{(r-2)} = 0$, we also get the partial zero-sum

$$\left\lfloor \sum_i \left(f_{K'}^{(r-1)} \circ f_S\right)\left(S_i^{(r-2)}\right)\right\rfloor_{109} = 0,$$

where $\lfloor x \rfloor_\ell$ is the value x truncated to the most significant ℓ bits, i.e., the identity part of the S-box layer. Now let

$$x_i = \left(f_{K'}^{(r-1)} \circ f_S\right)\left(S_i^{(r-2)}\right), \qquad y_i = \left(f_S^{-1} \circ f_{K'}^{(r)^{-1}} \circ f_L^{(r)^{-1}}\right)(C_i),$$

so x_i and y_i are the states right before and after the linear layer of the second-to-last round,

$$y_i = f_L^{(r-1)}(x_i).$$

Now assume (a, b) is a pair of consistent linear input- and output masks for $f_L^{(r-1)}$, that is, for all $x \in \mathbb{F}_2^{256}$,

$$\langle a, x \rangle = \langle b, f_L^{(r-1)}(x) \rangle.$$

We will call the mask pair (a, b) suitable if a is zero on its 147 least significant bits (i.e., all bits except the identity part of f_S), and b is zero on most of its 147 least significant bits. We refer to the S-boxes where b is non-zero on one of the corresponding 3 input bits as active.

We target mask pairs (a, b) with at most 6 active S-boxes. For a random matrix, the probability that an input mask a is mapped to an output mask b in which only 6 of 49 S-boxes are active is, by the inclusion-exclusion principle,

$$P[\leq 6 \text{ S-boxes active}] = \sum_{i=0}^{6} (-1)^i \cdot \binom{6}{i} \cdot \binom{49}{i} \cdot 2^{-3\cdot(49-i)} \approx 2^{-105.4}.$$

Since we have a total of 2^{109} possible input masks a available, we can expect a suitable mask pair to exist. In practical experiments, we were able to find suitable masks with 6 or even fewer active S-boxes in reasonable time.

Observe that if (a, b) is a suitable mask pair, then

$$\sum_i \langle b, y_i \rangle = \sum_i \left\langle b, f_L^{(r-1)}(x_i) \right\rangle = \sum_i \langle a, x_i \rangle = \left\langle a, \sum_i x_i \right\rangle = 0,$$

since a only selects from the 109 most significant bits, and the x_i have the partial zero-sum property $\lfloor \sum_i x_i \rfloor_{109} = 0$. This modified zero-sum property of the y_i depends only on the last-round key bits (of the equivalent key K') added to the active S-boxes, i.e., for 6 active S-boxes, on 18 key bits. The other key bits are either not selected by b (inactive S-boxes), or cancel out during summation (identity part). The probability of the 1-bit property to hold for a random key guess is $\frac{1}{2}$, so applying the attack to one zero-sum input set will eliminate half of the key guesses for the 18 key bits, or win 1 bit of key information. By repeating the attack for 18 input sets S (e.g., by adding 18 different constants to the original input set), we expect to recover all 18 round key bits. The attack is illustrated in Fig. 7.

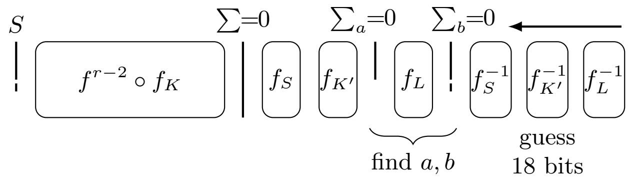

Fig. 7: Key recovery attack on r = 7, 8, or 9 rounds of LowMC-80, with 1-bit sums $\sum_a = \sum_i \langle a, x_i \rangle$ and $\sum_b = \sum_i \langle b, y_i \rangle$ (details of $f^{r-2} \circ f_K$ as in Fig. 6).

