A Distinguisher on PRESENT-Like Permutations with Application to SPONGENT

Guoyan Zhang[1,2] and Meicheng Liu[3,4](✉)

[1] School of Computer Science and Technology, Shandong University, Jinan, China
[2] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[3] Nanyang Technological University, Singapore
[4] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
guoyanzhang@sdu.edu.cn, meicheng.liu@gmail.com

Abstract. At Crypto 2015, Blondeau *et al.* showed a known-key analysis on the full PRESENT lightweight block cipher. Based on some of the best differential distinguishers, they introduced a meet in the middle (MitM) layer to pre-add the differential distinguisher, which extends the number of attacked rounds on PRESENT from 26 rounds to full rounds without reducing differential probability.

In this paper, we generalize their method and present a distinguisher on a kind of permutations called PRESENT-like permutations. This generic distinguisher is divided into two phases. The first phase is a truncated differential distinguisher with strong bias, which describes the unbalancedness of the output collision on some fixed bits, given the fixed input in some bits, and we take advantage of the strong relation between truncated differential probability and capacity of multidimensional linear approximation to derive the best differential distinguishers. The second phase is the meet-in-the-middle layer, which is pre-added to the truncated differential to propagate the differential properties as far as possible. Different with Blondeau *et al.*'s work, we extend the MitM layers on a 64-bit internal state to states with any size, and we also give a concrete bound to estimate the attacked rounds of the MitM layer.

As an illustration, we apply our technique to all versions of SPONGENT permutations. In the truncated differential phase, as a result we reach one, two or three rounds more than the results shown by the designers. In the meet-in-the-middle phase, we get up to 11 rounds to pre-add to the differential distinguishers. Totally, we improve the previous distinguishers on all versions of SPONGENT permutations by up to 13 rounds.

Keywords: symmetric ciphers, PRESENT, SPONGENT, truncated differential, meet-in-the-middle, multidimensional linear approximation.

1 Introduction

The design goal of cryptographic scheme is to meet the secure requirements, and the need of resource restricted applications such as RFID and sensor networks makes the research of lightweight cryptography naturally attract a lot of attention. Many lightweight cryptography including stream ciphers like Trivium [1] and Grain [2], blockciphers like KATAN/KTANTAN [3], PRESENT [4] and LED [5] *etc.* and hash functions like QUARK [6], SPONGENT [7,8] and PHOTON [9] have been proposed in the past few years. The aim to correctly evaluate the security of these proposals has become a primordial task. This has been proved by the big number of security analyses of the previous primitives that has appeared in [10,11,12,13,14,15].

The block cipher PRESENT has become ISO/IEC standard [16] because of its impressive hardware performance and strong security assurance. Inspired by the design of PRESENT, there are a series of ciphers, e.g. SPONGENT [7,8], Puffin [17], PRINTcipher [18], MAYA [19], EPCBC [20], and RECTANGLE [21]. The core parts of these PRESENT-like ciphers include three layers: a key addition layer, a substitution layer realized by many parallel small scale Sboxes and a bit-wise permutation layer. Recently, the SPONGENT family of hash functions has also become ISO/IEC standard [22].

In the context of lightweight cryptanalysis, quite a few security analyses are developed for PRESENT-like ciphers, and the influence of differential and linear cryptanalysis are obvious. Borghoff *et al.* [10] focused on the analysis of PRESENT-like ciphers with secret Sboxes and gave a novel differential-style attack on MAYA which enabled us to find Sboxes in the first round one by one. Cho [23] showed an attack on 26-round PRESENT using the easy-to-trace linear trails with large correlations. Bulygin [13] found an efficient method to compute the capacities of EPCBC to get the attack on the full round EPCBC-96, and also presented an attack on 26 rounds of PRESENT-128 with more high success probability than Cho's. For PRESENT, linear cryptanalysis-based attacks were much more powerful against differential cryptanalysis-based attacks with 19 rounds attack, until Blondeau and Nyberg [24] presented a link between differential probability and linear correlation, which converted a multidimensional linear distinguisher into a truncated differential one which made truncated differential attacks up to 26 rounds of PRESENT. Combining their 26-round truncated differential attack, Blondeau *et al.* [15] gave a full-round known-key distinguisher valid for both PRESENT-80 and PRESENT-128. Firstly, they got one of the best truncated differential distinguisher from [24] which showed a statistical bias of the number of collisions on a few predetermined output bits under the fixed input bits. Secondly, they extended the round number of the differential attack by prepending a meet-in-the-middle (MitM) layer under the constraints that the output bits of the MitM layer must be exactly the same with those set on the input bits of the truncated differential layer. Then enough number of plaintexts are provided to make the distinguish attack succeed with high probability, where the plaintexts satisfied the constraints on both input and output of the MitM layer.

