The MALICIOUS Framework: Embedding Backdoors into Tweakable Block Ciphers

Thomas Peyrin(✉) and Haoyang Wang(✉)

School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore
thomas.peyrin@ntu.edu.sg, wang1153@e.ntu.edu.sg

Abstract. Inserting backdoors in encryption algorithms has long seemed like a very interesting, yet difficult problem. Most attempts have been unsuccessful for symmetric-key primitives so far and it remains an open problem how to build such ciphers.

In this work, we propose the MALICIOUS framework, a new method to build tweakable block ciphers that have backdoors hidden which allows to retrieve the secret key. Our backdoor is differential in nature: a specific related-tweak differential path with high probability is hidden during the design phase of the cipher. We explain how any entity knowing the backdoor can practically recover the secret key of a user and we also argue why even knowing the presence of the backdoor and the workings of the cipher will not permit to retrieve the backdoor for an external user. We analyze the security of our construction in the classical black-box model and we show that retrieving the backdoor (the hidden high-probability differential path) is very difficult.

We instantiate our framework by proposing the LowMC-M construction, a new family of tweakable block ciphers based on instances of the LowMC cipher, which allow such backdoor embedding. Generating LowMC-M instances is trivial and the LowMC-M family has basically the same efficiency as the LowMC instances it is based on.

Keywords: Tweakable block cipher · Backdoor · Differential cryptanalysis · LowMC-M

1 Introduction

A backdoor in an encryption algorithm enables an entity who knows it to circumvent the security guarantees so that he can obtain the secret information more efficiently than with a generic black-box attack. There are two categories of backdoors. The first one is the backdoor implemented in a security product at the protocol or key-management level, which is generally considered in practice.

In this article, we focus on the second type: a cryptographic backdoor. A cryptographic backdoor is embedded directly during the design phase of a cryptographic primitive and renders the cipher susceptible to some dedicated cryptanalysis. Cryptographic backdoors have been extensively studied by Young and Yung, introducing the term "Kleptography" [41,44]. However, despite some interest from the academic community about this topic, there are very few publicly known backdoored primitives. A concrete example is the pseudorandom number generator Dual_EC_DBRG [8] designed by NSA, whose backdoor was revealed by Edward Snowden in 2013 and also in some research works [10,37].

Embedding backdoors into block ciphers is a challenging problem since block ciphers are deterministic and thus it is complex to exploit randomness in computations. Young and Yung have designed several backdoors in secret block ciphers [42,43,45], where it is assumed that the cipher specifications are unknown to the adversary. In this work, we will not make such assumption and we will consider the specifications of the cipher to be fully public.

A backdoor should be computationally difficult to retrieve, even if its general form is known. More concretely, the backdoor security (the cost of retrieving the backdoor) should be the same as the security generically provided by the cipher (otherwise the backdoor would naturally reduce the security of the block cipher). Besides, the backdoor should ideally lead to a practical key recovery attack, or at least reduce the brute force search cost for the adversary. For example, if a backdoor could reduce the security of AES-256 to 2^{128}, it would be a great theoretical advance, but would be unusable in practice. Last but not least, the resulting block cipher also has to be secure in the classical sense, that is, it is able to resist state-of-the-art cryptanalysis techniques.

There have been only limited works focusing on this direction and to the best of the authors' knowledge there is no such design satisfying the above requirements simultaneously. In 1997, Rijmen and Preneel proposed a special Sbox design strategy which was used to hide a high-probability linear approximation in an Sbox [35]. The knowledge of this backdoor leads to an efficient key recovery attack based on linear cryptanalysis, but only a part of the key information can be obtained. They presented concrete instantiations by applying the Sbox design to CAST and LOKI91 ciphers and claimed that the embedded backdoors are undetectable even if the general form of the backdoor is known. However, this design was broken subsequently in 1998 [39] by Wu *et al.* who found a way to easily recover the backdoor and showed that the security and practicability of the backdoor can't be guaranteed at the same time. Later in 1999, Paterson suggested that if the group generated by round functions acts imprimitively on the message group, then it is possible to create a backdoor in the cipher [31]. Built upon this mechanism, he introduced a DES-like cipher which allows an entity knowing the backdoor to retrieve the key with 2^{41} computations. However, as mentioned by the author, the backdoor is detectable and the cipher is vulnerable to differential attacks. Following on this idea, a backdoor based on partitioning cryptanalysis was studied in [5] and a concrete instance of an AES-like cipher called BEA-1 was later proposed in [6], but no explicit backdoor security was provided. One can also mention the work from Patarin and Goubin [29,30] who proposed "2R–schemes", basically Sbox-based asymmetric schemes secretly consisting of a 2-round secret Substitution-Permutation Network (SPN) but publicly represented as its corresponding algebraic equations. However, this research direction also suffered from attacks [12,40]. Two more backdoor designs [4,13] have been introduced, but neither of them provide solid proof for the backdoor security and even the security of the cipher itself is questionable. Lastly, in a different setting, a backdoored version of the SHA-1 hash function was proposed in [1], where the attacker is allowed to pre-choose the constants used in the design, so he can prepare in advance some specific collision messages for that particular instance.

Apart from these public researches, one can naturally question if there are some public block ciphers that might contain backdoors not claimed by the designers. In particular, primitives whose detailed design rationale is not provided are naturally more suspicious, especially when the ciphers have been designed by governmental agencies (as can be seen by the difficulties encountered by the NSA lightweight block ciphers SIMON and SPECK [9] to become ISO standards). For example, Perrin found a very strong algebraic structure [32] that is hidden inside the Sbox employed in both the block cipher Kuznyechik [36] and the hash function Streebog [27], both primitives being selected as Russian standards (GOST). Even though there is currently no attack based on this result, it illustrates the issue of potential backdoor in foreign encryption algorithms and more research is required to better understand the possibilities and implications of cryptographic backdoor.

We emphasize that inserting backdoors in an encryption algorithm itself is very different from inserting backdoors in an implementations, being in software or in hardware (like hardware trojans).

Our Contributions. In this paper, we propose a new method to generate backdoor encryption algorithms. We bring together tweakable block ciphers (TBC) and Extendable-Output Function (XOF) in a common framework called MALICIOUS, which enables the designer to embed backdoors into the TBC. The general representation of our construction is similar to that of the TWEAKEY framework [22], but the tweak is handled separately by a XOF and the round function has to be partially non-linear.

Our backdoor is based on differential cryptanalysis: due to the partial nonlinear layer, the designer can embed related-tweak differential characteristics with probability 1 over many rounds. In particular, the sub-tweak difference employed in an embedded differential characteristic is generated from a specific tweak pair that is chosen in advance by the designer. This malicious tweak pair **is** the backdoor, and the XOF applied in the tweak schedule is used to protect the malicious tweak pair: even knowing the high-probability related-tweak differential characteristic, it will remain computationally difficult to find a tweak pair that triggers it. More importantly, the backdoor security is ensured by the target-difference resistance ability of the chosen XOF. An attacker with the knowledge of the backdoor is able to retrieve the full key with negligible effort under the chosen-tweak scenario.

Based on the MALICIOUS framework, we also propose a concrete instantiation that we call LowMC-M. Our family of TBC LowMC-M is created based on some instances of the block cipher LowMC [2]. Compared to LowMC, our proposal LowMC-M has an additional sub-tweak addition in each round and the tweak schedule is a XOF, but the other parts of the round function and the number of rounds remain unchanged. Apart from its backdoor security that is naturally inherited from the MALICIOUS framework, we claim that its classical black-box security against state-of-the-art cryptanalysis is the same as the corresponding LowMC variants.

We believe this work is a first step in a new direction for the study of backdoors in encryption algorithms. We are confident that more exotic (based on other types of cryptanalysis techniques than plain differential cryptanalysis) and potentially more efficient instances following the MALICIOUS would be possible.

Paper Organization. In Sect. 2, we present the attacking scenario and some security notions for backdooring cryptographic primitives. In Sect. 3 the MALICIOUS framework is described and its backdoor security and design rationale are explained. We introduce a concrete instantiation of MALICIOUS (so-called LowMC-M) in Sect. 4. We then analyze LowMC-M with respect to the backdoor security and the classical black-box security in Sect. 5 and Sect. 6 respectively. Finally, we present our conclusions in Sect. 7.

2 Preliminaries

2.1 Attacking Scenario

For classical (tweakable) block ciphers, the attacking scenario considers only two entities: the *user* (or pair of users) who owns the secret key and the *attacker* who tries to break the cryptosystem, *i.e.*, to find out the secret key. For (tweakable) block ciphers with a backdoor, another entity has to be involved in the attacking scenario: the *designer*, who inserts the backdoor into the primitive. Thus, we have in total three entities: the designer (knows the backdoor, but not the secret key), the user (knows the secret key, but not the backdoor) and the attacker (neither backdoor nor key is known).

One can see that both the user and the attacker have some motivation to find out what is the backdoor. More importantly, in our model the backdoor is independent of the secret key, and therefore the user and the attacker possess the same capability in trying to uncover the backdoor (the cipher specifications are public known, so they can test the cipher with any chosen key they want). For the rest of this article, when considering the recovery of the backdoor, we will simply refer to both of them as the attacker.