To learn more key bits, we need to find more linear mask pairs (a, b), with different active S-boxes. Since the previously active S-boxes with previously recovered key bits can now be active for free, finding such masks becomes easier. In addition, we can re-use the same ciphertexts for different masks, so the data complexity does not increase. In summary, after precomputing suitable mask pairs, this attack described so far allows to recover the complete key for r instead of $r-1$ rounds at an additional cost factor of $18 \approx 2^{4.2}$ data complexity and about $2^{18} \cdot 80 \approx 2^{24.3}$ computational complexity.

However, the computational complexity can be further reduced by optimizing the repeated evaluation of the modified zero-sum check. Instead of summing all inputs for each of the 2^{18} key guesses, we can precompute partial bit sums, and only combine those to compute the final sum for each of the 2^{18} key candidates. The idea is to decompose the target sum into its S-box-wise components as

$$\sum_i \langle b, y_i \rangle = \sum_i \left\langle \sum_{s=0}^{49} b_s, y_i \right\rangle = \sum_{s=0}^{49} \sum_i \langle b_s, y_i \rangle ,$$

where b_s equals b on the 3 bit positions corresponding to S-box s, $1 \leq s \leq 49$ (or the 109 bits of the identity part for $s = 0$), and is zero otherwise. Then $\sum_i \langle b_s, y_i \rangle$ depends only on the 3 round key bits corresponding to S-box s (and 3 bits of $f_L^{-1}(C_i)$, see the definition of y_i), and can be precomputed in a first phase for all 2^3 possible values of these key bits, for each active S-box s. Then, in the second phase, to determine the test bit for each of the 2^{18} key candidates, it suffices to sum the 6 corresponding partial sums (of the active S-boxes). Considering that each linear layer alone needs about 2^{16} xor operations, the complexity of both phases phase is significantly smaller compared to the computational effort of generating all the required ciphertexts C_i. This step can be repeated 4 times with different mask pairs (a, b) to recover about $4 \cdot 18 = 72$ key bits; the remaining bits can easily be determined by brute force testing.

With this improvement, the computational complexity overhead factor incurred by this approach over the attack on $r-1$ rounds is dominated by the data complexity increase by a factor of about $2^{4.2}$. Based on the attacks of Sect. 4.2, we get full key recovery for 7 rounds with 2^{14} complexity, for 8 rounds with 2^{28}, and for 9 rounds with 2^{59}. We summarize all achieved attack complexities of Sect. 4.1, 4.2 and 4.3 in Table 4.

5 Application to other parameter sets

Besides the recommended versions LowMC-80 and LowMC-128, the designers also propose several alternative parameter sets for the 80-bit and 128-bit security level. For AES-like security, the design document discusses the performance of LowMC-128^{256,63} ($r = 12$ rounds, main variant) and LowMC-128^{512,86} ($r = 11$ or 12 rounds), all with data complexity limit $d = 128$; for PRESENT-like security, LowMC-80^{256,49} ($r = 11$ rounds, main variant, or $r = 10$) and LowMC-80^{128,34} ($r = 11$ rounds), all with data complexity limit $d = 64$.

Table 4: Key-recovery attacks for LowMC-80: number of rounds with computational and data complexity (all below logarithmic data complexity limit $d = 64$).

	Basic (4.1)		Initial key guess (4.2)		Final key guess (4.3)	
Cube degree	Rounds	Compl.	Rounds	Compl.	Rounds	Compl.
8 (f^3)	5/11	2^9	6/11	2^9	7/11	2^{14}
16 (f^4)	6/11	2^{17}	7/11	2^{23}	8/11	2^{28}
32 (f^5)	7/11	2^{33}	8/11	2^{54}	9/11	2^{59}

For LowMC-$128^{256,63}$, the attacks of Sect. 4.1 apply for the same number of rounds, with the same complexity. Furthermore, due to the increased logarithmic data complexity limit, an additional round can be added here (for a total of 8 rounds), and the data complexity increased accordingly. However, the size of the identity part, $\ell = 67$, is too small to append rounds with initial-key-guessing as in Sect. 4.2: the necessary number of about $3 \cdot 40$ guessed S-box key bits becomes prohibitive. Final-key-guessing as in Sect. 4.3, on the other hand, is applicable in a similar way. Again, the smaller identity part increases the complexity: instead of masks b with 6 active S-boxes, about 24 active S-boxes are necessary for a reasonably high probability. If the correct $3 \cdot 24$-bit subkey is recovered as described in Sect. 4.3, the computational complexity is about 2^{72} (for up to 9 rounds). However, it is possible to optimize this step at the cost of a slightly higher data complexity. For details, we refer to the full version of this paper.

