J Cryptol (2019) 32:1383–1422
https://doi.org/10.1007/s00145-018-9285-0

Journal of
CRYPTOLOGY

Nonlinear Invariant Attack: Practical Attack on Full SCREAM, iSCREAM, and Midori64

Yosuke Todo
NTT Secure Platform Laboratories, Tokyo, Japan
todo.yosuke@lab.ntt.co.jp

Gregor Leander
Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Bochum, Germany
gregor.leander@rub.de

Yu Sasaki
NTT Secure Platform Laboratories, Tokyo, Japan
sasaki.yu@lab.ntt.co.jp

Communicated by Kaisa Nyberg.

Received 15 June 2017 / Revised 27 February 2018
Online publication 2 April 2018

Abstract. In this paper, we introduce a new type of attack, called *nonlinear invariant attack*. As application examples, we present new attacks that are able to distinguish the full versions of the (tweakable) block ciphers Scream, iScream and Midori64 in a weak-key setting. Those attacks require only a handful of plaintext–ciphertext pairs and have minimal computational costs. Moreover, the nonlinear invariant attack on the underlying (tweakable) block cipher can be extended to a ciphertext-only attack in well-known modes of operation such as CBC or CTR. The plaintext of the authenticated encryption schemes SCREAM and iSCREAM can be practically recovered only from the ciphertexts in the nonce-respecting setting. This is the first result breaking a security claim of SCREAM. Moreover, the plaintext in Midori64 with well-known modes of operation can practically be recovered. All of our attacks are experimentally verified.

Keywords. Nonlinear invariant attack, Boolean function, Ciphertext-only message-recovery attack, SCREAM, iSCREAM, Midori64, CAESAR competition.

1. Introduction

Block ciphers are certainly among the most important cryptographic primitives. Since the invention of the DES [21] in the mid-1970s and even more with the design of the AES [23], a huge amount of research has been done on various aspects of block cipher design and block cipher analysis. In the last decade, many new block ciphers have been proposed that aim at highly resource-constrained devices. Driven by new potential applications like the internet of things, we have witnessed not only many new designs, but also several new cryptanalytic results. Today, we have at hand a well established set of cryptanalytic tools that, when are carefully applied, allow to gain significant confidence in the security of a block cipher design. The most prominent tools here are certainly differential [6] and linear [19] attacks and their numerous variations [2,5,12,14].

Despite this fact, quite some of the recently proposed lightweight block ciphers got broken rather quickly. One of the reasons for those attacks, on what is supposed to be a well-understood field of cryptographic designs, is that the new lightweight block ciphers are designed more aggressively than, e.g., most of the AES candidates. In particular, when it comes to the design of the key schedule, many new proposals keep the design very simple, often using identical round keys. While there is no general defect with such a key schedule, structural attacks become much more of an issue compared to a cipher that deploys a more complicated key schedule. In this paper, we introduce a new structural attack, named *nonlinear invariant attack*. At first glance, it might seem quite unlikely that such an attack could ever be successfully applied. However, we give several examples of ciphers that are highly vulnerable to this attack.

1.1. *Our Contribution*

Given a block cipher $E_k : \mathbb{F}_2^{\mathcal{N}} \to \mathbb{F}_2^{\mathcal{N}}$, the general principle of the nonlinear invariant attack is to find an efficiently computable nonlinear Boolean function $g : \mathbb{F}_2^{\mathcal{N}} \to \mathbb{F}_2$ such that

$$g(P) \oplus g(E_k(P))$$

is constant for any plaintext P and for many possible keys k. Keys such that this term is constant are called weak keys. The function g itself is called *nonlinear invariant* for E_k. Clearly, when the block cipher E_k has a (non-trivial) nonlinear invariant function g, $g(P) \oplus g(E_k(P))$ is constant for any plaintext P and any weak key k. On the other hand, for given h pairs of $(P, E_k(P))$, the probability that random permutations have this property is about 2^{-h+1} when g is balanced. Therefore, attackers can immediately execute a distinguishing attack. Moreover, if the constant depends on the secret key, an attacker can recover one bit of information about the secret key by using one known plaintext–ciphertext pair.

For round-based block ciphers, our attack builds the nonlinear invariants of the whole cipher from the nonlinear invariants of the single round functions. In order to extend the nonlinear invariant for a single round to the whole cipher, all round keys must be weak keys. It may be infeasible to find such weak-key classes for block ciphers with a non-trivial key schedule. However, as mentioned above, many recent block ciphers are designed for lightweight applications, and they adopt more aggressive designs to achieve high performance even in highly constrained environments. Several lightweight ciphers do not deploy any key schedule at all, but rather use the master key directly as the identical round key for all rounds. In such a situation, the weak-key class of round keys is trivially converted into the weak-key class of the secret key. In particular, when all round keys are weak, this property is iterative over an arbitrary number of rounds.

Table 1. Summary of the nonlinear invariant attack.

	# of weak keys	Max. # of recovered bits	Data complexity	Time complexity
SCREAM	2^{96}	32 bits	33 ciphertexts	32^3
iSCREAM	2^{97}	32 bits	33 ciphertexts	32^3
Midori64	2^{64}	$32h$ bits	$33h$ ciphertexts	$32^3 \times h$

h is the number of blocks in the mode of operation

1.1.1. *(Ciphertext-Only) Message-Recovery Attacks*

The most surprising application of the nonlinear invariant attack is an extension to ciphertext-only message-recovery attacks. Clearly, we cannot execute any ciphertext-only attack without some information on the plaintexts. Therefore, our attack is ciphertext-only attack under the following conditions. Suppose that block ciphers which are vulnerable against the nonlinear invariant attack are used in well-known modes of operation, e.g., CBC, CFB, OFB, and CTR. Then, if the same unknown plaintext is encrypted by the same weak key and different initialization vectors, attackers can practically recover a part of the plaintext only from the ciphertexts.

1.1.2. *Applications*

We demonstrate that our new attack practically breaks the full authenticated encryption schemes SCREAM[1] [10] and iSCREAM [9] and the low-energy block cipher Midori64 [1] in the weak-key setting.

We show that the tweakable block ciphers Scream and iScream have a nonlinear invariant function, and the numbers of weak keys are 2^{96} and 2^{97}, respectively. Midori64 also has a nonlinear invariant function, and the number of weak keys is 2^{64}. Table 1 summarizes the result of the nonlinear invariant attack against SCREAM, iSCREAM, and Midori64. The use of the tweakable block cipher Scream is defined by the authenticated encryption SCREAM, and the final block is encrypted like CTR when the byte length of a plaintext is not multiple of 16. We exploit this procedure and recover 32 bits of the final block of the plaintext if the final block length ranges from 12 to 15 bytes. We can also execute a similar attack against iSCREAM. Note that our attack breaks SCREAM and iSCREAM in the nonce-respecting model. Midori64 is a low-energy block cipher, and we consider the case that Midori64 is used by well-known modes of operation. As a result, we can recover 32 bits in every 64-bit block of the plaintext if Midori64 is used in CBC, CFB, OFB, and CTR.

1.1.3. *Comparison with Previous Attacks*

Leander et al. [17] proposed invariant subspace attack on iSCREAM, which is a weak-key attack working for 2^{96} weak keys. The attack can be a distinguishing attack and key recovery attack in the chosen-message and chosen-tweak model. Guo et al. [8] presented a weak-key attack on full Midori64, which works for 2^{32} weak keys, distinguishes the cipher with 1 chosen-plaintext query, and recovers the key with 2^{16} computations.

[1]Note that throughout the paper SCREAM always refers to the latest version as SCREAM, i.e., SCREAM (v3).

Compared to [17], our attack has double weak-key size, and we distinguish the cipher in the known-message and chosen-tweak model. Compared to [8], our weak-key class is much larger and the cipher is distinguished with 2 known-plaintext queries. In both applications, the key space can be reduced by 1 bit, besides a part of message/plaintext can be recovered from the ciphertext.

1.2. *Related Work*

The nonlinear invariant attack can be regarded as an extension of linear cryptanalysis [19]. While linear cryptanalysis uses a linear function to approximate the cipher, the nonlinear invariant attack uses a nonlinear function and the probability of the nonlinear approximation is one. When g is linear, ciphers that are resistant against the linear cryptanalysis never have a linear approximation with probability one.

The use of the nonlinear approximation has previously been studied. This extension was first discussed by Harpes et al. [13], and Knudsen and Robshaw [15] later investigated the effectiveness deeply. However, they showed that there are insurmountable problems in the general use of nonlinear approximations. For instance, one cannot join nonlinear approximations for more than one round of a block cipher because the actual approximations depend on the specific value of the state and key. Knudsen and Robshaw [15] demonstrated that nonlinear approximations can replace linear approximations in the first and last rounds only. Unfortunately, nonlinear cryptanalysis has not been successful because of this limited application. Our attack can be seen as the first application of the nonlinear cryptanalysis against real ciphers in the past two decades.

Other related attacks are the invariant subspace attack [16,17] and symmetric structures [4,22,24]. Similar to the nonlinear invariant attack, those attacks exploit a cryptanalytic property which continues over an arbitrary number of rounds in the weak-key setting. While the attacker has to choose plaintexts, i.e., are chosen-plaintext attacks, the nonlinear invariant attack does not need to choose plaintexts in general. This in particular allows us to extend the nonlinear invariant attack from a pure distinguishing attack to a (ciphertext-only) message-recovery attack.

1.3. *Paper Organization*

We explain the general ideas and principles of the new attack in Sect. 2. Section 3 explains how, in many cases, nonlinear invariants can be constructed by using an algorithmic approach for most practical ciphers. Moreover, we give a structural reason why some ciphers, more precisely some linear layers, are inherently weak against our attack and why our attack is possible against those ciphers. In Sect. 4, we explain in detail our attacks on SCREAM and iSCREAM. Moreover, Sect. 5 details our nonlinear invariant attack on Midori64. In Sect. 6, we show a new class of weak constants for the nonlinear invariant attack, which expands the class of vulnerable targets. Finally, in Sect. 7, we give some additional insights into the general structure of nonlinear invariant functions and outline some future work.

2. Nonlinear Invariant Attack

In this section, we describe the basic principle of the attack and its extension to (ciphertext-only) message-recovery attacks when used in common modes of operations. While our attack can be applied to any cipher structure in principle, we focus on the case of key-alternating ciphers and later on substitution permutation networks (SPN) ciphers to simplify the description. We start by explaining the basic idea and later how, surprisingly, the attack can be extended to a (ciphertext-only) message-recovery attack in many scenarios.

2.1. *Core Idea*

Let $R : \mathbb{F}_2^{\mathcal{N}} \to \mathbb{F}_2^{\mathcal{N}}$ be the round function of a key-alternating cipher and $R_k(x) = R(x \oplus k)$ be the round function including the key XOR. Thus, for an r-round cipher, the ciphertext C is computed from a plaintext P using round keys k_i as

$$\begin{aligned} x_0 &= P, \\ x_{i+1} &= R_{k_i}(x_i) = R(x_i \oplus k_i) \quad 0 \le i \le r-1, \\ C &= x_r, \end{aligned}$$

where we ignore post-whitening key for simplicity.

The core idea of the nonlinear invariant attack is to detect a nonlinear Boolean function g such that

$$g(R(x \oplus k)) = g(x \oplus k) \oplus c = g(x) \oplus g(k) \oplus c \quad \forall x$$

for many keys k, where c is a constant in $\mathbb{F}_2$. Keys for which this equality holds will be called *weak keys*. The function g itself is called *nonlinear invariant* in this paper.

The important remark is that, if all round keys k_i are weak then

$$\begin{aligned} g(C) &= g(R(x_{r-1} \oplus k_{r-1})) \\ &= g(x_{r-1}) \oplus g(k_{r-1}) \oplus c \\ &= g(R(x_{r-2} \oplus k_{r-2})) \oplus g(k_{r-1}) \oplus c \\ &= g(x_{r-2}) \oplus g(k_{r-2}) \oplus g(k_{r-1}) \\ &\vdots \\ &= g(P) \oplus \bigoplus_{i=0}^{r-1} g(k_i) \oplus \bigoplus_{i=0}^{r-1} c. \end{aligned}$$

Thus, the invariant is iterative over an arbitrary number of rounds and immediately leads to a distinguishing attack.

2.1.1. *Distinguishing Attack*

Assume that we found a Boolean function g that is nonlinear invariant for the round function R_{k_i} of a block cipher. Then, if all round keys are weak, this function g is also nonlinear invariant over an arbitrary number of rounds.

Let $(P_i, C_i)\ 0 \leq i \leq h-1$ be h pairs of plaintexts and their corresponding ciphertexts. Then, $g(P_i) \oplus g(C_i)$ is constant for all pairs. If g is balanced, i.e., $p = \#\{x \in \mathbb{F}_2^{\mathcal{N}} | g(x) = 1\}/2^{\mathcal{N}} = 1/2$, the probability that $g(P_i) \oplus g(C_i) = 0$ in random permutations is $p^2 + (1-p)^2 = 1/4 + 1/4 = 1/2$. Therefore, the probability that random permutations have this property is about 2^{-h+1}. We can practically distinguish the block cipher from random permutations under a *known-plaintext attack*.