2.2 Security Notions and More

We introduce below various notions regarding the security and the practicability of a backdoor:

- *Undetectability:* this security notion represents the inability for an external entity to realize the existence of the hidden backdoor.
- *Undiscoverability:* it represents the inability for an attacker to find the hidden backdoor, even if the general form of the backdoor is known.
- *Untraceability:* it states that an attack based on the backdoor should not reveal any information about the backdoor itself.
- *Practicability:* this usability notion stipulates that the backdoor is practical, in the sense that it is easy to recover the secret key once the backdoor is known.

If a cipher is publicly claimed as potentially backdoored, it will naturally increase the watchfulness of users, even if they do not know whether there is indeed backdoored or not embedded in the primitive. In this scenario, the undetectability notion models the incapacity of a user to find any hard evidence that a backdoor indeed exists.

For our proposal LowMC-M, the backdoor is claimed to be undetectable, undiscoverable and practicable, but not untraceable.

2.3 Notations

Given a bit string x, we will denote by $x[i]$ its i-th bit, counting from the least significant bit (LSB). Given two bit strings x and y, $x||y$ will represent the concatenation of x and y. Finally, we denote by k_j (respectively by t_j) the subkey (respectively sub-tweak) incorporated during the j-th round of the cipher, while k_0 and t_0 are added in as whitening material.

3 The MALICIOUS Framework

In this section, we introduce the MALICIOUS framework which allows to generate tweakable block ciphers that are embedded with hidden high-probability differential characteristics. This framework is based on partial non-linear layers for the internal state transformation and a tweak schedule based on an extendable-output function (XOF).

3.1 Block Ciphers with Partial Non-linear Layers

SPN-based block ciphers are usually designed to apply linear layers (L_i) and non-linear layers (S_i) to the entire state at every round i. In 2013, an irregular design was suggested by Gérard *et al.* [18], where the non-linear layer is only applied to a subpart of the state at each round. We consider such design with block size n bits and partial non-linear layers of size s ($< n$) bits. Assume, without loss of generality, that the non-linear layer is always applied before the linear layer at every round. Then, we can write $f_i(x) = L_i(S_i(x^{(0)})||x^{(1)})$ the round function f_i that transforms the state x at round i, the state being partitioned into two parts where the non-linear layer only operates on the part $x^{(0)}$ and not on the part $x^{(1)}$.

Such design allows efficient masking and thus can improve security against side-channel attacks. A concrete instantiation of this methodology named ZORRO was then proposed [18]. Even though ZORRO was rapidly broken [7,20,33,38], the general design strategy continued to attract interest from the research community: in 2015, another such design LowMC was proposed [2]. Its aim was to minimize the multiplicative complexity and depth of the cipher in order to have performance advantages in certain applications, including multi-party computation (MPC), fully homomorphic encryption (FHE) and zero-knowledge proofs (ZK). After a few tweaks due to security concerns, the current version v3 of LowMC remains solid after the several third party analysis [15,16,34].

Compared to a full non-linear layer, a partial non-linear layer inevitably weakens the security of a cipher. One notable property is that there will exist non-trivial differential characteristics that will not activate any Sboxes over one or more rounds of the cipher. In a single round, by setting the difference on $x^{(0)}$ to be 0, there are 2^{n-s} differences of x that do not differentially activate any Sboxes. Assuming a well designed linear layer with good mixing properties, one can still expect around 2^{n-2s} differences that will also not differentially activate any Sboxes in the second round. This reasoning can be continued until no difference survives and thus the maximal expected number of rounds that a deterministic differential characteristic can cover is $\lfloor \frac{n}{s} \rfloor$. Note that this number would of course vary depending on the specificities of the linear layers.

3.2 Tweakable Block Ciphers

The first formal treatment of tweakable block ciphers (TBC) was proposed by Liskov, Rivest and Wagner in [25,26]. The signature of a conventional block cipher can be described as $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ where an n-bit plaintext is encrypted to an n-bit ciphertext using a k-bit secret key. A tweakable block cipher accepts an additional t-bit public input called tweak, its signature thus being $E : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^n$. The introduction of a tweak input provides the ability for the user to select a permutation among a family of permutations even when the key is fixed.

Due to this extra degree of freedom that can potentially be leveraged by the attacker, designing a TBC is not straightforward. Block cipher-based TBC constructions have been studied, but comes with a non-negligible efficiency penalty. We can mention the TWEAKEY framework, a recent design strategy to build ad-hoc TBCs, that was proposed at ASIACRYPT 2014 by Jean *et al.* [22]. In this framework, the key and tweak inputs are treated equivalently in terms of design and this material is called tweakey: the tweakey input can be used as key or tweak value, which is up to the choice of the user.

Unlike the key input, the tweak does not need to be kept secret and therefore one should assume that an adversary has full control over it. Thus, besides the attack models of single-key (no difference in the key or tweak), related-key (difference in the key, but no difference in the tweak), related-tweak (no difference in the key, but difference in the tweak) and related-tweakey (difference in both the key and tweak), it is reasonable to consider the chosen-tweak model as a meaningful model in practice.

3.3 Extendable-Output Function

An extendable-output function (XOF) is a generalization of a hash function, where the output can be extended to any desired length. Similar to a hash function, it should be collision, preimage and second-preimage resistant. A XOF is a natural choice when an application requires a hash function to have non-standard digest length. Technically, it is also possible to use a XOF as a generic hash function by setting the output length fixed. Besides, it has some other applications, such as key derivation functions and stream ciphers.

Currently, there are many instances of XOF, such as SHAKE128 and SHAKE256 (defined in SHA-3 standard [17]) and the more efficient variant KangarooTwelve [11].

3.4 The MALICIOUS Construction

Motivation. Differential and linear cryptanalysis are among the most efficient and well-understood attacks against block ciphers, both in theory and in practice. Thus, it seems natural to try creating backdoors using these techniques. Yet, there have been only a few works focusing on this research direction. For example, [3] and [28] explored backdoors in hash functions based on differential cryptanalysis. As for block ciphers, to the best of our knowledge, there is only one work from 1997 [35] using linear cryptanalysis. In that paper, special Sboxes are designed to hide high-probability linear approximations, which then enable a practical linear cryptanalysis. However, this construction was easily broken by Wu *et al.* in the subsequent year [39]. The attack against this cipher shows that the higher the probability of the embedded linear approximation, the weaker the backdoor security. Consequently, the authors claimed that it is infeasible for such a cipher to build a practical backdoor while keeping acceptable backdoor security. They further noted that "*it seems that hiding differentials is more difficult than hiding linear relations*".

Even though other block ciphers embedding backdoors have been proposed [4–6,13,31], their design methodologies are usually very dedicated. On the other hand, as the topic of backdoor ciphers has not drawn much attention from the cryptography community, the backdoor security of these ciphers has not been well analyzed yet.

Considering the above facts, we introduce the MALICIOUS framework which allows to build efficient backdoors based on differential cryptanalysis. Moreover, we will show that the backdoor security can be reduced to a variation of the collision resistance notion of the XOF used in the tweak schedule.

The Construction. MALICIOUS is a framework to build a tweakable block cipher with n-bit blocksize, k-bit key and tweak of arbitrary size. It consists of three components:

– a round function f_i with partial non-linear layer, which can be expressed as $f_i(x) = L_i(S_i(x^{(0)})||x^{(1)})$,
– a tweak schedule based on a XOF,
– a key schedule.

The sub-tweak and sub-key values are XORed only to the non-linear part of the state, but are XORed to full state at the whitening stage[1]. The cipher is composed of r consecutive rounds. The framework is depicted in Fig. 1.

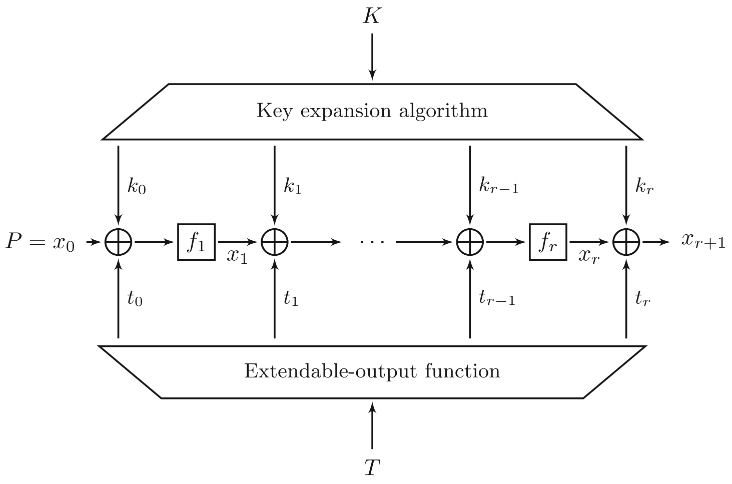

Fig. 1. The MALICIOUS framework

The backdoor introduced by MALICIOUS are related-tweak differential characteristics with probability 1 (deterministic). With the knowledge of this backdoor, a key recovery attack can be performed using various methods of differential cryptanalysis. It is to be noted that the attack is under the *chosen-tweak* model: both the designer and the attacker have complete freedom over the tweak values. This model is classical for TBC and realistic in practice.