For LowMC-$128^{512,86}$, on the other hand, the size of the identity part $\ell = 254$ is almost as large as the S-box part of $3 \cdot m = 258$ bits. This allows the application of initial-key-guessing for free, and 1 active S-box is expected to be sufficient for final-key-guessing. Additionally, due to the higher logarithmic data complexity limit of $d = 128$, the core cube degree can be increased to 64 (f^6) to add another round, for a total of 10 attacked rounds (out of 11 resp. 12).

For LowMC-$80^{128,34}$, $\ell = 26$, so the same problems as for LowMC-$128^{256,63}$ apply. For the final-key-guessing, about 14 active S-boxes would be required to find suitable a, b, to attack a total of 8 rounds.

We want to stress that all described attacks are generic for the design of LowMC, without requiring specific instances of the linear layer f_L or the key schedule matrices. For specific "weak" choices of the random matrices, it is likely that attacks on more rounds are feasible.

References

1. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. LNCS, vol. 9056, pp. 430–454. Springer (2015)
2. Aumasson, J., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In: Dunkelman, O. (ed.) Fast Software Encryption – FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer (2009)

3. Bar-On, A., Dinur, I., Dunkelman, O., Lallemand, V., Keller, N., Tsaban, B.: Cryptanalysis of SP networks with partial non-linear layers. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 315–342. Springer (2015)
4. Boura, C., Canteaut, A.: Zero-sum distinguishers for iterated permutations and application to Keccak-f and Hamsi-256. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) Selected Areas in Cryptography – SAC 2010. LNCS, vol. 6544, pp. 1–17. Springer (2010)
5. Boura, C., Canteaut, A., De Cannière, C.: Higher-order differential properties of Keccak and Luffa. In: Joux, A. (ed.) Fast Software Encryption – FSE 2011. LNCS, vol. 6733, pp. 252–269. Springer (2011)
6. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) Fast Software Encryption – FSE '97. LNCS, vol. 1267, pp. 149–165. Springer (1997)
7. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) Advances in Cryptology – EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer (2009)
8. Guo, J., Nikolic, I., Peyrin, T., Wang, L.: Cryptanalysis of Zorro. IACR Cryptology ePrint Archive 2013, 713 (2013), `http://eprint.iacr.org/2013/713`
9. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain family of stream ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 179–190. Springer (2008)
10. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) Advances in Cryptology – ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer (2007)
11. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption – FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer (2002)
12. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello Jr., D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography: Two Sides of One Tapestry. pp. 227–233. Kluwer Academic Publishers (1994)
13. Rasoolzadeh, S., Ahmadian, Z., Salmasizadeh, M., Aref, M.R.: Total break of Zorro using linear and differential attacks. IACR Cryptology ePrint Archive 2014, 220 (2014), `http://eprint.iacr.org/2014/220`
14. van Lint, J.H., Wilson, R.M.: A Course in Combinatorics. Cambridge University Press (2001)
15. Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack. IACR Cryptology ePrint Archive 2007, 413 (2007), `http://eprint.iacr.org/2007/413`
16. Wang, Y., Wu, W., Guo, Z., Yu, X.: Differential cryptanalysis and linear distinguisher of full-round Zorro. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) Applied Cryptography and Network Security – ACNS 2014. LNCS, vol. 8479, pp. 308–323. Springer (2014)
17. Z'aba, M.R., Raddum, H., Henricksen, M., Dawson, E.: Bit-pattern based integral attack. In: Nyberg, K. (ed.) Fast Software Encryption – FSE 2008. LNCS, vol. 5086, pp. 363–381. Springer (2008)