1.1 Our Contribution

We firstly define a kind of permutations called PRESENT-like permutations, which capture various cryptographic primitives such as PRESENT and SPONGENT. Then we propose a distinguisher for such PRESENT-like permutations. Similar to Blondeau *et al.*'s distinguisher, it includes two layers, that is, the MitM layer and the truncated differential distinguisher. The former is prepended to the latter, and provides enough number of plaintexts to ensure the distinguisher succeed with non-negligible probability.

In the MitM layer of Blondeau *et al.*'s distinguisher on PRESENT, the bits of internal state are divided into four groups. One of our observations on this MitM layer is that the bits of internal state can simply be divided into two groups. Noting that the internal state of PRESENT has 64 bits, one can easily generalize their method to obtain a similar MitM layer on any internal state with power-of-2 bits. Nevertheless, it is not trivial to generalize the MitM layer for the other cases. Our another observation on the MitM layer is that the number of its rounds is conducted by two factors, one of which is related to the size n of the internal state and the other of which is related to the factorization of the size n. Based on these observations, and according to the characteristic of PRESENT-like permutations, we construct the MitM layer and show a lower bound on the number of rounds extended by the MitM layer, for PRESENT-like permutations with any sizes. This bound is explicitly formulated by the size of the permutations.

We use the truncated differential distinguisher built based on [24,23]. The truncated differential distinguishing property is a statistical bias of the number of collisions on a few predetermined output bits when some predetermined input bits are fixed, which is related to the capacity of a multidimensional linear approximation [24]. For PRESENT-like permutations, this capacity can be obtained from the 1-bit linear trails, which benefits from the bit-permutation linear layer of the permutations.

Finally, we apply our distinguisher on all the versions of SPONGENT permutations and summarize our results compared with the previous distinguishers in Table 1. As shown in this table, we can distinguish up to 13 more rounds than the analysis [7,8] provided by the designers on all versions, and 7 more rounds than the result [12] which shows a distinguisher on the special version SPONGENT-88/80/8.

Table 1. Summary of Distinguishers of SPONGENT Permutations

Version	b	R	r_0	r_1	r	[7,8]	[12]	d
SPONGENT-88/80/8	88	45	7	23	30	22	23	7
SPONGENT-88/176/88	264	135	9	68	77	66		11
SPONGENT-128/128/8	136	70	7	36	43	34		9
SPONGENT-128/256/128	384	195	11	98	109	96		13
SPONGENT-160/160/16	176	90	7	46	53	44		9
SPONGENT-160/160/80	240	120	7	62	69	60		9
SPONGENT-160/320/160	480	240	9	123	132	122		10
SPONGENT-224/224/16	240	120	7	62	69	60		9
SPONGENT-224/224/112	336	170	9	86	95	84		11
SPONGENT-224/448/224	672	340	9	172	181	169		12
SPONGENT-256/256/16	272	140	9	69	78	68		10
SPONGENT-256/256/128	384	195	11	98	109	96		13
SPONGENT-256/512/256	768	385	11	194	205	192		13

b: the size of internal state
R: the number of full rounds
r_0: the number of rounds of the MitM layer
r_1: the number of rounds of truncated differential distinguisher
r: the total number of rounds of our distinguisher
d: the number of rounds we improve on the previous best distinguisher

1.2 Organization

This paper is organized as follows. In Section 2 the structures of PRESENT-like permutations and SPONGENT are briefly described. The framework of our generic attack, including the truncated differential distinguisher and the meet-in-the-middle layer, is shown in Section 3. The Section 4 presents the detailed distinguisher on all versions of SPONGENT. Section 5 concludes this paper.