We now describe how the backdoor can be embedded in the cipher. The core idea is that the sub-tweak difference of the backdoor chosen tweaks is used to cancel the difference of the non-linear part of the state in each round, so that the resulting differential characteristics will have no differentially active Sbox (as illustrated in Fig. 2). In Algorithm 1, we present the general steps to construct a MALICIOUS instance, in which a deterministic differential characteristic over r_0 ($\leq r$) rounds is embedded.

The key of the backdoor is the tweak pair generating these particular sub-tweak differences and the plaintext difference used in the embedded differential characteristic. We will use the prefix *malicious* to denote them. We also note that

[1] This is equivalent to a full state addition for all rounds, see Sect. 4.2 for details.

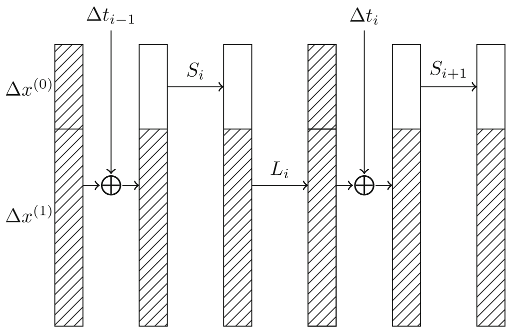

Fig. 2. Transitions of state difference in the embedded related-tweak differential characteristic. The differences of the hashed blocks can be zero or non-zero, while the differences of the white blocks are necessarily zero.

Algorithm 1: Constructing a MALICIOUS instance with an embedded deterministic differential characteristic over r_0 rounds

```
Select a XOF as the tweak schedule.
Choose uniformly at random a pair of tweak values $(T_1, T_2)$ of arbitrary length.
Compute $t_0^1||\ldots||t_r^1 \leftarrow \mathrm{XOF}(T_1)$ and $t_0^2||\ldots||t_r^2 \leftarrow \mathrm{XOF}(T_2)$.
Evaluate the differences $\Delta t_i = t_i^1 \oplus t_i^2$ for all $i \in [0, \ldots, r]$.
Randomly select a plaintext difference $\Delta P = \Delta x_0$ for the linear part $x_0^{(1)}$ and
set $\Delta x_0^{(0)} = \Delta t_0^{(0)}$.
for i from 1 to r do
    Determine a round function $f_i$ with partial non-linear layers such that:
    if $i < r_0$ then
        Given the input difference $(\Delta x_{i-1} \oplus \Delta t_{i-1})$, the output difference after
        $f_i$ has to satisfy $\Delta x_i^{(0)} = \Delta t_i$.
    end
end
Output the cipher description and the $r_0$-round related-tweak differential
characteristic that is embedded into it (with related tweaks $T_1$ and $T_2$).
```

it is possible to embed multiple differential characteristics simultaneously. Then, the key recovery complexity will depend on the number of embedded differential characteristics and the cryptanalysis method.

We emphasize that the framework only focuses on the requirements of the cipher to embed a backdoor. However, a concrete instantiation would also have to take into account many other design principles so that the cipher could resist all state-of-the-art cryptanalysis as well as the attack against the backdoor described in the following section.

3.5 The Backdoor Security

In this section, we will evaluate two particular aspects of the backdoor security: (1) the complexity for the attacker to find the embedded differential characteristics, (2) whether additional backdoors exist in the resulting primitives, and if so, what is the complexity to find them.

Firstly, we will discuss the relation between the malicious tweak pair and its corresponding plaintext difference. We consider in this article that the number of rounds for the embedded differential characteristic is publicly known. On the one hand, if the malicious tweak pair is known to the attacker, then the corresponding sub-tweak differences can of course be computed. From these sub-tweak differences, he can obtain partial information about the state differences expected during the differential characteristic. Note that the embedded differential characteristic being deterministic indicates that the transformations of state differences are linear. Hence, by reversing the linear transformations, the malicious plaintext difference can eventually be recovered. That is, the leakage of the malicious tweak pair reveals the malicious plaintext difference.

On the other hand, if the malicious plaintext difference is known to the attacker, he can compute its transformation through the linear layer and obtain the required value for the sub-tweak difference such that it cancels the non-linear layer difference (since the sub-tweak is only XORed to the non-linear part, there is only one such candidate), and continue this process in the following rounds. Eventually, the embedded differential characteristic will be revealed. However, it remains difficult to recover the actual malicious tweak pair due to the XOF-based tweak schedule: given the embedded related-tweak differential characteristic, finding a tweak pair that leads to it through the XOF will be difficult. We define this new security notion as *target-difference resistance*:

Definition 1 (Target-difference resistance). *A hash function H is target-difference resistant if it is hard to find two inputs x and y such that $H(x) \oplus H(y) = \Delta$, where Δ is a given non-zero constant.*

To better understand target-difference resistance, we introduce the limited-birthday problem, which was first proposed in [19]:

Definition 2 (The limited-birthday problem [21]). *Let H be an n-bit output hash function that can be randomized by some input (IV or tweak or etc.) and that processes any input message of fixed size m bits, where $m > n$. Let IN be a set of admissible input differences and OUT be a set of admissible output differences, with the property that IN and OUT are closed sets with respect to $\oplus$ operation. Then, for the limited-birthday problem, the goal of the adversary is to generate a message pair (x, y) such that $x \oplus y \in IN$ and $H(x) \oplus H(y) \in OUT$ for a randomly chosen instance of H.*

Let 2^I and 2^O denote the sizes of IN and OUT respectively. The lower bound on the time complexity to find a solution for the limited-birthday problem is

$\max(2^{\frac{n-O+1}{2}}, 2^{n-I-O+1})$[2]. If I is small, the complexity is $2^{n-I-O+1}$. However, even if I is very big, the complexity cannot be below $2^{\frac{n-O+1}{2}}$.

Target-difference resistance can be seen as a special case of the limited-birthday problem (as well as a generalisation of the classical collision resistance) where OUT is limited to a single value ($2^O = 1$) and IN is the full input space. Therefore, target-difference resistance has the same generic complexity as the classical collision resistance notion, that is the birthday bound $O(2^{n/2})$.

More generally, instead of the exact malicious tweak pair, the attacker could try to find another tweak pair whose sub-tweak differences are also the desired ones for the embedded differential characteristic. Yet, its complexity is still covered by the expected target-difference resistance of the XOF.

The above attack can possibly be applied to other plaintext differences. According to the construction of the MALICIOUS framework, the size of the input (tweak) to the XOF can be arbitrary long and thus any output of the XOF can potentially be obtained. For instance, if SHAKE128 is used as XOF, it can produce at most 2^b output streams (b being the state size between absorbing and squeezing phases in the sponge construction). Hence the number of possible sub-tweaks values is bounded by 2^b, no matter how many rounds it covers, and the number of sub-tweak differences is accordingly bounded by a greater value N ($\geq 2^b$). Thus, given a random plaintext difference and a certain number of rounds, if the size of the required sub-tweak differences for the deterministic related-tweak differential characteristic does not exceed $\log N$, then there will be a tweak pair matching the differential characteristic. We summarize this finding as follows:

Property 1. In addition to the embedded differential characteristics, there might exist other deterministic differential characteristics that would threaten the cipher security.

Consequently, we have to evaluate the security of the cipher with respect to all the potential deterministic differential characteristics, not only the planned ones. We consider a MALICIOUS instance that has a key size of 128 bits and employs SHAKE128 as tweak schedule. The security strength of SHAKE128 against collision attack is $\min(l/2, 128)$ bits, where l is the output length (or the length of the colliding part). In order to recover an r_0-round deterministic differential characteristic, the attacker has to find a tweak pair whose sub-tweak differences are the desired ones. The total size of these sub-tweak differences is $n + s \cdot (r_0 - 1)$ bits and thus the generic attack complexity is $2^{\min((n+s\cdot(r_0-1))/2, 128)}$, which becomes 2^{128} when $(n + s \cdot (r_0 - 1))/2 \geq 128$. The analysis is similar for the case where the key size is 256 bits and SHAKE256 is employed. We define r' to represent the value of r_0 that turns this inequality into an equality:

$$(n + s \cdot (r' - 1))/2 = k \tag{1}$$

All the deterministic related-tweak differential characteristics smaller than r' rounds can be recovered with a complexity smaller than the actual key size.

[2] The success probability here is about 0.63.

Therefore, in order to prevent these differential characteristics to weaken the cipher, r' must be taken into consideration when determining the number of rounds of the MALICIOUS instance. Actually, these related-tweak differential characteristics will decay exponentially in the remaining rounds as the corresponding sub-tweak differences are basically random.

3.6 Rationale Underlying the MALICIOUS Construction

When designing a backdoor for block ciphers, the first question that comes into mind is probably *what type of backdoor should be used?* While some existing backdoor designs directly insert a backdoor inside Sboxes or some other parts of the round function, we found out that the additional input tweak capability of a tweakable block cipher could be a perfect carrier of the backdoor. Suppose that a tweakable block cipher has a special property only when it is initiated with very specific tweak values, while it performs normally for all the other tweak values, then this property could be used as a backdoor. Moreover, if the tweak size is large enough, finding these special tweak values could be as hard as finding the secret key in the ideal case. One straightforward example of the special property is to build related-tweak differential characteristics using these tweaks. In the following, we provide more in-depth explanations on the design choices in MALICIOUS.