While the main focus of this paper is the case that g is balanced, we additionally discuss the case that g is unbalanced. When unbalanced g is applied, e.g., $p < 1/2$, we can still mount a distinguishing attack. When the known-plaintext scenario is used, the probability that $g(P_i) \oplus g(C_i) = 0$ in random permutations is $p^2 + (1-p)^2 = 2p^2 - 2p + 1$. Then the probability that $g(P_i) \oplus g(C_i)$ is preserved by 0 for all h pairs is $(2p^2 - 2p + 1)^{h-1}$. Therefore, the farther from $1/2$, the smaller the distinguishing probability is. When the chosen-plaintext scenario is applied, the success probability of distinguishing attack increases. In this scenario, we prepare the set of chosen plaintext defined as $\mathbb{X}$ satisfying $\#\{x \in \mathbb{X} | g(x) = 1\}/|\mathbb{X}| = 1/2$. Then, the probability that $g(P_i) \oplus g(C_i) = 0$ in random permutations is $\frac{1}{2} \times p + \frac{1}{2} \times (1-p) = 1/2$. Similarly to the case that g is balanced, the probability that $g(P_i) \oplus g(C_i)$ is preserved by 0 for all h pairs is about 2^{-h+1}.

2.1.2. *Suitable Nonlinear Invariants*

We next discuss a particular choice of a nonlinear invariant g for which it is directly clear that weak keys exist. Imagine we were able to identify a nonlinear invariant g for R, i.e., a function such that

$$g(R(x)) \oplus g(x)$$

is constant, where g is actually linear (or constant) in some of the inputs. In this case, all round keys that are zero in the nonlinear components of g are weak. More precisely, without loss of generality, assume that the nonlinear invariant g is linear in the last $\mathcal{N}_\ell$ bits of input (implying that g is nonlinear in the first $\mathcal{N}_f$ bits of input where $\mathcal{N} = \mathcal{N}_\ell + \mathcal{N}_f$). Namely, we can view g as

$$g : (\mathbb{F}_2^{\mathcal{N}_f} \times \mathbb{F}_2^{\mathcal{N}_\ell}) \to \mathbb{F}_2$$

such that

$$g(x, y) = f(x) \oplus \ell(x, y)$$

where the domain of f is nonlinearly involved, and that of ℓ is linearly involved. An example to understand this notation is shown in the following.

Example 1. Let $g : \mathbb{F}_2^4 \to \mathbb{F}_2$ be a nonlinear invariant as

$$g(x_3, x_2, x_1, x_0) = x_3x_2 \oplus x_2 \oplus x_1 \oplus x_0.$$

Then, the function g can be viewed as

$$g(x_3, x_2, x_1, x_0) = f(x_3, x_2) \oplus \ell(x_3, x_2, x_1, x_0),$$

where $f(x_3, x_2) = x_3x_2$ and $\ell(x_3, x_2, x_1, x_0) = x_2 \oplus x_1 \oplus x_0$.

Since g is a nonlinear invariant for R, it holds that

$$g(x, y) \oplus g(R(x, y)) = c,$$

where c is a constant in $\mathbb{F}_2$. Now consider a round key $k \in \mathbb{F}_2^{\mathcal{N}_f} \times \mathbb{F}_2^{\mathcal{N}_\ell}$ of the form $(0, k')$. That is, we consider a round key such that its first $\mathcal{N}_f$ bits are zero. Then it holds that

$$\begin{aligned} g(R_{(0,k')}(x, y)) &= g(R(x, y \oplus k')) \\ &= g(x, y \oplus k') \oplus c \\ &= f(x) \oplus \ell(x, y \oplus k') \oplus c \\ &= f(x) \oplus \ell(x, y) \oplus \ell(0, k') \oplus c \\ &= g(x, y) \oplus g(0, k') \oplus c. \end{aligned}$$

In other words, all those round keys that are zero in the first $\mathcal{N}_f$ bits are weak. Phrased differently, the density of weak keys is $2^{-\mathcal{N}_f}$.

2.1.3. *On Key Schedule and Round Constants*

Many block ciphers generate round keys from the master key by a key schedule. For a proper key schedule, it is very unlikely that all round keys are weak in the above sense. However, many recent lightweight block ciphers do not have a well-diffused key schedule, but rather use (parts of) the master key directly as the round keys. From a performance point of view, this approach is certainly preferable. However, the direct XORing with the secret key often causes vulnerabilities like the slide attack [7] or the invariant subspace attack [16]. To avoid those attacks, round constants are additionally XORed in such lightweight ciphers. While dense and random-looking round constant would be a conservative choice, many such ciphers adopt sparse round constants because they are advantageous in limited memory requirements.

Focusing on the case of identical round keys, assume that there is a Boolean function g which is nonlinear invariant for the round function R. Now if all used round constants rc_i are such that rc_i is only involved in the linear terms of g, the function g is nonlinear invariant for this constant addition. This follows by the same arguments for the weak keys above. We call such constants, in line with the notation of weak keys from above, *weak constants*.

To conclude, given a key-alternating cipher with identical round keys and weak round-constants, any master key that is weak, is immediately weak for an arbitrary number of rounds. In this scenario, the number of weak keys is $2^{\mathcal{N}_\ell}$, or equivalently, the density of weak keys is $2^{-\mathcal{N}_f}$.

2.2. *Message-Recovery Attack*

As described so far, the nonlinear invariant attack leaks at most one bit of the secret key. However, if a block cipher that is vulnerable to the nonlinear invariant attack is used in well-known modes of operation, e.g., CBC, CFB, OFB, and CTR, surprisingly, the attack can be turned into a *ciphertext-only message-recovery attack*.

Clearly, we cannot execute any ciphertext-only attack without some information on the plaintexts. When block ciphers are used under well-known modes of operation, the plaintext itself is not the input of block ciphers and the input is rather initialization vectors. Here we assume that an attacker can collect several ciphertexts where the same plaintext is encrypted by the same (weak) key and different initialization vectors. We like to highlight that this assumption is more practical not only compared to the chosen-ciphertext attack but also to the known-plaintext attack. In practice, for instance, assuming that an application sends secret password several times, we can recover the password practically. While the feasibility depends on the behavior of the application, our attack is highly practical in this case.

2.2.1. *Attack Against CBC Mode*

Figure 1 shows an encryption by the CBC mode. Let P_j be the $(j+1)$th plaintext block, and C_j^i denotes the $(j+1)$th ciphertext block by using the initialization vector IV^i. The attacker aims at recovering the plaintext $(P_0, P_1, \ldots, P_j)$ by observing the ciphertext $(IV^i, C_0^i, C_1^i, \ldots, C_j^i)$. Moreover, we assume that the block cipher E_k is vulnerable against the nonlinear invariant attack, i.e., there is a function g such that $g(x) \oplus g(E_k(x))$ is constant for any $x \in \mathbb{F}_2^{\mathcal{N}}$.

First, we explain how to recover the plaintext P_0 by focusing on the first block. Since E_k is vulnerable against the nonlinear invariant attack, there is a function g such that $g(P_0 \oplus IV^i) \oplus g(C_0^i)$ is constant for any $i \in \{0, 1, \ldots, h\}$. If g would be a linear function,

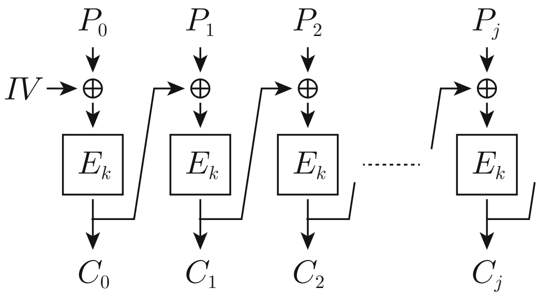

Fig. 1. CBC mode.

$$g(P_0 \oplus IV^i) \oplus g(C_0^i) = g(P_0) \oplus g(IV^i) \oplus g(C_0^i)$$

is constant, and the attacker could only recover at most one bit of secret information. However, g is nonlinear in our attack. Therefore, we can guess and determine the part of P_0 that is involved in the nonlinear term of g. More precisely, assume as above—without loss of generality—that g is nonlinear in the first $\mathcal{N}_f$ inputs and linear in the last $\mathcal{N}_\ell$ inputs, i.e.,

$$g : \mathbb{F}_2^{\mathcal{N}_f} \times \mathbb{F}_2^{\mathcal{N}_\ell} \to \mathbb{F}_2$$

such that

$$g(x, y) = f(x) \oplus \ell(x, y)$$

where f is any (nonlinear) Boolean function, and ℓ is a linear Boolean function. Consider again a plaintext $P_0 = (x, y)$ with $x \in \mathbb{F}_2^{\mathcal{N}_f}$ and $y \in \mathbb{F}_2^{\mathcal{N}_\ell}$. The corresponding ciphertext C_0^i is split as $C_0^i = (z_i, w_i)$ and the IVs as $IV^i = (u_i, v_i)$. With this notation, we can rewrite g for the first block as follows:

$$\begin{aligned} g(P_0 \oplus IV^i) &= (f(x \oplus u_i) \oplus \ell(x \oplus u_i, y \oplus v_i)), \\ g(P_0 \oplus IV^j) &= \big(f(x \oplus u_j) \oplus \ell(x \oplus u_j, y \oplus v_j)\big), \\ g(C_0^i) &= (f(z_i) \oplus \ell(z_i, w_i)), \\ g(C_0^j) &= \big(f(z_j) \oplus \ell(z_j, w_j)\big). \end{aligned}$$

Now, by using two distinct initialization vectors IV^i and IV^j

$$0 = g(P_0 \oplus IV^i) \oplus g(C_0^i) \oplus g(P_0 \oplus IV^j) \oplus g(C_0^j)$$

implies

$$f(x \oplus u_i) \oplus f(x \oplus u_j) = \ell(u_i \oplus u_j, v_i \oplus v_j) \oplus g(C_0^i) \oplus g(C_0^j). \tag{1}$$

Assuming that the left side of Eq. (1) randomly changes depending on x, that is the left $\mathcal{N}_f$ bits of P_0, we can recover one bit of information on P_0 by using two initialization vectors. Similarly, we can recover $h - 1$ bits of P_0 by using h initialization vectors. Note that we can usually efficiently recover these bits by solving linear systems if the algebraic degree of f is small [20]. We show the specific procedure for SCREAM and Midori64 in Sects. 4 and 5, respectively. The relationship among (P_0, IV, C_0) is equivalent to that among (P_i, C_{i-1}, C_i). Therefore, we can similarly guess and determine the part of P_i from C_{i-1} and C_i for any of the plaintext blocks. One interesting remark is that as long as we start to recover the message from the second block, the attack can be executed even without the knowledge of the IV.

2.2.2. *Attacks Against Other Modes*

We can execute a similar attack against the CFB, OFB, and CTR modes.

In the CFB mode, the $(j+1)$th ciphertext block C_j is encrypted as

$$C_j = E_k(C_{j-1}) \oplus P_j,$$

where the initialization vector IV is used as the input of the first block. For simplicity, let C_{-1} be IV. Then, we can recover the part of P_j from two ciphertext blocks C_{j-1} and C_j.

In the OFB mode, the $(j+1)$th ciphertext block C_j is encrypted as

$$C_j = (E_k)^j(IV) \oplus P_j,$$

where $(E_k)^j(IV)$ is j times multiple encryption. Since the nonlinear invariant property is iterative over an arbitrary number of rounds, the multiple encryption is also vulnerable against the nonlinear invariant attack. Therefore, we can recover the part of P_j from IV and C_j.

In the CTR mode, the $(j+1)$th ciphertext block C_j is encrypted as

$$C_j = E_k(IV + j) \oplus P_j.$$

Therefore, we can recover the part of P_j from $IV + j$ and C_j.

3. Finding Nonlinear Invariants for SP-Ciphers

We start by considering the very general problem of finding nonlinear invariants. Namely, given any function

$$F : \mathbb{F}_2^m \to \mathbb{F}_2^m,$$

our goal is to find a Boolean function

$$g : \mathbb{F}_2^m \to \mathbb{F}_2$$

such that

$$g(x) = g(F(x)) \oplus c \tag{2}$$

where c is a constant in $\mathbb{F}_2$. Moreover, let us denote by

$$U(F) := \{g : \mathbb{F}_2^m \to \mathbb{F}_2 \mid g(x) = g(F(x)) \oplus c\}$$

the set of all (nonlinear) invariants.

The description so far is generic in the sense that it applies to basically any block cipher. For now, and actually for the remainder of the paper, we focus on key-alternating ciphers with a round function using a layer of S-boxes and a linear layer, the so-called substitution–permutation networks (SPN).

3.1. *SPN Ciphers*

In the following, we consider the un-keyed round function only. That is to say that we ignore the key schedule and also any round constants.