2 Research Background

In this section, we start by briefly describing a generic view of a PRESENT-like permutation to capture various cryptographic primitives such as PRESENT [4] and SPONGENT [7,8].

2.1 Brief Description of PRESENT-Like Permutations

We define a PRESENT-like permutation as a permutation that applies R rounds of a round function to update an internal state consisting of n cells, where each of the cells has a size of c bits. In this paper, we focus on the case $c = 4$, while our technique can also be adapted to the other cases, e.g. for PRINTcipher [18], $c = 3$.

The round function uses a substitution-permutation network (SPN). It starts by xoring a round-dependent constant to the state. Then, it applies a substitution layer which relies on a $c \times c$ nonlinear bijective Sbox. Finally, the round function performs a bit-permutation linear layer L, where

$$L(i) = \begin{cases} i \cdot n \mod (c \cdot n - 1), & \text{if } i \in \{0, \ldots, c \cdot n - 2\}, \\ c \cdot n - 1, & \text{if } i = c \cdot n - 1. \end{cases} \tag{1}$$

Note that our distinguisher can also be adapted to other ciphers, such as PRINTcipher, which use the inverse of L as their linear layer.

In the case of PRESENT-like block ciphers analyzed in the known/chosen-key model, the subkeys generated by the key schedule are incorporated into the known constant addition layer. We state here that for PRESENT-like block ciphers the generic distinguisher presented in this paper is suitable in the known-key model.

2.2 Brief Description of SPONGENT

SPONGENT [7,8] is a sponge-based hash function family with 13 versions. Its various parameterizations are refered to as SPONGENT-$n/c/r$ for different hash sizes n, capacities c, and rates r, as depicted in Table 1. All variants with the same output size of n bits are referred to as SPONGENT-n, and there are five different output: SPONGENT-88, SPONGENT-128, SPONGENT-160, SPONGENT-224 and SPONGENT-256. In this section, we denote by b, n, c, r separately the size of the internal state, hash size, capacity and rate.

SPONGENT construction is an iterated design with three phases: the initialization phase is to pad the message into a multiple of r bits, and the absorbing phase is to xor the r-bit input message blocks into the first r bits of the state, and the squeezing phase is to get the n-bit output.

As a lightweight hash function, it does not use the lightweight block cipher as its core part, and it introduces the PRESENT-like permutation as its core permutation. The permutation layer operates linearly on the b bits as follows: the b bit state $STATE_i$ is firstly xored with round constant $C_b(i)$ at its leftmost bits and with round constant $IC_b(i)$ at its rightmost bits, where $C_b(i)$ is the state of an LFSR, and $IC_b(i)$ is the value of $C_b(i)$ with its bits in reversed order. Secondly, the 4-bit Sbox is described as follows:

$$S[\cdot] = \{\texttt{0xE,0xD,0xB,0x0,0x2,0x1,0x4,0xF,0x7,0xA,0x8,0x5,0x9,0xC,0x3,0x6}\}.$$

Finally, the bit i of the state is moved to the bit position $P_b(i)$, where

$$P_b(i) = \begin{cases} i \cdot b/4 \mod (b-1), & \text{if } i \in \{0, \ldots, b-2\}, \\ b-1, & \text{if } i = b-1. \end{cases} \quad (2)$$

It can be seen in Figure 1, which takes SPONGENT-88/80/8 as an example.

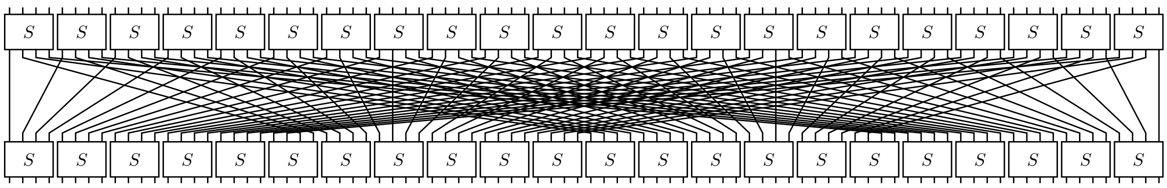

Fig. 1. The bit permutation layer of SPONGENT-88/80/8

2.3 Previous Results on SPONGENT

As claimed in [25,26], the sponge construction can get the preimage security of 2^r as well as the second preimage and collision securities of $2^{c/2}$, if this core permutation does not have any structural distinguishers. As far as we know, there are very few outside attacks on SPONGENT.