Components Rationale. When instantiating the MALICIOUS framework, some (security) notions have to be taken into account. The first and most important one is the undiscoverability: an entity who does not know the backdoor should not have increased chances to break the cipher. This requires that the backdoor security has to be as high as the cipher security. Thus, the MALICIOUS framework should provide a valid and solid security evaluation for the backdoor.

Another important notion is the practicability of the backdoor, and we will aim to make it as efficient as possible.

We detail in the following how the components of the MALICIOUS framework do follow these principles.

Tweak Schedule Based on XOF. As the malicious tweak values are the backdoor, the main task of the tweak schedule is to protect the malicious tweaks. According to the security analysis from Sect. 3.5, the backdoor security relies on the target-difference problem, where the attacker tries to find a tweak pair whose sub-tweak differences are the desired ones. This notion is simply a variation of the classical collision resistance for a hash function, so we expect a good cryptographic hash function to naturally provide this resistance.

Since MALICIOUS is a generalized framework, the total number of rounds will vary according to the different instantiations, so does the length of the sub-tweaks. Hence, the output length of the tweak schedule is expected to be flexible. Besides, if the tweak schedule was designed specifically for each MALICIOUS instantiation, it will render the backdoor evaluation much more difficult. Thus, for sake of simplicity of the analysis, it seems a better idea to make the tweak schedule uniform in the framework.

For all these reasons, a XOF seemed to be the best choice for our tweak schedule. The security of actual XOF functions such as SHAKE128 or SHAKE256 is rather well-analyzed and it can provide many choices in terms of security level.

Partial Non-linear Layers. The probability of a differential characteristic is determined by the number of differentially active Sboxes. Hence, in order to embed an efficient backdoor based on a differential characteristic, the best case is that the differential characteristic activates no Sbox at all. This is obviously very unlikely to happen in the MALICIOUS framework if the round functions are fully non-linear layers. Indeed, unless the related-key model is considered, a non-zero difference inserted in the plaintext would have to be cancelled by the first sub-tweak difference. However, when inserting differences in the tweak input, as the sub-tweak differences produced by the XOF will be random, they will force many active Sboxes in the subsequent rounds. Thus, it is unlikely for the MALICIOUS framework to be able to embed a deterministic related-tweak differential characteristic that covers more than a few rounds if full non-linear layers are utilized. Of course, it is possible to construct a differential characteristic with limited number of active Sboxes, but this is not the efficiency we are targeting.

We have also tried to modify the framework such that the sub-tweak addition is not performed every round. For example, an r rounds deterministic related-tweak differential characteristic can be realized by applying the tweak addition only once at the beginning, see Fig. 3. This way, the sub-tweak difference Δt_0 could neutralize the plaintext input difference Δx_0 and the resulting zero difference would get through the r rounds with probability 1. However, this candidate has an obvious fatal flaw: for any tweak pair the attacker can always set the plaintext input difference to be equal to Δt_0.

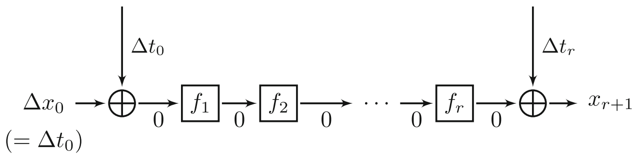

Fig. 3. A defective variant of the MALICIOUS framework. Key addition is omitted.

The above analysis shows that full non-linear layers seem not suitable for the MALICIOUS framework. On the contrary, partial non-linear layers satisfy our requirements. As in that case the Sbox only applies to a part of the internal state, the round function is able to map a non-zero input difference to a non-zero output difference while no active Sbox is activated. In term of building deterministic differential characteristics, we only have to set the difference of the non-linear part of the internal state to be zero rather than the full state. This allows to choose the linear transformation so that the output difference could satisfy the requirements from Algorithm 1.

4 Instantiating the MALICIOUS Framework with LowMC

In this section, we introduce a concrete instantiation of the MALICIOUS framework, called LowMC-M, which is based on the family of block ciphers LowMC.

4.1 LowMC

LowMC [2] is a family of block ciphers based on SPN structure with partial non-linear layers. The parameters are flexible and we denote the block size by n, the key size by k, the number of Sboxes applied each round by m and the maximum allowed data complexity by d (d is the $\log_2$ of the allowable data complexity up to which the cipher is expected to give the claimed security). In order to reach the security claims, the number of rounds r is then derived from all these parameters using a round formula, the latest version being given in [34].

At the beginning of the encryption process, a key whitening is performed. The round function at round i consists of four operations in the following order:

- `SboxLayer`. A 3-bit Sbox is applied in parallel on the $s = 3m$ LSBs of the state, while the transformation for the remaining $n - s$ bits is the identity.
- `LinearLayer`(i). The state is multiplied in GF(2) with an invertible $n \times n$ binary matrix L_i which is chosen independently and uniformly at random.
- `ConstantAddition`(i). The state is XORed with an n-bit round constant C_i which is chosen independently and uniformly at random.
- `KeyAddition`(i). The state is XORed with an n-bit round key k_i. To generate k_i, the master key K is multiplied in GF(2) with an $n \times k$ binary matrix KL_i. This matrix is chosen independently and uniformly at random with rank $\min(n, k)$.

4.2 Equivalent Representation of LowMC

As discussed in [14,23,34], round keys and constants in LowMC can be compressed due to the fact that the non-linear layer is partial.

In the round function, it is possible to exchange the order of consecutive linear operations. We swap the order of `LinearLayer` and `KeyAddition` operations while keeping `ConstantAddition` as the last step in round i. Then, the equivalent round key can be written as $k'_i = L_i^{-1}(k_i)$. We observe that the Sbox only operates on the first s bits of the state and does not change the rest of the $n-s$ bits. Thus, we split k'_i into $k_i'^{(0)}$ and $k_i'^{(1)}$, and we can move the addition of $k_i'^{(1)}$ to the beginning of the round. Next, we observe that $k_i'^{(1)}$ can move further up to be combined with k_{i-1} in the previous round. The procedure is illustrated in Fig. 4. In general, if we start from the last round and iterate this procedure recursively until all the additions to the linear part have been moved to the beginning of the algorithm, we will end up with an equivalent representation where all the round keys are reduced to s bits apart from the whitening key. We remark that the same reasoning can be applied to the round constants.

This optimized representation can also reduce the implementation cost of the key schedule. Since all transformations performed during the optimization are linear and since the key schedule is itself linear, these transformations can be composed with the key schedule in order to compute the new $3m$-bit round keys directly. We refer to [14] for details.

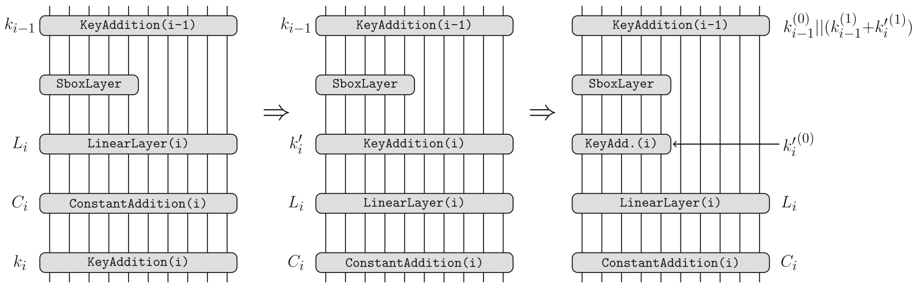

Fig. 4. Simplified representation of LowMC.

4.3 LowMC-M

We will directly use the simplified representation of LowMC as a starting point in our design, with a further modification: we move `LinearLayer` behind `SboxLayer` in every round[3].

LowMC-M is a family of tweakable block ciphers built upon LowMC with an additional transformation in each round:

– `TweakAddition`(i). The non-linear part of the state is XORed with an s-bit sub-tweak t_i just after `KeyAddition`. t_i is generated from a XOF whose input is the original tweak value T.

The XOF is based on SHAKE128 or SHAKE256, depending on the key size. All the other transformations of the round function are the same as for LowMC. The round function is finally composed of the following operations (Fig. 5):

$$\texttt{TweakAddition}(i) \circ \texttt{KeyAddition}(i) \circ \texttt{ConstantAddition}(i) \circ \texttt{LinearLayer}(i) \circ \texttt{SboxLayer}$$

The encryption starts with a key and tweak whitening and the sizes of k_0 and t_0 are both n. The derivation formula for the number of rounds r is the same as for LowMC.

[3] The resulting primitive is an equivalent representation of a LowMC instantiation with different linear layers, key schedule and round constants, because these components are chosen randomly.

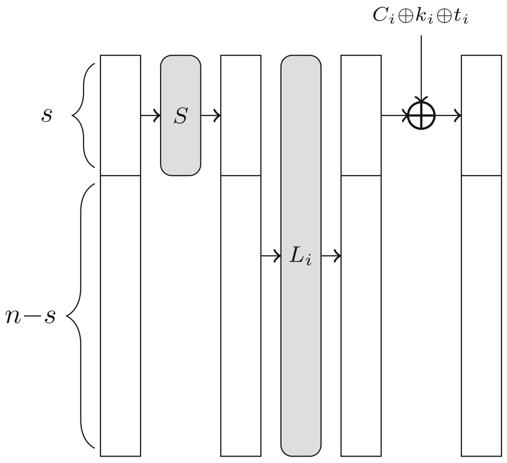

Fig. 5. A single round of LowMC-M.