For simplicity, we focus on the case of identical S-boxes, but the more general case can be handled in a very similar manner. In this section, we denote by t the number of S-boxes and by n the size of one S-box. Thus, the block size processed is $n \cdot t$ bits. With this notation, we consider one round function R of an SPN

$$R : \left(\mathbb{F}_2^n\right)^t \to \left(\mathbb{F}_2^n\right)^t$$

as consisting of a layer of S-boxes $\dot{\mathcal{S}}$ with

$$\dot{\mathcal{S}}(x_{t-1}, \ldots, x_0) = (S(x_{t-1}), \ldots, S(x_0))$$

where S is an n-bit S-box and a linear layer

$$L : \left(\mathbb{F}_2^n\right)^t \to \left(\mathbb{F}_2^n\right)^t$$

which can also be seen as

$$L : \mathbb{F}_2^{nt} \to \mathbb{F}_2^{nt}.$$

The round function R is given as the composition of the S-box layer and the linear layer, i.e.,

$$R(x) = L \circ \dot{\mathcal{S}}(x).$$

We would like to find nonlinear invariant g for R. However, computing this directly is difficult as soon as the block size is reasonable large. Recall the definition of $U(R)$, which is the set of all nonlinear invariants for R. Then it holds that

$$g \in \left(U(\dot{\mathcal{S}}) \cap U(L)\right) \subset U(R)$$

because functions that are invariant under both $\dot{\mathcal{S}}$ and L are clearly invariants for their composition $R = L \circ S$.

As we will explain next, computing parts of $U(\dot{\mathcal{S}}) \cap U(L)$ are feasible and sufficient to automatically detect the weaknesses described later in the paper.

3.1.1. *The S-box Layer*

We start by investigating the S-box layer. Given the S-box as a function

$$S : \mathbb{F}_2^n \to \mathbb{F}_2^n$$

computing $U(S)$ is feasible as long as n is only moderate in size.

Note that, for any function F, $U(F)$ is actually a subspace of Boolean functions. To see this, note that given two Boolean functions $f, g \in U(F)$, it holds

$$\begin{aligned}(f \oplus g)(x) &= f(x) \oplus g(x)\\ &= (f(F(x)) \oplus c) \oplus \big(g(F(x)) \oplus c'\big)\\ &= (f \oplus g)(F(x)) \oplus (c \oplus c')\end{aligned}$$

for any x. Thus, the sum, $f \oplus g$, is in $U(F)$ as well. Moreover, the all-constant function is in $U(F)$ for any F. Therefore, any nonlinear invariant $g_S \in U(S)$ can actually be described by a linear combination of basis elements of $U(S)$. More precisely, let $b_0, \dots, b_{d-1} : \mathbb{F}_2^n \to \mathbb{F}_2$ be a basis of $U(S)$, then any $g_S \in U(S)$ can be written as

$$g_S(x) = \bigoplus_{i=0}^{d-1} \gamma_i b_i(x)$$

for suitable coefficients γ_i in $\mathbb{F}_2$.

To identify a nonlinear invariant $g_S \in U(S)$, the idea is to consider the algebraic normal form (ANF) of g_S, that is to express g_S as

$$g_S(x) = \bigoplus_{u \in \mathbb{F}_2^n} \lambda_u x^u,$$

where $\lambda_u \in \mathbb{F}_2$ are the coefficients to be determined and x^u denotes $\prod x_i^{u_i}$. The key observation is that Eq. (2), for any fixed $x \in \mathbb{F}_2^n$, translates into one linear (or affine) equation for the coefficients λ_u, namely

$$\bigoplus_{u \in \mathbb{F}_2^n} \lambda_u (x^u \oplus S(x)^u) = c.$$

The ANF of $(x^u \oplus S(x)^u)$ is computed for all $u \in \mathbb{F}_2^n$, and we can easily solve the basis $b_0, \dots, b_{d-1} \in U(S)$ for n not too big. "Appendix A" shows the algorithm in detail. In particular, for commonly used §-box sizes of up to 8 bits, the space $U(S)$ can be computed in less than a second on a standard PC.

So far, we have considered only a single S-box, and it still needs to be discussed how those results can be translated into the knowledge of invariants for the parallel execution of S-boxes, i.e., for $\dot{\mathcal{S}}$. Again, for a layer of S-boxes $\dot{\mathcal{S}}$ computing $U(\dot{\mathcal{S}})$ directly using its ANF is (in general) too expensive. However, we can easily construct many elements in $U(\dot{\mathcal{S}})$ from elements in $U(S)$ as summarized in the following proposition.

Proposition 1. *Let $g_i \in U(S)$, for $i \in \{0, \ldots, t-1\}$ be any set of invariants for the S-box S. Then, any function of the form*

$$g_{\dot{\mathcal{S}}}(x_{t-1}, \ldots, x_0) = \bigoplus_{i=0}^{t-1} \alpha_i g_i(x_i)$$

with $\alpha_i \in \mathbb{F}_2$ is in $U(\dot{\mathcal{S}})$, that is an invariant for the entire S-box layer. The set of functions form a subspace of $U(\dot{\mathcal{S}})$ of dimension $d * t$ where d is the dimension of $U(S)$, and t is the number of parallel S-boxes.

Example 2. Let us consider $\dot{\mathcal{S}}$, where four 4-bit S-boxes are applied in parallel. Assuming that the following Boolean function

$$g_S(x[3], x[2], x[1], x[0]) = x[3]x[2] \oplus x[2] \oplus x[1] \oplus x[0]$$

is a nonlinear invariant for the 4-bit S-box,

$$g_S(x_3, x_2, \ldots, x_0) = \bigoplus_{i=0}^{3} \alpha_i (x_i[3]x_i[2] \oplus x_i[2] \oplus x_i[1] \oplus x_i[0]).$$

is nonlinear invariant for $\dot{\mathcal{S}}$ with any $(\alpha_3, \alpha_2, \alpha_1, \alpha_0) \in (\mathbb{F}_2)^4$. For example, when $(\alpha_3, \alpha_2, \alpha_1, \alpha_0) = (1, 0, 1, 0)$,

$$\begin{aligned} g_{\dot{\mathcal{S}}}(x_3, x_2, \ldots, x_0) = {} & (x_3[3]x_3[2] \oplus x_3[2] \oplus x_3[1] \oplus x_3[0]) \\ & \oplus (x_1[3]x_1[2] \oplus x_1[2] \oplus x_1[1] \oplus x_1[0]) \end{aligned}$$

is an nonlinear invariant for $\dot{\mathcal{S}}$.

We denote this subspace of invariants for $\dot{\mathcal{S}}$ by $U_\ell(\dot{\mathcal{S}})$, and $U_\ell(\dot{\mathcal{S}}) \subset U(\dot{\mathcal{S}})$.

It turns out that, in general, many more elements are contained in $U(\dot{\mathcal{S}})$ than those covered by the construction above. We decided to shift those details, which are not directly necessary for the understanding of the attacks presented in Sects. 4 and 5 to the end of the paper, in Sect. 7.

3.1.2. *The Linear Layer*

For the linear layer computing $U(L)$ using its ANF seems again difficult. But, as stated above, we focus on

$$g \in (U(L) \cap U_\ell(\dot{\mathcal{S}})) \subset (U(L) \cap U(\dot{\mathcal{S}})) \subset U(R),$$

and computing $U(L) \cap U_\ell(\dot{\mathcal{S}})$ is feasible in all practical cases.

Recall that any nonlinear invariant $g \in U(S)$ can actually be described by a linear combination of basis of $U(S)$ as

$$g_S(x) = \bigoplus_{i=0}^{d-1} \gamma_i b_i(x)$$

for suitable coefficients γ_i in $\mathbb{F}_2$.

As any f in $U_\ell(\dot{\mathcal{S}})$ is itself a direct sum of elements in $U(S)$, it can be written as

$$f(x_{t-1}, \ldots, x_0) = \bigoplus_{i=0}^{t-1} \bigoplus_{j=0}^{d-1} \beta_{i,j} b_j(x_i)$$

with $\beta_{i,j} \in \mathbb{F}_2$. Computing those coefficients $\beta_{i,j}$ can again be done by solving a linear system, as any fixed $x \in \left(\mathbb{F}_2^t\right)^n$ results in a linear equation for the coefficients by using

$$f(x) = f(L(x)).$$

As long as the dimension of $U_\ell(\dot{\mathcal{S}})$, i.e., the number of unknowns, is not too large, this again can be computed within seconds on a standard PC.

3.1.3. *Experimental Results*

When the procedure explained above was applied to the ciphers SCREAM and Midori, it instantaneously detected possible attacks. Actually, as we will explain next, there is a common structural reason why nonlinear invariant attacks are possible on those ciphers.

3.2. *Structural Weakness with Respect to Nonlinear Invariant*

Let us consider linear layers which are actually used in the LS-designs [11] (cf. Sect. 4) and also in any AES-like cipher that uses a binary diffusion matrix as a replacement for the usual MixColumns operation. Then, we consider a linear layer that can be decomposed into the parallel application of n identical $t \times t$ binary matrices M. The input for the first $t \times t$ matrix is composed of all the first output bits of the t S-boxes, and the input for the second matrix is composed of all the second output bits of the S-boxes, etc.

Here, when M is an orthogonal matrix, that is if

$$\langle x, y \rangle = \langle xM, yM \rangle \quad \forall\, x, y,$$

any quadratic nonlinear invariant for the S-box can be extended to a nonlinear invariant of the whole round function as described in Theorem 1.

Note that from a design point of view, taking M as an orthogonal matrix seems actually beneficial. Thanks to the orthogonality of M, bounds on the number of active S-boxes for differential cryptanalysis directly imply the same bounds on the number of active S-boxes for linear cryptanalysis.

Theorem 1. *For the SPN ciphers whose round function follows the construction used in LS-designs, let $M \in \mathbb{F}_2^{t\times t}$ be the binary representation of the linear layer and M is orthogonal. Assume there is a nonlinear invariant g_S for the S-box. If g_S is quadratic, then the function*

$$g(x_{t-1},\ldots,x_0) := \bigoplus_{i=0}^{t-1} g_S(x_i)$$

is a nonlinear invariant for the round function $L \circ \dot{\mathcal{S}}$.

Proof. First, due to Proposition 1, it is immediately clear that g is a nonlinear invariant for the S-box layer $\dot{\mathcal{S}}$.

Next, let us consider the linear layer L. Let $x \in (\mathbb{F}_2^n)^t$ and $y \in (\mathbb{F}_2^n)^t$ be the input and output of L, respectively. Moreover, $x_i[j]$ and $y_i[j]$ denotes the jth bit of x_i and y_i, respectively. For simplicity, let $x^T \in (\mathbb{F}_2^t)^n$ and $y^T \in (\mathbb{F}_2^t)^n$ be the transposed input and output, respectively, where $x_j^T \in \mathbb{F}_2^t$ denotes $(x_0[j], x_1[j], \ldots, x_{t-1}[j])$. Then, it holds $y_i^T = x_i^T \times M$ for all $i \in \{0,1,\ldots,n-1\}$. Since the Boolean function g_S is quadratic, the function is represented as

$$g_S(x_i) = \bigoplus_{i_1=0}^{n-1}\bigoplus_{i_2=0}^{n-1} \gamma_{i_1,i_2}(x_i[i_1] \wedge x_i[i_2]),$$

where γ_{i_1,i_2} are coefficients depending on the function g. From the inner product $\langle x_{i_1}^T, x_{i_2}^T\rangle = \bigoplus_{i=0}^{t-1} x_i[i_1] \wedge x_i[i_2]$,

$$g(x) = \bigoplus_{i=0}^{t-1} g_S(x_i) = \bigoplus_{i_1=0}^{n-1}\bigoplus_{i_2=0}^{n-1} \gamma_{i_1,i_2}\langle x_{i_1}^T, x_{i_2}^T\rangle.$$

Then,

$$g(y) = \bigoplus_{i_1=0}^{n-1}\bigoplus_{i_2=0}^{n-1} \gamma_{i_1,i_2}\langle x_{i_1}^T M, x_{i_2}^T M\rangle.$$

From the orthogonality of M,

$$\begin{aligned} g(y) &= \bigoplus_{i_1=0}^{n-1}\bigoplus_{i_2=0}^{n-1} \gamma_{i_1,i_2}\langle x_{i_1}^T, x_{i_2}^T\rangle \\ &= \bigoplus_{i=0}^{t-1} g_S(x_i) = g(x). \end{aligned}$$

Therefore, the function $g(x) = \bigoplus_{i=0}^{t-1} g_S(x_i)$ is a nonlinear invariant for L. □

Assuming that the matrix M is orthogonal, Theorem 1 shows that there is a nonlinear invariant for the round function $L \circ \mathcal{S}$ if there is a quadratic function which is nonlinear invariant for the S-box.

When M is not orthogonal, we have to execute the algorithm described in Sect. 3.1 to find nonlinear invariants for L. As far as we tried, we could not find balanced nonlinear invariant g without condition that M is orthogonal. On the other hand, if g is not balanced, we can regard that the invariant subspace [16] is a special case of the nonlinear invariant. We have not exploited the orthogonality of M in the invariant subspace attack, and it implies that there is the possibility to exist nonlinear invariant even if M is not orthogonal. We describe this observation in Sect. 7 in detail.