Many analyses on the core permutation were considered. The designers [7,8] considered input and output linear masks with hamming weight one, and showed that there were at most $(R/2)$-round linear trail with correlation $c_w \geq 2^{-b/2}$ (except that SPONGENT-160/320/160 allows $2+R/2$ rounds). They also estimated longest differential characteristics holding with probability in the range of 2^{-b}. Different with block cipher PRESENT, SPONGENT permutations have at most one linear trails with one active S-box in each round, so they cannot get the same high probability linear approximations as PRESENT. Abdelraheem [12] and Bao *et al.* [27] estimated the probabilities of differential and linear approximations for SPONGENT. Abdelraheem [12] considered input and output masks with Hamming weight ≤ 4 with large but sparse correlation matrix, which overcame the memory and time problems, and improved the linear cryptanalysis on a special variant, SPONGENT-88/80/8, by one more round than the result provided by the designers.

3 A Distinguisher for PRESENT-like Permutations

In this section, we present a generic distinguisher on PRESENT-like permutations that are defined in Section 2.1. This distinguisher include two phases, the truncated differential distinguisher and the meet-in-the-middle layer. The truncated differential phase describes the collision bias on the predetermined bits of the ciphertexts whose plaintexts are the same on some bits, while the meet-in-the-middle phase is prepended to the truncated differential to extend the distinguisher to more rounds without reducing the success probability. The following two subsections give the details of these two phases.

3.1 Truncated Differential Distinguisher

Definition of Truncated Differential Distinguisher. The technique of truncated differential attack was introduced by Knudsen [28]. Whereas ordinary differential cryptanalysis analyzes the full difference between two texts, the truncated variant considers differences that are only partially determined. That is, the attack makes predictions of only some of the bits instead of the full block. Assuming that the permutation is $F : \mathcal{F}_2^n \to \mathcal{F}_2^n, x = (x_s, x_t) \mapsto y = (y_q, y_r)$, we consider a truncated differential composed of 2^t input differences $(0, \delta_t)$ and 2^r output differences $(0, \Delta_r)$. The truncated differential distinguisher describes the unbalance of the collisions on the q bits of the output under the condition that the s bits of the input are fixed. The distinguishing attack model is described as in Algorithm 1, see also [15,24].

Algorithm 1 Truncate differential distinguisher

Input: Given a plaintext set P including N plaintexts with the same value on the s bits and the truncated differential $(0, \delta_t) \to (0, \Delta_r)$ with probability p.

1: Set a counter D to 0 and a table T with size 2^q to 0.
2: For each plaintext $x = (x_s, x_t)$:
  Compute $(y_q, y_r) = E_K(x_s, x_t)$,
  $T[y_q] = T[y_q] + 1$.
3: Compute scoring function $D = \sum_{0 \leq l \leq 2^q - 1} T[l](T[l] - 1)/2$.
4: For threshold function τ, if $D > \tau$, conclude that this is the cipher.

The success probability P_S of the distinguisher is defined by the following equation:

$$P_S(N^2/2) = \Phi(\frac{\mu_R + \mu_W}{\sigma_R - \sigma_W}),$$

where $\mu_R = N^2/2 \cdot p, \sigma_R^2 \approx N^2/2 \cdot 2^{-q}, \mu_W = N^2/2 \cdot 2^{-q}, \sigma_W^2 = N^2/2 \cdot 2^{-q}$, and τ is the threshold function satisfying $\tau = \mu_R - \sigma_R \phi^{-1}(N^2/2)$.

Of course, as claimed in [15], the bias is usually small, so we cannot pre- or post-add other differential characteristic to extend the attacked rounds number, which will cause lower differential probability.