Notations for LowMC-M. During a differential cryptanalysis, we denote by X_i the i-th round state difference before the `LinearLayer` transformation. Given a matrix L_i, we denote its j-th row by $L_i[j,*]$, and partition L_i into four sub-matrices:

$$L_i = \left[\begin{array}{c|c} L_i^{00} & L_i^{01} \\ \hline L_i^{10} & L_i^{11} \end{array}\right]$$

where $L_i^{00} \in \mathrm{GF}(2)^{s\times s}$, $L_i^{01} \in \mathrm{GF}(2)^{s\times(n-s)}$, $L_i^{10} \in \mathrm{GF}(2)^{(n-s)\times s}$, $L_i^{11} \in \mathrm{GF}(2)^{(n-s)\times(n-s)}$. With this notation, L_i^{00} and L_i^{01} will map $X_i^{(0)}$ and $X_i^{(1)}$ to the non-linear part of the state, respectively. And L_i^{10} and L_i^{11} will map $X_i^{(0)}$ and $X_i^{(1)}$ to the linear part of the state, respectively.

4.4 Embedding a Backdoor into LowMC-M

There are many forms of differential cryptanalysis that can perform a key recovery attack, such as the impossible differential attack, the boomerang attack, etc. For LowMC-M, we use the plain version where the attacker can deduce full or partial information about the r-th round key from a differential characteristic over $r-1$ rounds.

Since an $(r-1)$-round deterministic differential characteristic can only reveal the s-bit sub-key k_r of the r-th round, more deterministic differential characteristics should be added in order to eventually recover the full key. After k_r has been retrieved, the cipher can be reduced to $r-1$ rounds and thus another s-bit sub-key k_{r-1} can be recovered from an $(r-2)$-round deterministic differential characteristic. Finally, assume that there are a total of a such deterministic differential characteristics embedded in LowMC-M (one on $r-1$ rounds, one on $r-2$ rounds, etc., see Fig. 6), then $a \cdot s$ sub-key bits can be recovered. As the key schedule is fully linear and each matrix inside the key schedule is generated independently and uniformly at random, it implies that one will recover $a \cdot s$ bits of information about the key by solving a system of linear equations. Therefore, at most $a = \lceil k/s \rceil$ deterministic differential characteristics are needed to recover the full key.

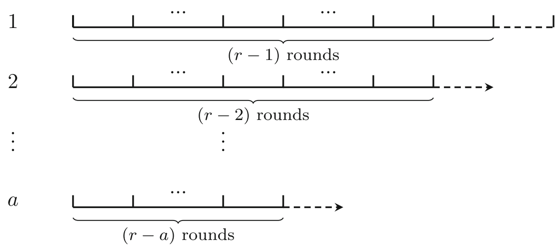

Fig. 6. The deterministic differential characteristics embedded into LowMC-M.

Now, we explain how to embed such differential characteristics into an instantiation of LowMC-M. The general procedure is given in Algorithm 1. The a malicious tweak pairs are chosen by the designer at the very beginning and the corresponding sub-tweak differences are computed. Then, the linear layer matrix L_i is generated along with the generation of the deterministic differential characteristics, **round by round**.

Firstly, we explain how to generate the linear layer matrices. Note that in order to have a deterministic differential characteristic over i rounds, only the linear layer matrices of the first $i-1$ rounds have to be specifically designed as the matrix L_i has no impact on the differences of the i-th round Sboxes. Assuming we have already embedded a deterministic differential characteristics over i rounds, then all the linear layer matrices of the first $i-1$ rounds of LowMC-M have been fixed accordingly. If we plan to extend b ($b \leq a$) of the a deterministic differential characteristics by one more round, the matrix L_i should be specified. Denote by SX_i the set of $X_i^{(1)}$ of those deterministic differential characteristics that will be extended in the next round. Here, SX_i refers to the b differential characteristics. Since the non-linear state difference $X_i^{(0)}$ equals to zero for all the b differential characteristics, the set SX_i will determine the differential in the following round. Given the difference set SX_i, the output differences after the multiplication by the matrix L_i^{01} should cancel the following sub-tweak differences so that the b differential characteristics will activate no Sbox in round $i+1$. We detail the generation of L_i in Algorithm 2.

Denote the $b \times (n-s)$ matrix in Equation (2) by MX_i. We emphasize that the rank of MX_i should be $\min(b, n-s)$, otherwise Equation (2) is likely to have no solution. In practice, b is always smaller than $n-s$ for a normal parameters set of LowMC-M. Thus, this requirement also means that the binary vectors of $X_i^{(1)}$ in SX_i should be linearly independent.

Algorithm 2: Generate linear layer matrix L_i.

```
Input  : The set $SX_i = (X_{i,1}^{(1)}, X_{i,2}^{(1)}, \cdots, X_{i,b}^{(1)})$ and the sub-tweak differences
         $(\Delta t_i^1, \Delta t_i^2, \cdots, \Delta t_i^b)$ for the $b$ differential characteristics.
Output: Matrix $L_i$

while True do
    for j from 1 to s do
        Solve the following system of linear equations and randomly pick one
        solution of $x = (x_1, x_2, ..., x_n)$ as $L_i^{01}[j, *]$.
        $$\begin{pmatrix} X_{i,1}^{(1)}[1] & X_{i,1}^{(1)}[2] & ... & X_{i,1}^{(1)}[n-s] \\ X_{i,2}^{(1)}[1] & X_{i,2}^{(1)}[2] & ... & X_{i,2}^{(1)}[n-s] \\ & & \vdots & \\ X_{i,b}^{(1)}[1] & X_{i,b}^{(1)}[2] & ... & X_{i,b}^{(1)}[n-s] \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-s} \end{pmatrix} = \begin{pmatrix} \Delta t_i^1[j] \\ \Delta t_i^2[j] \\ \vdots \\ \Delta t_i^b[j] \end{pmatrix} \quad (2)$$
    end
    Randomly select the sub-matrices $L_i^{00}, L_i^{10}$ and $L_i^{11}$.
    if $L_i$ is full rank then
        return $L_i$
    end
end
```

The whole process of generating an instance of LowMC-M is given here:

1. Select a different pairs of tweaks of any desired length and compute the corresponding sub-tweak differences in all rounds for each pair of tweaks.
2. For each tweak pair, choose an n-bit value of the plaintext difference ΔP as the input difference for the embedded differential characteristic, while setting the first s bits of ΔP to be equal to $\Delta t_0^{(0)}$.
3. For the a differential characteristics, compute $X_1^{(1)} = \Delta P^{(1)} \oplus \Delta t_0^{(1)}$ and if the binary vectors of SX_1 are not linearly independent, then go back to step 2.
4. For round i from 1 to $r-2$:
   - Generate the matrix L_i using Algorithm 2 with SX_i and the corresponding sub-tweak differences as inputs[4].
   - Except for the last loop, compute the set of SX_{i+1} through the matrix multiplication of L_i. If the binary vectors of SX_{i+1} are not linearly independent, repeat this loop.
5. Choose L_{r-1} and L_r independently and uniformly at random from all invertible $n \times n$ binary matrices.
6. For all rounds i, choose KL_i independently and uniformly at random from all $n \times k$ binary matrices of rank $\min(n, k)$ and the round constants C_i as well.

[4] Starting from round $r-a+1$, the number of deterministic differential characteristics decrements by 1 at every loop.

Recovering the Secret Key With the Backdoor. The backdoor is the a malicious tweak pairs and the corresponding plaintext differences. With the knowledge of these related-tweak differential characteristics, the designer can recover the full key in a very short time. To create the a plaintext differences, the designer can firstly choose a random P, then compute $P_i = P \oplus \Delta P_i$ for $i \in \{1, \cdots, a\}$. We note the fact that for any non-zero probability differential (Δ_1, Δ_2) of LowMC-M Sbox, where $\Delta_1 \neq 0$ and $\Delta_2 \neq 0$, there is only one unordered pair of inputs/outputs of the Sbox satisfying the differential. If each plaintext difference is used only once in the attack, then two sub-key candidates will remain for each Sbox as we cannot determine which order of the input/output pair of the targeted Sbox should be in the attack. The wrong sub-key candidate can be filtered by repeating the attack with another pair of plaintexts of the same difference. By doing so, $a \cdot s$ bits of information of the key can be retrieved in the end. Later, the remaining $(k - a \cdot s)$ key bits, if they exist, can be brute forced. Finally, the key recovery requires $2(a+1) + \max(k - a \cdot s, 0)$ encryptions and the data complexity is $2(a + 1)$.

Note that the bit length of $X_i^{(1)}$ is $n - s$. In order to ensure that Equation (2) is solvable, the number of differential characteristics that are embedded in LowMC-M should not be higher than $n - s$. Generally, this bound is much higher than the number of differential characteristics that is actually needed in a concrete instantiation. Last but not least, one may wonder why we chose different malicious tweak pairs for the a related-tweak differential characteristics (indeed using a single malicious tweak pair would work), but we recommend doing so for security reasons as we will explain in Sect. 5.1.