4. Practical Attack on SCREAM

The most interesting application of the nonlinear invariant attack is a practical attack against the authenticated encryption schemes SCREAM and iSCREAM in the nonce-respecting model. SCREAM and iSCREAM have 2^{96} and 2^{97} weak keys, respectively, and we then practically distinguish their ciphers from a random permutation. Moreover, we can extend this attack to a ciphertext-only attack.

4.1. *Specification of SCREAM*

SCREAM is an authenticated encryption scheme and a candidate of the CAESAR competition [10]. It uses the tweakable block cipher Scream, which is based on the tweakable variant of LS-designs [11].

4.1.1. *LS-Designs*

LS-designs were introduced by Grosso et al. in [11] and are used to design block ciphers. We do not refer to the design rational in this paper, and we only show the brief structure to understand this paper. The state of LS-designs is represented as an $n \times t$ matrix, where every element of the matrix is only one bit, i.e., the block length is $\mathcal{N} = n \times t$. The ith round function proceeds as follows:

1. The n-bit S-box S is applied to t columns in parallel.
2. The t-bit L-box L is applied to n rows in parallel.
3. The round constant $RC(r)$ is XORed with the state.
4. The secret tweakey $TK(\sigma)$ is XORed with the state.

Figure 2 shows the components of an LS-design. Let SB and LB be the S-box layer and L-box layer, respectively. Then, we call the composite function $(LB \circ SB)$ an LS-function. Let $x \in \mathbb{F}_2^{n \times t}$ be the state of LS-designs. Then $x[i, \star] \in \mathbb{F}_2^t$ denotes the row of index i of x, and $x[\star, j] \in \mathbb{F}_2^n$ denotes the column of index j of x. Moreover, let $x[i, j]$ be the bit in the $(i + 1)$th row and $(j + 1)$th column. The S-box S is applied to $x[\star, j]$ for all $j \in [0, t)$, and the L-box L is applied to $x[i, \star]$ for all $i \in [0, n)$.

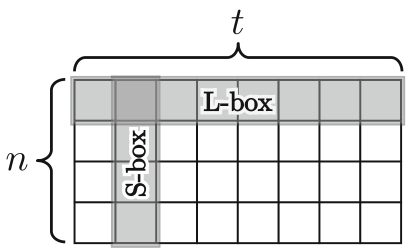

Fig. 2. The components of an LS-design.

Algorithm 1 Specification of Scream

```
1: $x \leftarrow P \oplus TK(0)$
2: for $0 < \sigma \le N_s$ do
3:     for $0 < \rho \le 2$ do
4:         $r = 2(\sigma - 1) + \rho$
5:         for $0 \le j < 16$ do
6:             $x_j^T = S[x[\star, j]]$
7:         end for
8:         $x \leftarrow x \oplus RC(r)$
9:         for $0 \le i < 8$ do
10:            $x_i = L[x[i, \star]]$
11:        end for
12:    end for
13:    $x \leftarrow x \oplus TK(\sigma)$
14: end for
15: return $x$
```

4.1.2. *Tweakable Block Cipher* Scream

Scream is based on a tweakable LS-design with an 8×16 matrix, i.e., the block length is $8 \times 16 = 128$ bits. Let $x \in \mathbb{F}_2^{8\times 16}$ be the state of Scream, then the entire algorithm is defined as Algorithm 1, where N_s be the number of steps depending on the security parameter. Here S and L denote the 8-bit S-box and 16-bit L-box, respectively, which are fully specified in "Appendix B." The round constant $RC(r)$ is defined as

$$RC(r) = 2199 \cdot r \bmod 2^{16}.$$

The binary representation of $RC(r)$ is XORed with the first row $x[0, \star]$. Scream uses a 128-bit key K and 128-bit tweak T as follows. First, the tweak is divided into 64-bit halves, i.e., $T = T_0 \| T_1$. Then, every tweakey is defined as

$$TK(\sigma = 3i) = K \oplus (T_0 \| T_1),$$
$$TK(\sigma = 3i + 1) = K \oplus (T_0 \oplus T_1 \| T_1),$$
$$TK(\sigma = 3i + 2) = K \oplus (T_1 \| T_0 \oplus T_1).$$

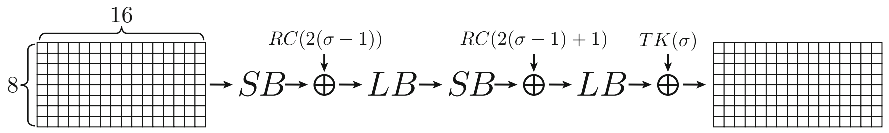

Fig. 3. The σth step function of Scream.

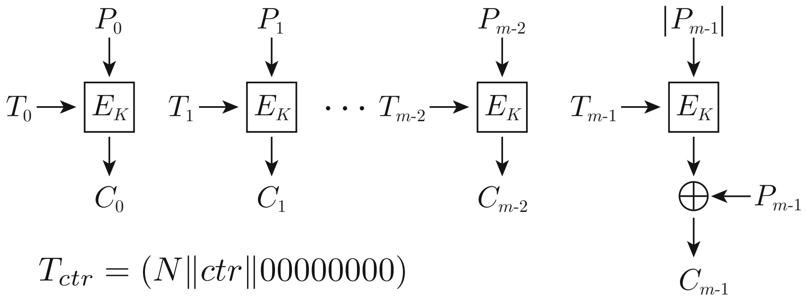

Fig. 4. Encryption of plaintext blocks.

Here, the $x[i, \star]$ contains state bits from $16i$ to $16(i+1)-1$, e.g., $x[0, \star]$ contains state bits from 0 to 15 and $x[1, \star]$ contains state bits from 16 to 31. Moreover, Fig. 3 shows the step function, where SB and LB are the S-box layer and L-box layer, respectively.

4.1.3. *Authenticated Encryption SCREAM*

The authenticated encryption scheme SCREAM uses the tweakable block cipher Scream in the TAE mode [18]. SCREAM consists of three steps: associated data processing, encryption of the plaintext block, and tag generation. Since our attack exploits encryption of the plaintext block, we explain the specification (see Fig. 4). Plaintext values are encrypted by using Scream in order to produce the ciphertext values, and all blocks use $T_{ctr} = (N \| ctr \| 00000000)$, where let n_b be the bytesize of the nonce N, and it can be chosen by the user between 1 and 15 bytes. Moreover, the counter c can take $(120 - 8n_b)$-bit value. If the last block is a partial block, its bitlength is encrypted to generate a mask, which is then truncated to the partial block size and XORed with the partial plaintext block. Therefore, the ciphertext length is the same as the plaintext length.

4.1.4. *Security Parameter*

Finally, we like to recall the security parameters of SCREAM, as described by the designers. While the nonce bytesize n_b can be chosen by the user, the designers recommend that $n_b = 11$, and we also use the recommended parameter in this paper.

SCREAM has three security parameters, i.e., lightweight security, single-key security, and related-key security. They are summarized as follows.

Lightweight security 80-bit security, with a protocol avoiding related keys. Tight parameters: 6 steps, Safe parameters: 8 steps.

Single-key security 128-bit security, with a protocol avoiding related keys. Tight parameters: 8 steps, Safe parameters: 10 steps.
Related-key security 128-bit security, with possible related keys. Tight parameters: 10 steps, Safe parameters: 12 steps.
More precisely, designers order their recommended sets of parameters as follows:

- First set of recommendations: SCREAM with 10 steps, single-key security.
- Second set of recommendations: SCREAM with 12 steps, related-key security.

4.2. *Nonlinear Invariant for* Scream

The L-box of Scream is chosen as an orthogonal matrix. Therefore, there is a nonlinear invariant for the LS-function from Theorem 1 if we can find a quadratic Boolean function $g_S : \mathbb{F}_2^8 \to \mathbb{F}_2$ which is a nonlinear invariant for the S-box S.

Let $x \in \mathbb{F}_2^8$ and $y \in \mathbb{F}_2^8$ be the input and output of the S-box S, respectively. Moreover, $x[j] \in \mathbb{F}_2$ and $y[j] \in \mathbb{F}_2$ denote the jth bits of x and y, respectively. Then, the Scream S-box has the following property

$$(x[1] \wedge x[2]) \oplus x[0] \oplus x[2] \oplus x[5] = (y[1] \wedge y[2]) \oplus y[0] \oplus y[2] \oplus y[5] \oplus 1.$$

Let $g_S : \mathbb{F}_2^8 \to \mathbb{F}_2$ be a quadratic Boolean function, where

$$g_S(x) = (x[1] \wedge x[2]) \oplus x[0] \oplus x[2] \oplus x[5].$$

Then, the function g_S is a *quadratic* nonlinear invariant for S because

$$g_S(x) \oplus g_S(S(x)) = g_S(x) \oplus g_S(x) \oplus 1 = 1.$$

Therefore, due to Theorem 1, the Boolean function

$$g(x) = \bigoplus_{j=0}^{15} g_S(x[\star, j]) = \bigoplus_{j=0}^{15} \Big(x[1, j] \wedge x[2, j] \oplus x[0, j] \oplus x[2, j] \oplus x[5, j] \Big)$$

is a nonlinear invariant for the round function of Scream. Note that this nonlinear invariant g is clearly balanced, as it is linear (and not constant) in parts of its input.

Next, we show that this Boolean function is also a nonlinear invariant for the constant addition and tweakey addition. The round constant $RC(r)$ is XORed with only $x[0, \star]$. Since $RC(r)$ linearly affects the output of the function g,

$$g(x \oplus RC(r)) = g(x) \oplus g(RC(r))$$

for any x. The tweakey $TK(\sigma)$ is defined as

$$\begin{aligned} TK(\sigma = 3i) &= K \oplus (T_0 \| T_1), \\ TK(\sigma = 3i + 1) &= K \oplus (T_0 \oplus T_1 \| T_1), \\ TK(\sigma = 3i + 2) &= K \oplus (T_1 \| T_0 \oplus T_1), \end{aligned}$$

where $T = T_0 \| T_1$. Therefore, if we restrict the key and tweak by fixing

$$K[1, \star] = K[2, \star] = 0,$$
$$T[1, \star] = T[2, \star] = T[5, \star] = T[6, \star] = 0,$$

$TK(\sigma)[1, \star]$ and $TK(\sigma)[2, \star]$ are always zero vectors. Then, since the tweakey linearly affects the output of the function g,

$$g(x \oplus TK(\sigma)) = g(y) \oplus g(TK(\sigma)),$$

and all those keys are weak. Therefore, the density of weak keys is 2^{-32}, i.e., there are 2^{96} weak keys.

Let P and C be the plaintext and ciphertext of Scream, respectively. In N_s-step Scream, the relationship between P and C is represented as

$$\begin{aligned} g(P) &= g(C) \oplus \bigoplus_{r=1}^{2N_s} g\big(RC(r)\big) \oplus \bigoplus_{\sigma=0}^{N_s} g\big(TK(\sigma)\big) \\ &= g(C) \oplus \bigoplus_{r=1}^{2N_s} g\big(RC(r)\big) \oplus g_T(N_s, T) \oplus g_K(N_s, K), \end{aligned}$$

where $g_T(N_s, T)$ and $g_K(N_s, K)$ are defined as

$$g_T(N_s, T) = \begin{cases} g(T_0 \| T_1) & N_s = 0 \bmod 6, \\ g(T_1 \| 0) & N_s = 1 \bmod 6, \\ g(0 \| T_0 \oplus T_1) & N_s = 2 \bmod 6, \\ g(T_0 \| T_0) & N_s = 3 \bmod 6, \\ g(T_1 \| T_0 \oplus T_1) & N_s = 4 \bmod 6, \\ 0 & N_s = 5 \bmod 6, \end{cases}$$

and

$$g_K(N_s, K) = \begin{cases} g(K) & N_s = 0 \bmod 2, \\ 0 & N_s = 1 \bmod 2, \end{cases}$$

respectively. When the master key belongs to the class of weak keys, $g(P) \oplus g(C) \oplus g_T(N_s, T)$ is constant for all plaintexts and a given key. When the key does not belong to the weak-key class, the probability that the output is constant is about 2^{-h+1} given h known plaintexts. Therefore, we can easily distinguish whether or not the using key belongs to the weak-key class. Note that all recommended numbers of rounds are even number. Therefore, from

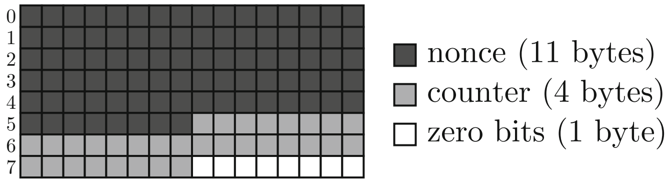

Fig. 5. Tweak mapping.

$$g(K) = g(P) \oplus g(C) \oplus \bigoplus_{r=1}^{2N_s} g\big(RC(r)\big) \oplus g_T(N_s, T),$$

we can recover one bit of information about the secret key K.