Truncated Differential for PRESENT-Like Permutations. In order to get a truncated differential with more strong bias, as more number of high differential characteristics as possible are considered to improve the bias, especially for those ciphers with bad differential characteristics and good linear approximation. But due to the limitation of time and memory complexity, only few differentials are involved, which usually causes the truncated differential with low probability. Blondeau and Nyberg [24] provided a link between truncated differential and multidimensional linear properties to convert a multidimensional linear distinguisher into a truncated differential distinguisher, which greatly balanced the differential-based attack and linear-based attack on the same ciphers. For example, the link made it possible for the truncated differential attack up to 26 rounds of PRESENT [24]. The following is the strong relation of the truncated differential probability with the capacity of the multidimensional linear approximation.

Theorem 1 ([24]) *Let* $F : \mathcal{F}_2^s \times \mathcal{F}_2^t \rightarrow \mathcal{F}_2^q \times \mathcal{F}_2^r$ *denote a vectorial Boolean function satisfying* $s + t = q + r = n$. *Given a multidimensional approximation* $[(a_s, 0), (b_q, 0)]_{a_s \in \mathcal{F}_2^s, b_q \in \mathcal{F}_2^q}$ *with capacity* C *and a truncated differential composed of* 2^t *input differences* $(0, \delta_t) \in \{0\} \times \mathcal{F}_2^t$ *and* 2^r *output differences* $(0, \gamma_r) \in \{0\} \times \mathcal{F}_2^r$ *with probability* p, *then*

$$p = 2^{-q}(C + 1).$$

Note that the above result applies to PRESENT-like permutations. Thus their truncated differential with strong bias can be converted from multidimensional linear approximation. For PRESENT-like permutations, their constructions are similar to the core part of PRESENT, then we can find the multidimensional linear approximation for PRESENT-like permutations according to the analysis on PRESENT presented by Cho [23]. We consider $r_1 = r_1^* + 2$ rounds linear characteristic as follows. In the first round, we consider that the c-bit (c is the size of the cell) input mask α of the active Sbox can take arbitrary value from 1 to 2^c and the output mask takes a single-bit value. In the last round, we require that each input mask of the active Sbox takes a single-bit value and the output mask β can take arbitrary value from 1 to 2^c. Furthermore, we limit that both the two rounds have only one active Sbox. In the middle r_1^* rounds, linear trails satisfying input mask and output mask with hamming weight 1 in each round are considered. We can set up the correlation matrix of one round to get the correlation of the middle r_1^* rounds. The probability of truncated differential is computed as $P = 2^{-q}(1 + C)$ by the computation of the correlation of multidimensional linear approximation.

3.2 Generic Meet-in-the-Middle Layer for PRESENT-Like Permutations

In Blondeau *et al.*'s known-key attack on PRESENT [15], the key step was the MitM layer which propagated the differential properties up to 26 rounds by prepending the extra 7 rounds to the truncated differential distinguisher. In order to make the truncated differential distinguisher valid, a set of plaintexts satisfying the input constraints of the truncated differential must be identified. Namely, the internal states of these plaintexts after several rounds as the output of the MitM layer should satisfy the input constraints of the truncated differential. In order to efficiently identify such a set of plaintexts, after separately fixing the few bits of the input and output of the plaintexts, their MitM layer carried out a forward computation to get partial internal state bits after few rounds of the MitM layer by guessing just few bits and carried out an independent backward computation to get partial internal state bits for the last one and half round of the MitM layer. Finally the set of plaintext was identified by carrying out a gradually matching process. Because of the small Sbox and the bit-permutation linear layer of PRESENT, partial output bits can be determined by guessing few bits. Benefitting from the 64-bit internal states, the internal state can be divided into few groups, and gradually matching process can be easily carrying out.

In the MitM layer of Blondeau *et al.*'s distinguisher on PRESENT, the bits of internal state are divided into four groups. One of our observations on this MitM layer is that the bits of internal state can simply be divided into two groups. Noting that the internal state of PRESENT has 64 bits, one can easily generalize their method to obtain a similar MitM layer on any internal state with power-of-2 bits. Nevertheless, it is not trivial to generalize the MitM layer for the other cases. Our another observation on the MitM layer is that the number of its rounds is conducted by two factors, one of which is related to the size n of the internal state and the other of which is related to the factorization of the size n. Based on these observations and according to the characteristic of PRESENT-like permutations, we show a generic MitM layer for PRESENT-like permutations with any sizes, and we also give a lower bound on the number of rounds extended by this MitM layer.