4.5 Parameters

The design goal of LowMC-M is to keep the backdoor and the cipher secure, but also to ensure the efficiency of the key recovery using the backdoor. Based on these principles, we selected some instantiation parameters[5] and we present them in Table 1. The security analysis is given in Sect. 5 and Sect. 6.

Regarding the performances, we evaluated the corresponding LowMC used in the LowMC-M instances. The LowMC implementations we benchmarked are optimized for AVX2 instructions. Measurements were performed on an AMD EPYC 7401 running Ubuntu 18.04. We tested several instances and we observed that a single encryption generally costs around 10000 to 30000 cycles depending on the parameters, the block size (= key size) ranging from 128 to 256 bits.

[5] The reference code of LowMC-M generation can be found at https://github.com/MaliciousLowmc/LowMC-M.

Table 1. A range of different parameters sets of LowMC-M instantiations. For each instantiation, the malicious tweak pair that triggers each embedded differential characteristic is unique. d is the log_2 of the allowed data complexity, a is the number of differential characteristics embedded.

block size n	non-linear s	key size k	data d	rounds r	#differentials a	XOF
128	3	128	64	208	43	SHAKE128
	6	128	64	104	21	SHAKE128
	9	128	64	70	14	SHAKE128
	30	128	64	23	5	SHAKE128
	90	128	64	14	2	SHAKE128
256	3	256	64	384	85	SHAKE256
	9	256	64	129	28	SHAKE256
	60	256	64	21	5	SHAKE256
	120	256	64	14	3	SHAKE256

5 Backdoor Security

In this section, we will discuss the backdoor security of LowMC-M with respect to the notions mentioned in Sect. 2.2: undetectability, undiscoverability, untraceability and practicability.

5.1 Undetectability

In this subsection, we discuss whether a LowMC-M instance containing a backdoor is distinguishable from a random LowMC-M with no backdoor embedded. Since the only difference between these two cases lies in the way the linear layer matrices are generated, we will investigate the properties of these matrices.

We now would like to show that all embedded differential characteristics must use distinct tweak pairs in order to maintain undetectability. Assuming there is a backdoored LowMC-M instance that is generated following the steps described in Sect. 4.4 and a total of a deterministic related-tweak differential characteristics are embedded, while only a' $(< a)$ different tweak pairs are used during the generation phase. Let c_j denote the number of embedded differential characteristics triggered by the same tweak pair, with $j \in \{1, \ldots, a'\}$. We will show that some dependency will exist in the linear layer matrices for the first i $(\leq r - a)$ rounds, consequently some additional deterministic related-tweak differential characteristics over the first i rounds can be recovered.

Definition 3. *For a* LowMC-M *instance, A_i is the matrix of dimension $(i \cdot s) \times (n - s)$ defined as:*

$$\begin{pmatrix} L_1^{01} \\ L_2^{01} \cdot L_1^{11} \\ L_3^{01} \cdot L_2^{11} \cdot L_1^{11} \\ \vdots \\ L_i^{01} \cdot L_{i-1}^{11} \cdot \ldots \cdot L_1^{11} \end{pmatrix}$$

We remark that a malicious plaintext difference ΔP can be retrieved if the corresponding malicious tweak pair is provided: in order to have a deterministic differential characteristic all Sboxes must be differentially inactive (*i.e.*, the input difference of each Sbox should be zero) and thus for a malicious tweak pair that takes any of the a' different values, recovering $\Delta P^{(0)}$ (the non-linear part of ΔP) is straightforward as it is equal to the sub-tweak difference $\Delta t_0^{(0)}$. After that, one just needs to retrieve the remaining part $\Delta P^{(1)}$. In order to have a deterministic differential characteristic over the first two rounds, $L_1^{01}(X_1^{(1)})$ should be equal to Δt_1, where $X_1^{(1)} = \Delta P^{(1)} \oplus \Delta t_0^{(1)}$. To extend the differential characteristic to the third round, $L_2^{01} \cdot L_1^{11}(X_1^{(1)})$ should be equal to Δt_2. Continuing this process until the i-th round, we can create a system of linear equations with $n-s$ binary variables:

$$\begin{pmatrix} L_1^{01} \\ L_2^{01} \cdot L_1^{11} \\ L_3^{01} \cdot L_2^{11} \cdot L_1^{11} \\ \vdots \\ L_{i-1}^{01} \cdot L_{i-2}^{11} \cdot \ldots \cdot L_1^{11} \end{pmatrix} \cdot (X_1^{(1)}) = A_{i-1} \cdot (X_1^{(1)}) = \begin{pmatrix} \Delta t_1 \\ \Delta t_2 \\ \Delta t_3 \\ \vdots \\ \Delta t_{i-1} \end{pmatrix} \tag{3}$$

Solving Eq. (3) will output the solution of $X_1^{(1)}$, then the remaining part $\Delta P^{(1)}$ can be recovered naturally. However, there may be more solutions as the number of solutions is determined by the rank of A_{i-1}.

In cases where the number of rounds i is large enough such that $(i-1) \cdot s \gg (n-s)$, if all the linear layer matrices are chosen independently and uniformly at random, the rank of A_{i-1} will be $n-s$ with very high probability. However, for a LowMC-M instance with backdoor embedded, since the linear layer matrices are specially designed, the rank of A_{i-1} can not be determined similarly.

Determining the Rank of A_{i-1}. We first introduce the following definition.

Definition 4. *If M is an $n \times m$ binary matrix and v is an n-bit vector, the solution space $sol(M, v)$ is defined as: $sol(M, v) = \{x^T \in \{0,1\}^m : Mx = v\}$.*

Assume that a special LowMC-M instance is generated with c related-tweak deterministic differential characteristics over i rounds while only one malicious tweak pair is used. During the generation of L_j^{01}, $j \in \{1, \ldots, i-1\}$, Equation (2) could be simplified as:

$$MX_j \cdot x = \mathbf{1} \;\; or \;\; MX_j \cdot x = \mathbf{0} \tag{4}$$

where $\mathbf{0}$ and $\mathbf{1}$ are c-bit vectors full of zeros and ones, respectively.

Denote by V the union of $sol(MX_1, \mathbf{1})$ and $sol(MX_1, \mathbf{0})$, the rows of L_1^{01} are chosen from V. Since the dimensions of $sol(MX_1, \mathbf{1})$ and $sol(MX_1, \mathbf{0})$ are both $n-s-c$, then the dimension of V is $n-s-c+1$. When $j = 2$, Eqs. (4) can be represented by:

$$MX_1 \cdot (L_1^{11})^T \cdot x = \mathbf{1} \;\; or \;\; MX_1 \cdot (L_1^{11})^T \cdot x = \mathbf{0}$$

because $X_2^{(1)} = L_1^{11} \cdot X_1^{(1)}$. The rows of L_2^{01} are chosen from $sol(MX_1 \cdot (L_1^{11})^T, \mathbf{1})$ or $sol(MX_1 \cdot (L_1^{11})^T, \mathbf{0})$. Before we continue, we will use the following lemma.

Lemma 1. *Let M_1 and M_2 be two binary matrices of dimension $(n \times m)$ and $(m \times m)$ respectively. If $x \in sol(M_1 \cdot M_2, v)$, then $x \cdot M_2^T \in sol(M_1, v)$ for any n-bit vector v.*

Proof. For any $x \in sol(M_1 \cdot M_2, v)$, we have $(M_1 \cdot M_2) \cdot x^T = v$. It can be represented by $M_1 \cdot (M_2 \cdot x^T) = v$, thus $(M_2 \cdot x^T)^T = x \cdot M_2^T \in sol(M_1, v)$. $\square$

According to Lemma 1, if $x \in sol(MX_1 \cdot (L_1^{11})^T, \mathbf{1})$, then $x \cdot L_1^{11} \in sol(MX_1, \mathbf{1})$ and also if $x \in sol(MX_1 \cdot (L_1^{11})^T, \mathbf{0})$, then $x \cdot L_1^{11} \in sol(MX_1, \mathbf{0})$. Thus, all the rows of $L_2^{01} \cdot L_1^{11}$ are in the space V. Similarly, we can get the same results for $L_3^{01} \cdot L_2^{11} \cdot L_1^{11}, \cdots, L_{i-1}^{01} \cdot L_{i-2}^{11} \cdot \ldots \cdot L_1^{11}$. To summarize, all the rows of A_{i-1} for this special LowMC-M instance are chosen from the space V of dimension $n-s-c+1$. Thus, the rank of A_{i-1} is $n-s-c+1$.

Let us return back to the previous LowMC-M instance mentioned at the beginning of this subsection. We can divide the a differential characteristics into a' sub-groups where each sub-group includes c_j differential characteristics that are triggered with the same tweak pair, $j \in \{1, \ldots, a'\}$. Then, the space V will be the intersection of all the spaces that are determined by the a' sub-groups. We summarize the result as follows.