4.3. *Practical Attack on* SCREAM

4.3.1. *Known-Plaintext Attack*

We exploit the encryption step of SCREAM (see Fig. 4). The nonlinear invariant attack is a chosen-tweak attack under the weak-key setting. First, let us consider the class of weak tweaks. In the encryption step, the tweak $T_{ctr} = (N \| ctr \| 00000000)$ is used, where we assume that $n_b = 11$. Figure 5 shows the structure of T_{ctr}. From the condition of the nonlinear invariant attack, the following T_{ctr}

$$T_{ctr}[1, \star] = T_{ctr}[2, \star] = T_{ctr}[5, \star] = T_{ctr}[6, \star] = 0$$

are weak tweaks. Namely, we choose N whose 3rd, 4th, 5th, 6th, and 11th bytes are zero. Then, if the counter is less than 256, i.e., from T_0 to T_{255}, the tweak fulfills the condition. Moreover, the actual nonce fulfills the needs of the tweak if the nonce is implemented as a counter increment, which seems to occur in practice. If the master key belongs to the weak-key class, we can recover one bit of information about the secret key by using only one known plaintext. Moreover, by using h known plaintexts, the probability that the output is constant is about 2^{-h+1} when the key does not belong to weak-key class. Therefore, an attacker can distinguish whether or not the used key belongs to the weak-key class.

4.3.2. *Ciphertext-Only Message-Recovery Attack*

The interesting application of the nonlinear invariant attack is a ciphertext-only attack. This setting is more practical than the known-plaintext attack.

We focus on the procedure of the final block. The input of Scream is the bitlength of P_{m-1}, and the bitlength is encrypted to generate a mask. Then the mask is truncated to the partial block size and XORed with P_{m-1}. Therefore, the ciphertext length is the same as the plaintext length. In the ciphertext-only attack, we cannot know P_{m-1}. On the other hand, we know ciphertext C_{m-1} and the bitlength $|P_{m-1}|$ can be obtained from $|C_{m-1}|$. Therefore, we guess P_{m-1} and evaluate

$$g(|P_{m-1}|) \oplus g(P_{m-1} \oplus C_{m-1}) \oplus g_T(N_s, T),$$

and the above value is always constant for any weak tweaks T. Therefore, from two ciphertexts corresponding to the same final plaintext block encrypted by distinct tweaks, we create a linear equation as

$$g(P_{m-1} \oplus C_{m-1}) \oplus g(P_{m-1} \oplus C'_{m-1}) = g_T(N_s, T) \oplus g_T(N_s, T'). \tag{3}$$

We can compute the right side of Eq. (3). Moreover, we regard the function g as

$$g(X) = f(X) \oplus \ell(X),$$

where

$$f(X) = \bigoplus_{j=0}^{15} \Big(X[1, j] \wedge X[2, j] \Big),$$
$$\ell(X) = \bigoplus_{j=0}^{15} X[0, j] \oplus X[2, j] \oplus X[5, j].$$

Then,

$$\begin{aligned}
&g(P_{m-1} \oplus C_{m-1}) \oplus g(P_{m-1} \oplus C'_{m-1}) \\
&\quad = f(P_{m-1} \oplus C_{m-1}) \oplus f(P_{m-1} \oplus C'_{m-1}) \oplus \ell(C_{m-1}) \oplus \ell(C'_{m-1}) \\
&\quad = \bigoplus_{j=0}^{15} \Big(C_{m-1}[1, j]P_{m-1}[2, j] \oplus P_{m-1}[1, j]C_{m-1}[2, j] \oplus C_{m-1}[1, j]C_{m-1}[2, j] \Big) \\
&\qquad \bigoplus_{j=0}^{15} \Big(C'_{m-1}[1, j]P_{m-1}[2, j] \oplus P_{m-1}[1, j]C'_{m-1}[2, j] \oplus C'_{m-1}[1, j]C'_{m-1}[2, j] \Big) \\
&\qquad \oplus \ell(C_{m-1}) \oplus \ell(C'_{m-1}).
\end{aligned}$$

The equation above is actually a linear equation in 32 unknown bits, $P_{m-1}[1, j]$ and $P_{m-1}[2, j]$, as all other terms are known. Therefore, we can create h linear equations by collecting $h + 1$ ciphertexts encrypted by distinct tweaks. We can recover 32 bits, $P_{m-1}[1, j]$ and $P_{m-1}[2, j]$, by solving this system as soon as the corresponding system has full rank. Assuming that the system behaves like a randomly generated system of linear equations, we can expect that the system has full rank already when taking slightly more than 33 equations. The time complexity for solving this system is negligible.

Note that the system involves four 16-bit words, $C_{m-1}[0, j]$, $C_{m-1}[1, j]$, $C_{m-1}[2, j]$, and $C_{m-1}[5, j]$. Since the bitlength of C_{m-1} is equal to that of P_{m-1}, we cannot solve

Table 2. The success probability of recovering the correct 32 plaintext bits on SCREAM .

# nonces ($h+1$)	33	34	35	36	37	38	39	40	41	42	43
Experimental	0.289	0.571	0.762	0.885	0.942	0.976	0.991	0.995	0.998	0.999	1
Theoretical	0.289	0.578	0.770	0.880	0.939	0.969	0.984	0.992	0.996	0.998	0.999

this system if $|P_{m-1}| < 96$. Therefore, the necessary condition of this attack is $96 \leq |P_{m-1}| < 128$.

4.3.3. *Experimental Results*

In order to verify our findings and in particular to verify that the system indeed behaves like a random system of linear equations, we implemented our ciphertext-only message-recovery attack for SCREAM. In our experiment, the key is randomly chosen from the weak-key class. Moreover, we use h distinct nonces such that the corresponding tweak is weak, and collect h corresponding ciphertexts. We repeated our attack 1000 times. Table 2 summarizes the success probability of recovering the correct 32 bits. Moreover, in the table we compare the experimental success probability to the theoretically expected probability in the case of a randomly generated system of linear equations. As can be seen, the deviation of the experimental results to the theoretically expected results is very small.

4.4. *Application to iSCREAM*

The authenticated encryption scheme iSCREAM also has a similar structure to SCREAM, and the L-box of Scream is chosen as an orthogonal matrix. The full specification is referred to "Appendix C." There is a nonlinear invariant for the LS-function from Theorem 1 if we can find a quadratic Boolean function $g_S : \mathbb{F}_2^8 \to \mathbb{F}_2$ which is a nonlinear invariant for the S-box S.

Let $x \in \mathbb{F}_2^8$ and $y \in \mathbb{F}_2^8$ be the input and output of the S-box S, respectively. Moreover, $x[j] \in \mathbb{F}_2$ and $y[j] \in \mathbb{F}_2$ denote the jth bits of x and y, respectively. We first searched for bases of nonlinear invariants and found six bases of quadratic nonlinear invariants. Found six bases are shown in "Appendix A.2." Since round constants are XORed with $x[0]$, the following bases of quadratic nonlinear invariants are available.

$$g_S(x) = x[2] \oplus (x[5] \wedge x[6]) \oplus x[5] \oplus x[6] \oplus x[7],$$
$$g_S(x) = x[0] \oplus (x[4] \wedge x[5]) \oplus x[6].$$

Due to Theorem 1, the following 2 Boolean functions

$$g(x) = \bigoplus_{j=0}^{15} g_S(x[\star, j]) = \bigoplus_{j=0}^{15} \Big(x[2, j] \oplus (x[5, j] \wedge x[6, j]) \oplus x[5, j] \oplus x[6, j] \oplus x[7, j] \Big),$$
$$g(x) = \bigoplus_{j=0}^{15} g_S(x[\star, j]) = \bigoplus_{j=0}^{15} \Big(x[0, j] \oplus (x[4, j] \wedge x[5, j]) \oplus x[6, j] \Big)$$

are nonlinear invariants for the round function of iScream. Note that this nonlinear invariant g is clearly balanced, as it is linear (and not constant) in parts of its input. Moreover, all linear combinations of two bases become nonlinear invariants. Therefore, there are 3 $(= 2^2 - 1)$ nonlinear invariants for the S-box of iScream.

We omit the detailed attack procedure against authenticated encryption scheme iSCREAM because it is similar to the attack against SCREAM. Note that the number of weak keys is 2×2^{96} because we have two independent nonlinear invariants.

5. Practical Attack on Midori64

5.1. *Specification of Midori64*

Midori is a lightweight block cipher recently proposed by Banik et al. [1], which is particularly optimized for low-energy consumption. There are two versions depending on the block size; Midori64 for 64-bit block and Midori128 for 128-bit block. Both use 128-bit key. The nonlinear invariant attack can be applied to Midori64; thus, we only explain the specification of Midori64 briefly.

Midori64 adopts an SPN structure with a non-MDS matrix and a very light key schedule. The state is represented by a 4×4-nibble array. At first the plaintext is loaded to the state, then the key whitening is performed. The state is updated with a round function 16 times, and a final key whitening is performed. The resulting state is the ciphertext. The overall structure is illustrated in Fig. 6. More details on each operation will be given in the following paragraphs.

5.1.1. *Key Schedule Function*

A user-provided 128-bit key is divided into two 64-bit key states K_0 and K_1. Then, a whitening key WK and 15 round keys $RK_i, i = 0, 1. \dots, 14$ are generated as follows.

$$WK \leftarrow K_0 \oplus K_1, \qquad\qquad RK_i \leftarrow K_{i \bmod 2} \oplus \alpha_i,$$

where α_i are fixed 64-bit constants. The round constants α_i are binary for each nibble, i.e., any nibble in α_i is either `0000` or `0001`. Using such constants is beneficial to keep

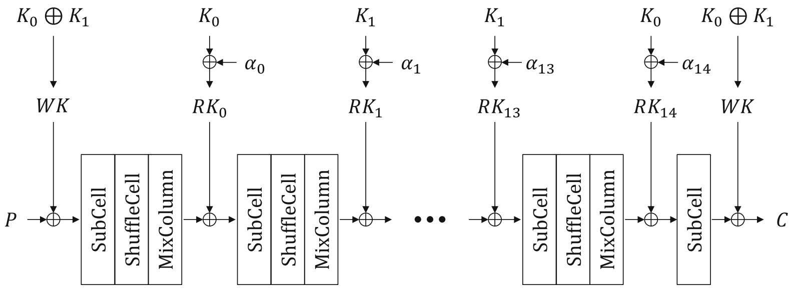

Fig. 6. Computation structure of Midori64.

Table 3. Examples of round constant α_i.

α_0	α_1	α_2	α_3	α_4	α_5
0 0 1 0	0 1 1 0	1 0 0 0	0 0 0 0	0 0 0 1	1 0 0 0
0 1 0 0	1 0 1 0	0 1 0 1	1 0 0 0	0 0 1 1	1 0 1 0
0 0 1 1	1 0 0 0	1 0 1 0	1 1 0 1	0 0 0 1	0 0 1 0
1 1 1 1	1 0 0 0	0 0 1 1	0 0 1 1	1 0 0 1	1 1 1 0

the energy consumption low. The exact values of α_i are given in Table 3 for the first 6 rounds. We refer to [1] for the complete specification.

5.1.2. *Round Function*

The round function consists of four operations: SubCell, ShuffleCell, MixColumn, and KeyAdd. Each operation is explained in the following.

SubCell The 4-bit S-box S defined below is applied to each nibble in the state.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	A	D	3	E	B	F	7	8	9	1	5	0	2	4	6

ShuffleCell Each cell of the state is permuted as ShiftRows in AES. Let s_0, s_1, s_2, s_3 be four nibbles in the first row. Let $s_4, \ldots, s_{15}$ be the other 12 nibbles similarly defined. Then, the cell permutation is specified as follows.

$$(s_0, s_1, \ldots, s_{15}) \leftarrow (s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8)$$

Note that our nonlinear invariant attack would actually work in exactly the same way for any other cell permutation as well.

MixColumn The following 4×4 *orthogonal* binary matrix M is applied to every column of the state.

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

KeyAdd The round key RK_i is XORed to the state in round i.

In the last round, only SubCell (followed by the post-whitening) is performed.

5.2. *Nonlinear Invariant for Midori64*

The matrix used in MixColumn is a binary and orthogonal matrix. Thus, Theorem 1 implies that any quadratic Boolean function $g_S : \mathbb{F}_2^4 \to \mathbb{F}_2$, which is a nonlinear invariant for the S-box S, allows us to find nonlinear invariant for the entire round function. Similarly to the previous section, we use the notation $x[j] \in \mathbb{F}_2$ and $y[j] \in \mathbb{F}_2$ to denote the jth bits of 4-bit S-box input x and 4-bit S-box output y, respectively.

We searched for g_S such that $g_S(x) = g_S(S(x))$ and found four bases of quadratic nonlinear invariants. Found four bases are shown in "Appendix A.2." Therefore, there are $15(= 2^4 - 1)$ nonlinear invariants for the S-box of Midori64. We then pick up these ones that are also nonlinear invariant for the key addition RK_i, which is computed by $RK_i \leftarrow K_{i \bmod 2} \oplus \alpha_i$. Here, α_i takes `0000` or `0001` in each nibble, i.e., the second, third, and fourth bits are always 0. Thus, we need to avoid g_S in which the first bit is included in the nonlinear component, i.e., g_S cannot involve $x[0]$ and $y[0]$ in their nonlinear component. Among the 15 choices, only one can satisfy this condition. The picked S-box property of Midori64 is as follows.

$$x[0] \oplus x[1] \oplus (x[2] \wedge x[3]) \oplus x[2] = y[0] \oplus y[1] \oplus (y[2] \wedge y[3]) \oplus y[2].$$

Then, the following $g_S : \mathbb{F}_2^4 \rightarrow \mathbb{F}$ is a nonlinear invariant for S;

$$g_S(x) = x[0] \oplus x[1] \oplus (x[2] \wedge x[3]) \oplus x[2].$$

Here, ShuffleCell does not affect the nonlinear invariant. Therefore, from Theorem 1, the following Boolean function

$$g(x) = \bigoplus_{i=0}^{15} g_S(x_i)$$

is a nonlinear invariant for the round function of Midori64. Note, as for SCREAM, the Boolean function g is actually balanced.

5.3. *Distinguishing Attack*

As mentioned in Sect. 2, the simple distinguishing attack can be mounted against a weak key. Let $\ell : \mathbb{F}_2^{64} \rightarrow \mathbb{F}_2$ be a linear part of g, namely $\ell(x) = \bigoplus_{i=0}^{15}(x_i[2] \oplus x_i[1] \oplus x_i[0])$. We have $g(P) \oplus g(C) = c$, where c is a linear part of the values injected to round function during the encryption process;

$$\begin{aligned} c &= \ell(WK) \oplus \ell(RK_0) \oplus \ell(RK_1) \oplus \cdots \oplus \ell(RK_{14}) \oplus \ell(WK), \\ &= \ell(RK_0) \oplus \ell(RK_1) \oplus \cdots \oplus \ell(RK_{14}). \end{aligned}$$

Given $RK_i = K_{i \bmod 2} \oplus \alpha_i$, the above equation is further converted as

$$c = \ell(K_1) \oplus \ell(\alpha_0) \oplus \ell(\alpha_1) \oplus \cdots \oplus \ell(\alpha_{14}).$$

Table 4. The success probability of recovering the correct 32 bits on Midori64-CBC.

# blocks $(h+1)$	33	34	35	36	37	38	39	40	41	42	43
Experimental	0.279	0.574	0.753	0.883	0.931	0.968	0.988	0.991	0.999	0.997	1
Theoretical	0.289	0.578	0.770	0.880	0.939	0.969	0.984	0.992	0.996	0.998	0.999

As $\alpha_i[2] = \alpha_i[1] = 0$ for any i, it can be simply written as

$$c = \ell(K_1) \oplus \bigoplus_{i=0}^{14} \bigoplus_{j=0}^{15} \alpha_{i,j},$$

where $\alpha_{i,j}$ is the jth nibble of α_i. We confirmed that the total number of 1 in all α_i is even, thus $\bigoplus_{i=0}^{14} \bigoplus_{j=0}^{15} \alpha_{i,j} = 0$. In the end, $g(P) \oplus g(C) = \ell(K_1)$ always holds for Midori64, while this holds with probability $1/2$ for a random permutation.

5.4. *Experimental Results*

As mentioned in Sect. 2, the above property can reveal 32 bits (the two most significant bits from each nibble) of an unknown plaintext block in the weak-key setting when Midori64 is used in well-known block cipher modes.

We implemented our ciphertext-only message-recovery attack for Midori64 in the CBC mode. In our experiment, the key and IV are chosen uniformly at random from the weak-key space and the entire IV space. We also choose a 64-bit plaintext block P, uniformly at random, and assume that P is iterated over h blocks, where $33 \leq h \leq 43$. We executed our attack of repeating 1000 times, and Table 4 summarizes the success probability of recovering the correct 32 bits. Similarly to the case of SCREAM the system of equations behaves very much like a random system of equation in the sense that the probability that it has full rank is very close to the corresponding probability for a random system with the same dimensions.

6. Extending Weak-Key Classes for Nonlinear Invariant

Recall Sect. 2.1. Without loss of generality, input bits are separated into two parts as $g : (\mathbb{F}_2^{\mathcal{N}_f} \times \mathbb{F}_2^{\mathcal{N}_\ell}) \to \mathbb{F}_2$, where the last $\mathcal{N}_\ell$ bits are only linearly involved in the nonlinear invariant g. Then

$$g(x, y) = f(x) \oplus \ell(x, y)$$

where f is the nonlinear part of g, and ℓ is the linear part of g. As the class of weak keys, we considered a round key such that its first $\mathcal{N}_f$ bits are zero as $(0, k')$ because the nonlinear invariant is preserved for such a weak key.

Now we extend a class of weak keys for the nonlinear invariant, where a nonzero constant is accepted even if it is XORed to the nonlinear term of g. We explain such a case by using a simple example. Assume that the following Boolean function

$$g_S(x[3], x[2], x[1], x[0]) = x[0]x[3] \oplus x[0] \oplus x[2]x[3] \oplus x[3]$$

is nonlinear invariant for a 4-bit S-box. In the class of weak keys described above, nonzero constants are only accepted for $x[1]$ because $x[0]$, $x[2]$, and $x[3]$ are nonlinearly involved in g_S. Namely, the class of weak keys is $\{\texttt{0000}, \texttt{0010}\}$. Here we observe that when constant bits corresponding to $x[0]$ and $x[2]$ are 1 at the same time, i.e., `0101`, the nonlinear invariant is preserved because

$$\begin{aligned} g_S(x \oplus \texttt{0101}) &= (x[0] \oplus 1)x[3] \oplus (x[0] \oplus 1) \oplus (x[2] \oplus 1)x[3] \oplus x[3], \\ &= x[0]x[3] \oplus x[3] \oplus x[0] \oplus 1 \oplus x[2]x[3] \oplus x[3] \oplus x[3], \\ &= x[0]x[3] \oplus x[0] \oplus x[2]x[3] \oplus x[3] \oplus 1 = g_S(x) \oplus 1. \end{aligned}$$

Therefore, the weak class is extended to $\{\texttt{0000}, \texttt{0010}, \texttt{0101}, \texttt{0111}\}$ from $\{\texttt{0000}, \texttt{0010}\}$.

As another example, assume that a following Boolean function

$$g_S(x[3], x[2], x[1], x[0]) = x[0]x[1] \oplus x[1]x[2] \oplus x[2]x[3] \oplus x[3]x[0]$$

is nonlinear invariant for a 4-bit S-box. Since all bits are nonlinearly involved, the weak key can only be `0000` by the approach in the previous section. However, a constant `1111` is also weak because $g_S(x \oplus \texttt{1111}) = g_S(x)$. Therefore, the weak class is extended to $\{\texttt{0000}, \texttt{1111}\}$.

In both example, new weak keys having `1` in quadratic terms are detected, and they expand the attack targets that are vulnerable against the nonlinear invariant attack.

6.1. *Application to Modified* ***Midori64***

To demonstrate the effectiveness of the new class of weak keys for the nonlinear invariant attack, we first modify the round constant of Midori64 so that the attack in Sect. 5 is prevented and then show that even this modified version can be attacked by using the extended weak-key class.

Recall that g_S used to attack Midori64 in Sect. 5 is $g_S(x) = x[0] \oplus x[1] \oplus (x[2] \wedge x[3]) \oplus x[2]$; thus, the weak keys are $\{\texttt{0000, 0001, 0010, 0011}\}$. Hence, we replace the generation of the ith round key by $RK_i \leftarrow K_{i \bmod 2} \oplus \alpha_i$ in the original Midori64 with $RK_i \leftarrow K_{i \bmod 2} \oplus (\texttt{0x5} \cdot \alpha_i)$. In other words, any nibble in α_i is either `0000` or `0101` in the modified Midori64. Obviously, the usage of `0101` prevents the attack in Sect. 5.

On the other hand, the S-box has another nonlinear invariant

$$g_S(x) = x[0]x[3] \oplus x[0] \oplus x[2]x[3] \oplus x[3],$$

where `0101` belongs to the (extended) class of weak keys of this $g_S(x)$. Since ShuffleCell does not affect the nonlinear invariant, the following Boolean function

$$g(x) = \bigoplus_{i=0}^{15} g_S(x_i)$$

is a nonlinear invariant for the round function of the modified Midori64 from Theorem 1. Note that this Boolean function g is also balanced.

In summary, we can attack modified Midori64 by exploiting the new class of weak keys for the nonlinear invariant attack. Note that K_i should be chosen from weak keys, where key bits corresponding to $x[3]$ in every S-box must be 0 and key bits corresponding to $(x[0], x[2])$ in every S-box must be `00` or `11`. As a result, the density of weak keys is 2^{-64}.

7. Extensions and Future Work

In this section, we outline some extensions to the previously described attacks. Furthermore, we give some additional insights into the structure of nonlinear invariants in general. Finally, we explain how invariant subspace attacks can be seen as a special, *chosen plaintext*, variant of nonlinear invariant attacks. It is important to point out that none of the observations in this section lead to any attacks. But we feel that those explanations provide good starting points for future investigations.

7.1. *More General Nonlinear Invariant*

We continue to use the notation that we fixed in Sect. 3. First recall Proposition 1, that allowed to construct nonlinear invariants for the whole S-box layer by linearly combining nonlinear invariants for each single S-box. This proposition can actually be easily extended. Instead of only linearly combining the nonlinear invariants for each S-box, any combination by using an arbitrary Boolean function results in an invariant for the whole S-box layer as well. The following proposition summarizes this observation.

Proposition 2. *Given any Boolean function $f : \mathbb{F}_2^t \to \mathbb{F}_2$ and t elements*

$$g_0, \dots, g_{t-1} : \mathbb{F}_2^n \to \mathbb{F}_2$$

from $U(S)$ it holds that

$$\begin{aligned} g_{\dot{\mathcal{S}}} &: \left(\mathbb{F}_2^n\right)^t \to \mathbb{F}_2 \\ g_{\dot{\mathcal{S}}}(x_{t-1}, \dots, x_0) &= f(g_{t-1}(x_{t-1}), \dots, g_0(x_0)) \end{aligned}$$

is an element of $U(\dot{\mathcal{S}})$

Note that the special case of f being linear actually corresponds to the choice made in Proposition 1.

While this generalization potentially allows a much larger variety of invariants, and therefore potential attacks, we like to mention that the restriction made in Proposition 1 has two crucial advantages. First, the choice is small enough, so that it can be handled exhaustively and second, the invariants generated by Proposition 1 are usually balanced when g_i is balanced, while this is not necessarily the case for the generalization.

At first sight, one might be tempted to assume that the above construction actually covers all invariants for the S-box layer. However, in general, this is not the case.

One counter-example, that is a nonlinear invariant not covered by this construction, can easily be identified as follows: For simplicity, consider an S-box layer consisting of two identical n-bit S-boxes only. If the two inputs to those two S-boxes are equal, so are the outputs. Thus, the function

$$g_{\dot{\mathcal{S}}} : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2$$
$$g_{\dot{\mathcal{S}}}(x, y) = \begin{cases} 1 & \text{if} \quad x = y \\ 0 & \text{else} \end{cases}$$

is an nonlinear invariant of the S-box layer as

$$g_{\dot{\mathcal{S}}}(x, y) = 1 \Leftrightarrow x = y \Leftrightarrow S(x) = S(y) \Leftrightarrow g_{\dot{\mathcal{S}}}(S(x), S(y)) = 1.$$

Moreover, this nonlinear invariant can certainly not be generated by Proposition 2.