Proposition 1. *If there is a total of a' different malicious tweak pairs and each of them is used to build c_j deterministic differential characteristics over i rounds in an instance of LowMC-M, with $(i-1) \cdot s \gg (n-s)$, then the rank of A_{i-1} will be $n - s - \sum_{j=1}^{a'}(c_j - 1)$.*

As a result, the rank of A_{i-1} is $n-s-\sum_{j=1}^{a'}(c_j-1)$ and a total of $2^{\sum_{j=1}^{a'}(c_j-1)}$ deterministic differential characteristics for each of the a' tweak pairs can be recovered by the designer. Note that the rank of A_{i-1} can be easily computed by any entity. Compared to the full rank A_{i-1} for a random LowMC-M with no backdoor embedded, the unusual property of A_{i-1} for the backdoored LowMC-M will uncover the existence of the backdoor if $a' < a$. However, if $a' = a$, that is,

$c_j = 1$ for all $j \in \{1, \ldots, a'\}$, then A_{i-1} will be full rank. Therefore, in order to keep the backdoor of LowMC-M undetectable, we recommend to not use the same tweak pair for building more than one differential characteristics in the generation phase.

5.2 Undiscoverability

In this subsection, we discuss whether the backdoor from a LowMC-M instance can be efficiently recovered by an attacker. Recall that some unknown deterministic related-tweak differential characteristics potentially exist in LowMC-M, according to Property 1. Instead of considering the embedded backdoor exclusively, we evaluate the complexity of finding any useful deterministic related-tweak differential characteristics for an attacker. Basically, the complexity is based on the XOF security properties.

We simply adopt the security analysis for the general MALICIOUS framework in Sect. 3.5. For any LowMC-M instance, the bound r' derived from Formula 1 is much smaller compared to the total number of rounds, which poses no threat to the backdoor. We list the evaluation for some instances in Table 2.

We can examine the undiscoverability security from another perspective. Note that deterministic related-tweak differential characteristics can be derived as long as Eq. (3) is solvable. The requirement for the equation to be solvable is that the ranks of the coefficient matrix A_{i-1} and the augmented matrix of Eq. (3) are equal, which means that the vector on the right side of the equation, denoted as v, has to be a combination of the columns of A_{i-1}. Observe that the number of such combinations is 2^{α}, α being the rank of A_{i-1} and it can be computed according to Proposition 1. As for vector v, it is random due to the XOF and its size is $s \cdot (i-1)$. In conclusion, Eq. (3) is solvable with probability $2^{\alpha - s\cdot(i-1)}$, that is, the complexity of finding an i-round deterministic related-tweak differential characteristic is $2^{s\cdot(i-1)-\alpha}$. We define r'' to represent the value of i that turns the complexity to be equal to the key space size

$$r'' = \frac{k+\alpha}{s} + 1 \tag{5}$$

The maximal value is $r'' = \frac{k+n}{s}$ when A_{i-1} is full rank of $n-s$. Still, r'' is much smaller than the number of rounds of any LowMC-M instance, see examples in Table 2.

To summarize, the backdoor and the other potential deterministic related-tweak differential characteristics of the same length are fully protected by the XOF, and its recovery is as hard as brute forcing the key.

Table 2. Backdoor security evaluation for LowMC-M-n/s with block size n, key size n, non-linear layer size s and log_2 data complexity 64. r is the actual number of rounds of the instance, r' and r'' are defined in Formulas 1 and 5 respectively.

Parameters	r	r'	r''
LowMC-M-128/3	208	44	86
LowMC-M-128/6	104	23	43
LowMC-M-128/9	70	16	29
LowMC-M-128/30	23	6	9
LowMC-M-128/90	14	3	3
LowMC-M-256/3	384	87	170
LowMC-M-256/9	129	30	57
LowMC-M-256/60	21	6	9
LowMC-M-256/120	14	4	5

5.3 Untraceability and Practicability

As for practicability, only negligible data and computation are required to launch a full key recovery attack with the knowledge of the backdoor, as explained in Sect. 4.4. Thus, the full key can be recovered within seconds.

Since the usage of the backdoor requires chosen tweaks, the malicious tweaks can be detected by the user once the designer makes queries to attack him, which means the backdoor is traceable. Besides, as only a few queries are needed to launch an attack with the knowledge of the backdoor, the user is able to quickly brute force the queries to find out the malicious tweak pairs.

6 Cipher Security

In this section, we study the security of LowMC-M as a tweakable block cipher.

6.1 Attacks Based on Tweak

In comparison to LowMC, an additional tweak addition is introduced in LowMC-M. Theoretically, this feature will provide extra degrees of freedom for the attacker and might naturally weaken LowMC-M when compared to LowMC. However, since the tweak schedule is an XOF, the attacker cannot control its output. Even if the attacker could brute force some structures on the sub-tweaks for a few rounds, this will result in the remaining rounds containing completely random structure, which consequently prevent the attacker utilizing these remaining rounds for what should have been the best attack on LowMC. Hence, we believe that the extra degrees of freedom provided by the tweak is not easily usable and will not lead to any important improvement over classical attack, including the existing cryptanalysis [15,16,34] on LowMC.

6.2 Attacks Without Tweak

All the current attacks [15,16,34] on LowMC have been conducted under the assumption that the linear layer matrices of LowMC are chosen independently and uniformly at random. Except the tweak addition, LowMC-M has the equivalent specification to LowMC. The only difference lies in the way the linear layer matrices L_i are chosen during the generation phase. In order to prove that the security of LowMC-M is on par with that of LowMC, we need to show that the linear layer matrices of LowMC-M are indistinguishable from those of LowMC-M from the perspective of the attacker. We will evaluate this with respect to the randomness and independence.

Randomness of Linear Layer Matrices. The randomness of the linear layer matrix L_i is analyzed by scrutinizing its four sub-matrices one by one.

L_i^{00} and L_i^{10}. As described in Algorithm 2, the two sub-matrices L_i^{00} and L_i^{10} of L_i are chosen independently and uniformly at random for each round.

L_i^{11}. Even though L_i^{11} is chosen randomly in Algorithm 2, there is a supplementary requirement during the generation phase. That is, the binary vectors of SX_{i+1} have to be linearly independent, which adds an extra constraint to L_i^{11} since each binary vector of SX_{i+1} is obtained by:

$$X_{i+1}^{(1)} = L_i^{11} \cdot X_i^{(1)} \tag{6}$$

and thus the transformation of L_i^{11} should map a set of linearly independent vectors to another set of linearly independent vectors. Since L_i^{11} is chosen randomly and all the $X_i^{(1)}$ involved are linearly independent, every $X_{i+1}^{(1)}$ in SX_{i+1} produced by Formula 6 can be regarded as random binary vectors and are independent from each other. On the other hand, note that at most $a = \lceil k/s \rceil$ differential characteristics are embedded in LowMC-M, which means that the size of SX_{i+1} is $\lceil k/s \rceil$ at most. For any reasonable parameter set, we will have $\lceil k/s \rceil \ll (n-s)$. Based on Lemma 2 below, we can compute the probability that the set SX_{i+1} is linearly independent. As a result, the probability is almost 1, which is also verified from our experiments.

Hence, the constraint on L_i^{11} is very loose. The final selection of L_i^{11} will not introduce any special property.

Lemma 2. *[24, adapted] For m random n-bit vectors over $\mathbb{F}_2$ $(m \leqslant n)$, the probability that they are linearly independent is $p(m) = \prod_{k=0}^{m-1}(1 - 2^{k-n})$. In particular, $p(n) > 0.2887$.*

L_i^{01}. L_i^{01} is the essential part for embedding backdoors, and thus it is the one specially designed. The row length of L_i^{01} is $n - s$ bits, while in the generation phase each row is chosen from a sub-space of dimension $n - s - b$ which is determined by the corresponding Equation (2), b being the size of SX_i. However, we will show that for the attacker this special chosen L_i^{01} is still indistinguishable from a randomly chosen one.

Observe that both MX_i and the sub-tweak difference vector in Equation (2) are unknown for the attacker, thus the solution space is unidentified. Moreover, the solution space for each row of L_i^{01} could be different due to the sub-tweak difference. Therefore, it is impossible for the attacker to trace some rows of L_i^{01} to the targeted hidden sub-space.

To summarize, the four sub-matrices are indistinguishable from random matrices for the attacker. The only connection between these four sub-matrices is that the combined matrix L_i should be invertible, which is also the same for LowMC, so it reveals no additional information. Hence, we conclude that for the attacker the matrix L_i is indistinguishable from a random matrix.

Independence Between Linear Layer Matrices. The definition of A_r captures partial information of the matrices that includes L_i^{10} and L_i^{11} over r consecutive rounds. If the linear layer matrices are chosen independently and uniformly at random, the resultant A_r should be random, thus the rank of A_r will be $n-s$ when $r \cdot s \gg (n-s)$. If the rank for a LowMC-M instance is smaller than $n-s$, it will imply a connection between these matrices. As suggested in Proposition 1, the rank of A_r can be computed by $n - s - \sum_{i=1}^{a'}(c_i - 1)$. In order to eliminate the connections, each c_i should equal to 1, that is, different malicious tweak pairs should be used to build different differential characteristics during the generation phase.

The two sub-matrices L_i^{00} and L_i^{10} are chosen randomly and independently, so it will not impose any connection between the matrices.

We remark that even if there is some dependence existing between the linear layer matrices, the cipher security is still unlikely to be threatened. Yet, we conservatively recommend to avoid such dependency in a LowMC-M instance.