7.2. *Cycle Structure*

Actually, there is a nice, and potentially applicable way, of describing all nonlinear invariants for a given permutation F by considering its cycles. Recall that a cycle of F being a set

$$\mathbb{C}_x := \{F^i(x) \mid i \in \mathbb{N}\}$$

for a value $x \in \mathbb{F}_2^n$. Actually, one can show that a mapping g is contained in $U(F)$ if and only if g is either constant on all cycles of F or alternating along the cycles of F. The later case corresponds to nonlinear invariants such that

$$g(x) + g(F(x)) = 1.$$

This is because $g(x) = g(F(x))$ implies

$$g(x) = g(F(x)) = g(F(F(x))) = \cdots = g(F^i(x)).$$

Thus, looking at the cycle structure of F, we can assign to each cycle one value the function g should evaluate to on this cycle. That view point also shows that the number of invariant functions g is equal to

$$|U(F)| = 2^{(\text{\# cycles of } F)},$$

in the case where there exists at least one cycle of odd length or

$$|U(F)| = 2^{(\text{\# cycles of } F)+1},$$

in the case where all cycles of F have even length. This perspective allows to actually compute a basis of $U(F)$ very efficiently, and "Appendix A.1" shows its sage code. Consider, for simplicity, the case were not all cycles are of even length. Then, a basis of $U(F)$ clearly consists of the set of all indicator functions of $\mathbb{C}_x$, i.e.,

$$U(F) = \text{span}\{\delta_{\mathbb{C}_a} \mid a \in \mathbb{F}_2^n\}.$$

Here, for a subset $\mathbb{V} \subseteq \mathbb{F}_2^n$, the function $\delta_{\mathbb{V}}$ denotes the indicator function of the set $\mathbb{V}$, i.e.,

$$\delta_{\mathbb{V}}(x) = \begin{cases} 1 & \text{if} \quad x \in \mathbb{V} \\ 0 & \text{else} \end{cases}$$

Example 3. Consider the function $F : \mathbb{F}_2^2 \to \mathbb{F}_2^2$ with

x	0	1	2	3
$F(x)$	1	2	0	3

The cycle composition of F is

$$(0, 1, 2)(3).$$

Thus, we have two cycles of odd length. Following the above, any nonlinear invariant of F is constant on those cycles. In this case, we have the following invariants

$$g_1(x) = \delta_{\{0,1,2\}}(x)$$
$$g_2(x) = \delta_{\{3\}}(x)$$

or, more explicitly

x	0	1	2	3
$g_1(x)$	1	1	1	0

and

x	0	1	2	3
$g_2(x)$	0	0	0	1

together with the trivial invariants, that is the identical zero or identical one functions. So in total F has 4 invariants.

7.3. *Relation to Invariant Subspace Attack*

Along the same lines, one can also see the invariant subspace attack as a special case of a nonlinear invariant. Recall that a subspace $\mathbb{V} \subseteq \mathbb{F}_2^n$ is called invariant under (a block cipher) F if

$$F(x) \in \mathbb{V}$$

for all $x \in \mathbb{V}$. That is, the set $\mathbb{V}$ is mapped to itself by the function F. Note that the complement $\bar{\mathbb{V}}$ is also mapped to itself because the function F is permutation. This means nothing else than that the nonlinear Boolean function $\delta_{\mathbb{V}}(x)$ is a nonlinear invariant for F as

$$\delta_{\mathbb{V}}(x) = 1 \Leftrightarrow x \in \mathbb{V} \Leftrightarrow F(x) \in \mathbb{V} \Leftrightarrow \delta_{\mathbb{V}}(F(x)) = 1,$$
$$\delta_{\mathbb{V}}(x) = 0 \Leftrightarrow x \in \bar{\mathbb{V}} \Leftrightarrow F(x) \in \bar{\mathbb{V}} \Leftrightarrow \delta_{\mathbb{V}}(F(x)) = 0.$$

In other words, invariant subspace attacks are nonlinear invariant attacks where the support of the nonlinear invariant is a subspace of $\mathbb{F}_2^n$. And as such, nonlinear invariant attacks could be called *invariant set attacks*, as the function g splits in the inputs into two sets, the support of g and its complement, that are invariant under F.

7.4. *Further Research*

Other interesting directions for further research include the generalization of the nonlinear invariant to the case where one does not consider the same function g in every round, but rather a sequence of functions that can be chained together. In fact, we also found quadratic Boolean function $g'_S : \mathbb{F}_2^4 \to \mathbb{F}_2$ such that $g_S(x) = g'_S(S(x))$ for Midori64. Owing to the involution property of the S-box, $g_S(x) = g'_S(S(x))$ always implies $g'_S(x) = g_S(S(x))$. Combining with the alternative use of K_0 and K_1 in the key schedule, such g_S, g'_S may be exploited in the attack. Unfortunately, since such Boolean functions are not nonlinear invariant for the constant addition in Midori64, we cannot exploit them in real cryptanalysis. However, it is clearly worth discussing this extension. And last but not least, even so it seems notoriously difficult, it would be nice to be able to use a statistical variant of the attack described here, i.e., consider nonlinear functions such that $g(F(x)) = g(x)$ for many—but not necessarily for all—inputs x.

7.5. *Countermeasure against Nonlinear Invariant Attacks*

Designers in future might be asked to argue that a new cipher is resistant to nonlinear invariant attacks. However, the number of potential nonlinear invariants for a given round function might be huge. It is therefore of interest to have at hand simple and/or efficiently verifiable conditions that allow to argue about the resistance. Intuitively, for strong resistance, the use of some complicated round constant is helpful. As a followup work of our conference version, Beierle et al. proposed a way how to choose round constants for obtaining provable security [3]. On the other hand, currently exploited nonlinear invariants are limited to quadratic ones, and adopting a binary orthogonal matrix is a necessary condition to overcome the linear layer. It is difficult to avoid quadratic nonlinear invariant when 4-bit S-boxes are used. Therefore, it is a reasonable countermeasure to avoid the use of binary orthogonal matrices. Note that orthogonality is often useful to guarantee the security against both differential and linear attacks. If designers want to use an orthogonal matrix, non-binary ones are recommended. Moreover, when 8-bit S-boxes are used, it is easy to choose S-boxes that do not have quadratic nonlinear invariants.

Acknowledgements

We would like to thank Christof Beierle for development of sage code to detect bases of nonlinear invariants.

A. Algorithm to Solve Basis of $U(S)$

Let $g_S \in U(S)$, where S denotes an n-bit S-box. Then the algebraic normal form (ANF) is expressed as

$$g_S(x) = \bigoplus_{u \in \mathbb{F}_2^n} \lambda_u x^u,$$

where $\lambda_u \in \mathbb{F}_2$ are the coefficients to be determined and x^u denotes $\prod x_i^{u_i}$. From the definition of the nonlinear invariant, the following equation

$$\bigoplus_{u \in \mathbb{F}_2^n} \lambda_u x^u = \bigoplus_{u \in \mathbb{F}_2^n} \lambda_u S(x)^u$$

holds for any $x \in \mathbb{F}_2^n$. We define $g_{S,u}(x)$ as

$$g_{S,u}(x) = x^u \oplus S(x)^u = \bigoplus_{v \in \mathbb{F}_2^n} \lambda_{u,v} x^v.$$

Our goal is to find the ANF coefficient λ_u such that

$$\bigoplus_{u \in \mathbb{F}_2^n} \lambda_u \cdot g_{S,u}(x) = \bigoplus_{u \in \mathbb{F}_2^n} \lambda_u \cdot \bigoplus_{v \in \mathbb{F}_2^n} \lambda_{u,v} x^v$$

is constant for any $x \in \mathbb{F}_2^n$. Then, we prepare a matrix $[I \| M]$, where I is a $(2^n \times 2^n)$ identical matrix and coefficients of M are computed as

$$M[u, v] = \lambda_{u,v}$$

Then, by some computation similar to Gaussian elimination, we compute matrix $M' = [M'_1 \| M'_2]$. If rows of M'_2 are $[0, 0, \ldots, 0]$ or $[1, 0, 0, \ldots, 0]$, the corresponding row of M'_1 is the basis of $U(S)$. In particular, for commonly used S-box sizes of up to 8 bit, the space $U(S)$ can be computed in less than a second on a standard PC.

A.1. *Sage Code to Detect Basis of Nonlinear Invariants*

As shown in Sect. 7, nonlinear invariants are also found by analyzing the cycle structure. The following sage code detects the basis of nonlinear invariant up to degree d.

```
from sage.crypto.boolean_function import BooleanFunction

# returns the Hamming Weight of v
def hamming_weight(v):
  w = 0
  for i in v:
    if (i != 0):
      w = w+1
  return w

# returns 1 if the vector a is smaller than or equal to v
in the lex def is_smaller_equal(a,v):
  r = 1
  for i in range(len(v)):
    if (a[i] > v[i]):
      r = 0
  return r

# input: the subspace of anf coefficient vectors
# computes the basis of boolean functions from anf
coefficient vector space def basis_from_anfspace(V):
  BV = V.basis()
  bitlength_S = int(log(len(BV[0]),2))
  R = []
  W = VectorSpace(GF(2),bitlength_S)
  for v in BV:
    b_value_vector = []
    for k in range(len(W)):
      value = 0
      for i in range(len(BV[0])):
        if (is_smaller_equal(W[i], W[k])):
          value = value + v[i]
      value = (value % 2)
      b_value_vector.append(value)
    R.append(BooleanFunction(b_value_vector))
  return R

# returns a basis of all invariants (g(x) = g(S(x))) for the
```

```
 Sbox S.
# x0 will be the LSB (rightmost bit of input)
def basis_invariants(S):
  all_even = 1
  R = []
  # represent the S-box in cycle notation (all elements
 are +1)
  Cl = Permutation([x+1 for x in S]).to_cycles()
  for L in Cl:
    if (len(L) % 2 == 1):
      all_even = 0
    # initialze list of len(S)
    % that will identify the boolean function in the basis
    B = [0] * len(S)
    for l in L:
      B[l-1] = 1
    R.append(BooleanFunction(B))
  # if all cycles are even,
  # there exists one more basis element which is alternating
 on the cycles
  if (all_even):
    B = [0] * len(S)
    for L in Cl:
      value = 0
      for i in range(len(L)):
        B[L[i]-1] = value
        value = (value + 1) % 2
    R.append(BooleanFunction(B))
  return R

# returns all boolean functions given by basis B_S
def all_from_basis(B_S):
    L = []
    bitlength_S = B_S[0].nvariables()
    V = VectorSpace(GF(2),len(B_S))
    for v in V:
        B = BooleanFunction([0] * 2^bitlength_S)
        for i in range(len(B_S)):
            if (v[i] == 1):
                B = B + B_S[i]
        L.append(B)
    return L

# return coefficient vector of ANF polynomial of boolean
 function b.
# The coefficient a_u is at the same position as the vector
 u in the lex def coefficient_vector_ANF(b):
    V = VectorSpace(GF(2),b.nvariables())
    # Mobius transform
    r = vector(GF(2),2**(b.nvariables()))
    for i in range(len(V)):
        value = 0
        for a in V:
            if (is_smaller_equal(a,V[i])):
                value = value + b(list(a))
        r[i] = (value % 2)
    return r

# returns all basis of invariants for S upto degree d
def all_invariants_upto_degree(S, d):
    B_S = basis_invariants(S)
    bitlength_S = B_S[0].nvariables()
    B = []
    W = VectorSpace(GF(2),bitlength_S)
    for b in B_S:
        B.append(coefficient_vector_ANF(b))
    VS = span(B, GF(2))
    C = []
    for i in range(2**bitlength_S):
        if (hamming_weight(W[i]) <= d):
            v = [0] * i
            v = v + [1]
            v = v + ([0] * ((2**bitlength_S)-1-i))
            C.append(vector(v))
    V = span(C,GF(2))
    R = V.intersection(VS)
    return basis_from_anfspace(R)
```

A.2. *List of Bases of Nonlinear Invariants*

Let $x \in \mathbb{F}_2^n$ and $y \in \mathbb{F}_2^n$ be the input and output of an S-box, respectively. Moreover, x_i and y_i denote the ith bits of x and y, where x_0 and x_{n-1} are LSB and MSB, respectively.

A.2.1. *Scream S-Box*

Table 5 in "Appendix B" shows the table representation of Scream S-box. We executed the algorithm shown in "Appendix A.1." As a result, we found that the dimension of $U(S)$ is 12 and the following one base

$$g(x) = x_0 \oplus x_1x_2 \oplus x_2 \oplus x_5$$

is only quadratic nonlinear invariant.

A.2.2. *iScream S-Box*

Table 6 in "Appendix C" shows the table representation of iScream S-box. We executed the algorithm shown in "Appendix A.1." As a result, we found that the dimension of $U(S)$ is 135. Moreover, the following 6 Boolean functions

$$\begin{aligned}
g_1(x) &= x_0 \oplus x_4x_5 \oplus x_6, \\
g_2(x) &= x_2 \oplus x_5x_6 \oplus x_5 \oplus x_6 \oplus x_7, \\
g_3(x) &= x_0x_6 \oplus x_1x_7 \oplus x_1 \oplus x_2x_5 \oplus x_2x_7 \oplus x_3x_4 \oplus x_3x_5 \oplus x_3x_6 \\
&\quad \oplus x_3x_7 \oplus x_4x_7 \oplus x_5x_7 \oplus x_5 \oplus x_6x_7 \oplus x_7, \\
g_4(x) &= x_0x_5 \oplus x_0x_6 \oplus x_0x_7 \oplus x_1x_6 \oplus x_2x_4 \oplus x_2x_6 \oplus x_2x_7 \oplus x_3x_4
\end{aligned}$$

$$\oplus x_3x_5 \oplus x_3 \oplus x_4x_5 \oplus x_4x_7 \oplus x_5x_7 \oplus x_5,$$
$$g_5(x) = x_0x_4 \oplus x_0x_5 \oplus x_0x_6 \oplus x_1x_5 \oplus x_2x_4 \oplus x_2x_6 \oplus x_3x_7 \oplus x_5x_6,$$
$$g_6(x) = x_0x_5 \oplus x_0x_6 \oplus x_1x_4 \oplus x_1x_5 \oplus x_1x_6 \oplus x_2x_5 \oplus x_2x_6 \oplus x_2x_7$$
$$\oplus x_3x_4 \oplus x_3x_5 \oplus x_3x_7 \oplus x_4x_5 \oplus x_6x_7 \oplus x_7,$$

are the basis of quadratic nonlinear invariants.