7 Conclusion

In this article, we proposed the MALICIOUS framework for embedding backdoors into tweakable block ciphers. The backdoor is a set of related-tweak differential characteristics with probability 1, from which the secret key can be recovered fully and efficiently. Besides, the backdoor security of our proposal is reduced to the target-difference resistance (a variant of the classical collision resistance, with the same generic complexity) of the XOF employed in the cipher. We also proposed several concrete instances LowMC-M, which are directly inspired from the block cipher LowMC.

We have proved that it is possible to build a secure and efficient backdoor into tweakable block ciphers. Third party analysis is of course required to fully understand its security, but our proposal could be a new interesting direction towards building backdoors in symmetric-key primitives.

Not only this result will increase the community's awareness to potential backdoors in symmetric-key primitives, but it can also lead to new applications.

It has been shown in [35] that a backdoored block cipher is equivalent to a public key encryption where the backdoor is regarded as the secret key. Even though our proposal does not yet reach the usability of a public encryption scheme, building public-key primitives out of symmetric-key ones has been a long standing open problem.

We envision several future works after this first step. Other cryptanalysis techniques than just a plain differential attack (such as impossible differential attacks, boomerang attacks, integral attacks, etc.) might also be used to build backdoors and could allow us to build more efficient or more usable designs. It would also be interesting to build other types of backdoored primitives, such as Message Authentication Codes (MAC), Authenticated Encryption (AE), etc. which might require totally different design strategies. Finally, our proposal remains somewhat traceable (once the backdoor used against him, a user could try to check all of its tweak values queried and check which tweaks pair leads to a related-tweak differential with a very good probability) and it would be interesting to study new techniques or protocols to reduce this detection surface as much as possible.

Acknowledgements. The authors would like to thank the anonymous referees and Eik List for their helpful comments. We also thank Shiyao Chen for his help on automatic search tools. We furthermore thank the Picnic team for providing standalone optimized implementations of LowMC.

References

1. Albertini, A., Aumasson, J.-P., Eichlseder, M., Mendel, F., Schläffer, M.: Malicious hashing: Eve's variant of SHA-1. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 1–19. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13051-4_1
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 430–454. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_17
3. AlTawy, R., Youssef, A.M.: Watch your constants: malicious Streebog. Cryptology ePrint Archive, Report 2014/879 (2014). https://eprint.iacr.org/2014/879
4. Angelova, V., Borissov, Y.: Plaintext recovery in DES-like cryptosystems based on S-boxes with embedded parity check. Serdica J. Comput. **7**(3), 257–270 (2013)
5. Bannier, A., Bodin, N., Filiol, E.: Partition-based trapdoor ciphers. Cryptology ePrint Archive, Report 2016/493 (2016). http://eprint.iacr.org/2016/493
6. Bannier, A., Filiol, E.: Mathematical backdoors in symmetric encryption systems-proposal for a backdoored AES-like block cipher. arXiv preprint arXiv:1702.06475 (2017)
7. 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.) EUROCRYPT 2015. LNCS, Part I, vol. 9056, pp. 315–342. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_13

8. Barker, E.B., Kelsey, J.M.: Recommendation for random number generation using deterministic random bit generators (revised). US Department of Commerce, Technology Administration, National Institute of Standards and Technology, Computer Security Division, Information Technology Laboratory (2007)
9. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013). http://eprint.iacr.org/2013/404
10. Bernstein, D.J., Lange, T., Niederhagen, R.: Dual EC: a standardized back door. In: Ryan, P.Y.A., Naccache, D., Quisquater, J.-J. (eds.) The New Codebreakers. LNCS, vol. 9100, pp. 256–281. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49301-4_17
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R., Viguier, B.: KangarooTwelve: fast hashing based on Keccak-p. In: Preneel, B., Vercauteren, F. (eds.) ACNS 2018. LNCS, vol. 10892, pp. 400–418. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93387-0_21
12. Biham, E.: Cryptanalysis of Patarin's 2-round public key system with S boxes (2R). In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 408–416. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_28
13. Calderini, M., Sala, M.: On differential uniformity of maps that may hide an algebraic trapdoor. In: Maletti, A. (ed.) CAI 2015. LNCS, vol. 9270, pp. 70–78. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23021-4_7
14. Dinur, I., Kales, D., Promitzer, A., Ramacher, S., Rechberger, C.: Linear equivalence of block ciphers with partial non-linear layers: application to LowMC. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, Part I, vol. 11476, pp. 343–372. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_12
15. Dinur, I., Liu, Y., Meier, W., Wang, Q.: Optimized interpolation attacks on LowMC. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, Part II, vol. 9453, pp. 535–560. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_22
16. Dobraunig, C., Eichlseder, M., Mendel, F.: Higher-order cryptanalysis of LowMC. In: Kwon, S., Yun, A. (eds.) ICISC 2015. LNCS, vol. 9558, pp. 87–101. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30840-1_6
17. Dworkin, M.J.: SHA-3 standard: permutation-based hash and extendable-output functions. Technical report (2015)
18. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.-X.: Block ciphers that are easier to mask: how far can we go? In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 383–399. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40349-1_22
19. Gilbert, H., Peyrin, T.: Super-Sbox cryptanalysis: improved attacks for AES-like permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13858-4_21
20. Guo, J., Nikolic, I., Peyrin, T., Wang, L.: Cryptanalysis of Zorro. Cryptology ePrint Archive, Report 2013/713 (2013). http://eprint.iacr.org/2013/713
21. Iwamoto, M., Peyrin, T., Sasaki, Y.: Limited-birthday distinguishers for hash functions. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, Part II, vol. 8270, pp. 504–523. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_26
22. Jean, J., Nikolić, I., Peyrin, T.: Tweaks and keys for block ciphers: the `TWEAKEY` framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, Part II, vol. 8874, pp. 274–288. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_15

23. Kales, D., Perrin, L., Promitzer, A., Ramacher, S., Rechberger, C.: Improvements to the linear operations of LowMC: a faster picnic (2018)
24. Kolchin, V.: Random Graphs. Cambridge University Press, Cambridge (1999)
25. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_3
26. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. J. Cryptol. **24**(3), 588–613 (2011)
27. Matyukhin, D., Rudskoy, V., Shishkin, V.: A perspective hashing algorithm. In: Materials of XII Scientific Conference RusCrypto 2010 (2010)
28. Morawiecki, P.: Malicious Keccak. Cryptology ePrint Archive, Report 2015/1085 (2015). https://eprint.iacr.org/2015/1085
29. Patarin, J., Goubin, L.: Asymmetric cryptography with S-Boxes is it easier than expected to design efficient asymmetric cryptosystems? In: Han, Y., Okamoto, T., Qing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 369–380. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0028492
30. Patarin, J., Goubin, L.: Trapdoor one-way permutations and multivariate polynomials. In: Han, Y., Okamoto, T., Qing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 356–368. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0028491
31. Paterson, K.G.: Imprimitive permutation groups and trapdoors in iterated block ciphers. In: Knudsen, L. (ed.) FSE 1999. LNCS, vol. 1636, pp. 201–214. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48519-8_15
32. Perrin, L.: Partitions in the S-Box of Streebog and Kuznyechik. IACR Trans. Symm. Cryptol. **2019**(1), 302–329 (2019)
33. Rasoolzadeh, S., Ahmadian, Z., Salmasizadeh, M., Aref, M.R.: Total break of Zorro using linear and differential attacks. Cryptology ePrint Archive, Report 2014/220 (2014). http://eprint.iacr.org/2014/220
34. Rechberger, C., Soleimany, H., Tiessen, T.: Cryptanalysis of low-data instances of full LowMCv2. IACR Trans. Symm. Cryptol. **2018**(3), 163–181 (2018)
35. Rijmen, V., Preneel, B.: A family of trapdoor ciphers. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 139–148. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052342
36. Shishkin, V., Dygin, D., Lavrikov, I., Marshalko, G., Rudskoy, V., Trifonov, D.: Low-weight and hi-end: draft Russian encryption standard. CTCrypt **14**, 05–06 (2014)
37. Shumow, D., Ferguson, N.: On the possibility of a back door in the NIST SP800-90 Dual Ec Prng. In: Proceedings of Cryptology, vol. 7 (2007)
38. Wang, Y., Wu, W., Guo, Z., Yu, X.: Differential cryptanalysis and linear distinguisher of full-round Zorro. Cryptology ePrint Archive, Report 2013/775 (2013). http://eprint.iacr.org/2013/775
39. Wu, H., Bao, F., Deng, R.H., Ye, Q.-Z.: Cryptanalysis of Rijmen-Preneel trapdoor ciphers. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 126–132. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49649-1_11
40. Ye, D.-F., Lam, K.-Y., Dai, Z.-D.: Cryptanalysis of "2 R" schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 315–325. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_20
41. Young, A., Yung, M.: The dark side of "Black-Box" cryptography or: should we trust capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_8

42. Young, A., Yung, M.: Monkey: black-box symmetric ciphers designed for MONopolizing KEYs. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 122–133. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-69710-1_9
43. Young, A., Yung, M.: A subliminal channel in secret block ciphers. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 198–211. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30564-4_14
44. Young, A., Yung, M.: Malicious Cryptography: Exposing Cryptovirology. Wiley, New York (2004)
45. Young, A.L., Yung, M.: Backdoor attacks on black-box ciphers exploiting low-entropy plaintexts. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 297–311. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45067-X_26