A.2.3. *Midori64 S-Box*

We executed the algorithm shown in "Appendix A.1." As a result, we found that the dimension of $U(S)$ is 9. Moreover, the following 4 Boolean functions

$$g_1(x) = x_0x_3 \oplus x_0 \oplus x_2x_3 \oplus x_3,$$
$$g_2(x) = x_0x_3 \oplus x_1 \oplus x_2 \oplus x_3,$$
$$g_3(x) = x_0x_1 \oplus x_1x_2 \oplus x_2x_3 \oplus x_2,$$
$$g_4(x) = x_0x_2 \oplus x_0x_3,$$

are the basis of quadratic nonlinear invariants. To avoid involving x_0 to quadratic term, quadratic Boolean function $g_1 \oplus g_2$ is used to analyze full Midori64.

B. Specification of S-Box and L-Box of Scream

Table 5 shows the table representation of Scream S-box. Let M be the binary representation of the L-box. Then M is defined as

$$M = \begin{pmatrix}
0&0&1&1&1&0&1&0&0&0&0&1&1&1&0&0\\
1&0&0&1&0&1&0&1&0&1&0&0&1&0&1&0\\
1&1&0&0&1&1&0&1&1&1&0&1&1&1&1&0\\
1&0&0&0&0&0&1&1&0&1&1&0&1&0&0&1\\
1&0&1&1&0&1&1&0&1&1&1&0&1&0&1&1\\
0&0&0&0&0&1&1&1&0&1&0&1&1&1&0&0\\
0&0&1&0&0&1&0&0&1&0&1&0&0&1&1&1\\
1&0&1&0&0&1&0&1&0&1&1&1&1&1&1&1\\
1&1&0&1&0&0&1&0&0&1&1&0&0&0&1&0\\
1&1&1&1&0&1&0&1&1&0&0&0&1&1&1&1\\
1&1&0&0&1&1&0&0&1&0&0&0&0&1&0&1\\
0&0&1&0&1&1&1&0&1&1&1&1&1&1&1&0\\
0&1&0&0&1&0&0&0&1&1&1&0&0&1&1&0\\
1&1&1&1&0&1&1&0&0&1&0&1&1&1&1&0\\
1&1&0&1&1&0&0&0&0&0&0&0&1&1&1&0\\
1&0&0&0&1&1&0&1&0&1&0&1&0&0&0&1
\end{pmatrix}.$$

Table 5. Table representation of Scream S-box.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	20	8D	B2	DA	33	35	A6	FF	7A	52	6A	C6	A4	A8	51	23
10	A2	96	30	AB	C8	17	14	9E	E8	F3	F8	DD	85	E2	4B	D8
20	6C	01	0E	3D	B6	39	4A	83	6F	AA	86	6E	68	40	98	5F
30	37	13	05	87	04	82	31	89	24	38	9D	54	22	7B	63	BD
40	75	2C	47	E9	C2	60	43	AC	57	A1	1F	27	E7	AD	5C	D2
50	0F	77	FD	08	79	3A	49	5D	ED	90	65	7C	56	4F	2E	69
60	CD	44	3F	62	5B	88	6B	C4	5E	2D	67	0B	9F	21	29	2A
70	D6	7E	74	E0	41	73	50	76	55	97	3C	09	7D	5A	92	70
80	84	B9	26	34	1D	81	32	2B	36	64	AE	C0	00	EE	8F	A7
90	BE	58	DC	7F	EC	9B	78	10	CC	2F	94	F1	3B	9C	6D	16
A0	48	B5	CA	11	FA	0D	8E	07	B1	0C	12	28	4C	46	F4	8B
B0	A9	CF	BB	03	A0	FC	EF	25	80	F6	B3	BA	3E	F7	D5	91
C0	C3	8A	C1	45	DE	66	F5	0A	C9	15	D9	A3	61	99	B0	E4
D0	D1	FB	D3	4E	BF	D4	D7	71	CB	1E	DB	02	1A	93	EA	C5
E0	EB	72	F9	1C	E5	CE	4D	F2	42	19	E1	DF	59	95	B7	8C
F0	9A	F0	18	E6	C7	AF	BC	B8	E3	1B	D0	A5	53	B4	06	FE

Table 6. Table representation of iScream S-box.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	85	65	D2	5B	FF	7A	CE	4D	E2	2C	36	92	15	BD	AD
10	57	F3	37	2D	88	0D	AC	BC	18	9F	7E	CA	41	EE	61	D6
20	59	EC	78	D4	47	F9	26	A3	90	8B	BF	30	0A	13	6F	C0
30	2B	AE	91	8A	D8	74	0B	12	CC	63	FD	43	B2	3D	E8	5D
40	B6	1C	83	3B	C8	45	9D	24	52	DD	E4	F4	AB	08	77	6D
50	F5	E5	48	C5	6C	76	BA	10	99	20	A7	04	87	3F	D0	5F
60	A5	1E	9B	39	B0	02	EA	67	C6	DF	71	F6	54	4F	8D	2E
70	E7	6A	C7	DE	35	97	55	4E	22	81	06	B4	7C	FB	1A	A1
80	D5	79	FC	42	84	01	E9	5C	14	93	33	29	C1	6E	A8	B8
90	28	32	0C	89	B9	A9	D9	75	ED	58	CD	62	F8	46	9E	19
A0	CB	7F	A2	27	D7	60	FE	5A	8E	95	E3	4C	16	0F	31	BE
B0	64	D3	3C	B3	7B	CF	40	EF	8F	94	56	F2	17	0E	AF	2A
C0	2F	8C	F1	E1	DC	53	68	72	44	C9	1B	A0	38	9A	07	B5
D0	5E	D1	03	B1	23	80	1F	A4	34	96	E0	F0	C4	49	73	69
E0	DA	C3	09	AA	4A	51	F7	70	3E	86	66	EB	21	98	1D	B7
F0	DB	C2	BB	11	4B	50	6B	E6	9C	25	FA	7D	82	3A	A6	05

C. Specification of S-Box and L-Box of iScream

Table 6 shows the table representation of iScream S-box. Let M be the binary representation of the L-box. Then M is defined as

$$M = \begin{pmatrix} 0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\ 1&0&0&0&0&0&1&1&0&0&1&1&0&0&1&1\\ 1&0&0&0&0&1&0&1&0&1&0&1&0&1&0&1\\ 1&0&0&0&1&0&0&1&1&0&0&1&1&0&0&1\\ 1&0&0&1&0&0&0&1&0&1&1&0&0&1&1&0\\ 1&0&1&0&0&0&0&1&1&0&1&0&1&0&1&0\\ 1&1&0&0&0&0&0&1&1&1&0&0&1&1&0&0\\ 1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\ 1&0&0&1&0&1&1&0&0&0&0&0&0&1&1&1\\ 1&0&1&0&1&0&1&0&0&0&0&0&1&0&1&1\\ 1&1&0&0&1&1&0&0&0&0&0&0&1&1&0&1\\ 1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&0\\ 1&0&0&1&0&1&1&0&0&1&1&1&0&0&0&0\\ 1&0&1&0&1&0&1&0&1&0&1&1&0&0&0&0\\ 1&1&0&0&1&1&0&0&1&1&0&1&0&0&0&0\\ 1&1&1&1&0&0&0&0&1&1&1&0&0&0&0&0 \end{pmatrix}.$$

References

[1] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, F. Regazzoni, Midori: a block cipher for low energy. in T. Iwata, J.H.Cheon, (eds), *ASIACRYPT Part II*. LNCS, vol. 9453 (Springer, 2015), pp. 411–436

[2] E. Biham, A. Biryukov, A. Shamir, Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials, in J. Stern, editor, *EUROCRYPT*, LNCS, vol. 1592 (Springer, 1999), pp. 12–23

[3] C. Beierle, A. Canteaut, G. Leander, Y. Rotella, Proving resistance against invariant attacks: how to choose the round constants, in J. Katz, H. Shacham, editors, *CRYPTO 2017, Part II. LNCS*, vol. 10402 (Springer, 2017), pp. 647–678

[4] C. Bouillaguet, O. Dunkelman, G. Leurent, P.-A. Fouque, Another look at complementation properties, in S. Hong , T. Iwata, editors, *FSE. LNCS*, vol. 6147 (Springer, 2010), pp. 347–364

[5] A. Bogdanov, V. Rijmen, Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Des. Codes Cryptogr.*, **70**(3), 369–383, (2014)

[6] E. Biham, A. Shamir, Differential cryptanalysis of DES-like cryptosystems, in A. Menezes, S.A. Vanstone, editors, *CRYPTO*. LNCS. vol. 537 (Springer, 1990), pp. 2–21

[7] A. Biryukov, D. Wagner, Slide attacks, in L.R. Knudsen, editor, *FSE*. LNCS, vol. 1636 (Springer, 1999), pp. 245–259

[8] J. Guo, J. Jean, I. Nikolic, K. Qiao, Y. Sasaki, S. Sim, Invariant subspace attack against Midori64 and the resistance criteria for S-box designs. *IACR Trans. Symm. Cryptol.*, **2016**(1), 33–56, (2016)

[9] V. Grosso, G. Leurent, F.-X. Standaert, K. Varici, A. Journault, F. Durvaux, L. Gaspar, S. Kerckhof, SCREAM v1. 2014. Submission to CAESAR competition

[10] V. Grosso, G. Leurent, F.-X. Standaert, K. Varici, A. Journault, F. Durvaux, L. Gaspar, S. Kerckhof, SCREAM v3. 2015. Submission to CAESAR competition

[11] V. Grosso, G. Leurent, F.-X. Standaert, K. Varici, LS-Designs: Bitslice encryption for efficient masked software implementations, in C. Cid, C. Rechberger, editors, *FSE*. LNCS, vol. 8540 (Springer, 2014), pp. 18–37

[12] M. Hermelin, J.Y. Cho, K. Nyberg, Multidimensional linear cryptanalysis of reduced round Serpent, in Y. Mu, W. Susilo, J. Seberry, editors, *ACISP*.LNCS, vol. 5107 (Springer, 2008), pp. 203–215

[13] C. Harpes, G.G. Kramer, J.L. Massey, A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma, in L.C. Guillou, J.-J. Quisquater, editors, *EUROCRYPT*. LNCS, vol. 921 (Springer, 1995), pp. 24–38

[14] L.R. Knudsen, Truncated and higher order differentials, in B. Preneel, editor, *FSE*. LNCS, vol. 1008 (Springer, 1994), pp. 196–211

[15] L.R. Knudsen, M.J.B. Robshaw, Non-linear approximations in linear cryptanalysis, in U.M. Maurer, editor, *EUROCRYPT*. LNCS, vol. 1070 (Springer, 1996), pp. 224–236

[16] G. Leander, M.A. Abdelraheem, H. AlKhzaimi, E. Zenner, A cryptanalysis of PRINTCIPHER: the invariant subspace attack, in P. Rogaway, editor, *CRYPTO*. LNCS, vol. 6841 (Springer, 2011), pp. 206–221

[17] G. Leander, B. Minaud, S. Rønjom, A generic approach to invariant subspace attacks: cryptanalysis of robin, iscream and zorro, in E. Oswald, M. Fischlin, editors, *EUROCRYPT*. LNCS, vol. 9056 (Springer, 2015), pp. 254–283

[18] M. Liskov, R.L. Rivest, D. Wagner, Tweakable block ciphers. *J. Cryptol.*, **24**(3), 588–613, (2011)

[19] M. Matsui, Linear cryptanalysis method for DES cipher, in T. Helleseth, editor, *EUROCRYPT*. LNCS, vol. 765 (Springer, 1993), pp. 386–397

[20] S. Moriai, T. Shimoyama, T. Kaneko, Higher order differential attak of CAST cipher, in S. Vaudenay, editor, *FSE*. *LNCS*, vol. 1372 (Springer, 1998), pp. 17–31

[21] National Bureau of Standards, *Data Encryption Standard (DES)*, (1977). Federal Information Processing Standards Publication 46

[22] M. Özen, M. Çoban, F. Karakoç, A guess-and-determine attack on reduced-round Khudra and weak keys of full cipher. *IACR Cryptol. ePrint Arch.*, **2015**, 1163, (2015).

[23] U.S. Department of Commerce/National Institute of Standards and Technology, *Specification for the Advanced Encryption Standard (AES)*, (2001). Federal Information Processing Standards Publication 197

[24] T. Van Le, R. Sparr, R. Wernsdorf, Y. Desmedt, Complementation-like and cyclic properties of AES round functions, in H. Dobbertin, V. Rijmen, A. Sowa, editors, *AES Conference*. LNCS, vol. 3373 (Springer, 2004), pp. 128–141