Hereinafter, we assume that a PRESENT-like permutation uses an internal state of $4n$ bits, that is, it consists of n 4-bit cells, for even n. We first provide the details of the procedure, then discuss the number of rounds the generic MitM layer consists of and the complexity of this procedure.

The Procedure. As in [15], we denote by X_i the internal state after i-th round of a PRESENT-like permutation, and by Y_i the internal state after applying Sbox layer to X_i. The detailed procedure for generic MitM layer is described as below.

1. Set four bits input to a single Sbox of plaintexts to a randomly chosen 4-bit value, and compute the corresponding bits of X_1 in the forward direction. These bits are input to four different Sboxes in the second round. Then we guess the other 12 bits input to these Sboxes, and compute in the forward direction to get 16 bits of X_2. Iteratively, we guess the other $4^i - 4^{i-1}$ bits input to the 4^{i-1} active Sboxes of X_{i-1} and compute in the forward direction to get 4^i bits of X_i, for $i = 3, \cdots, \mu_n$. In total we get a set of $2^{4^{\mu_n}-4}$ such values of X_{μ_n} and each value has 4^{μ_n} bits determined.
2. Similarly, set the four bits at expected positions of Y_{r_0} to a randomly chosen 4-bit value, and compute the corresponding bits of Y_{r_0-1} in the backward direction. These bits are input to the inversion of four different Sboxes in the $(r_0 - 1)$-th round. Then we guess the other 12 bits input to the inversion of these Sboxes of Y_{r_0-1}, and compute in the backward direction to get 16 bits of Y_{r_0-2}. Iteratively, we guess the other $4^i - 4^{i-1}$ bits input to the inversion of the 4^{i-1} active Sboxes of Y_{r_0-i+1} and compute in the backward direction to get 4^i bits of Y_{r_0-i}, for $i = 3, \cdots, \mu_n$. In total we get a set of $2^{4^{\mu_n}-4}$ such values of $Y_{r_0-\mu_n}$ and each value has 4^{μ_n} bits determined.
3. For each partially determined value of X_{μ_n} and $Y_{r_0-\mu_n}$, repeat the following:
   (a) Divide the bits of X_{μ_n} into two disjoint groups, each of which contains half of determined bits and half of undetermined bits. Each group consists of $2n$ bits which are input to neighbouring $\frac{1}{2}n$ Sboxes. Then for each group, we guess the $2n - \frac{1}{2}4^{\mu_n}$ undetermined bits of X_{μ_n}, and compute in the forward direction to get $2n$ bits of $X_{\mu_n+t_n}$. We store in a table $T_{X,i}$ the values of partially determined $X_{\mu_n+t_n}$ computed from the i-th group, $i = 0, 1$.
   (b) Similarly, divide the bits of $Y_{r_0-\mu_n}$ into two disjoint groups, each of which contains half of determined bits and half of undetermined bits. Each group consists of $2n$ bits which are input to the inversion of $\frac{1}{2}n$ Sboxes at carefully chosen positions. Then for each group, we

guess the $2n - \frac{1}{2}4^{\mu_n}$ undetermined bits of $Y_{r_0-\mu_n}$, and compute in the backward direction to get $2n$ bits of $X_{\mu_n+t_n}$, which correspond to $2n$ bits of $Y_{r_0-\mu_n-t_n} = X_{\mu_n+t_n-1}$ up to a bit-permutation linear layer. We store in a table $T_{Y,i}$ the values of partially determined $X_{\mu_n+t_n}$ computed from the i-th group, $i = 0, 1$.

(c) Then merge those tables to find a set of fully-determined values of $X_{\mu_n+t_n}$.

i. Merge the tables $T_{X,i}$ and $T_{Y,i}$ to T_i respectively for $i = 0, 1$. By merging these two tables, we mean to merge every two partially-determined values of $X_{\mu_n+t_n}$, each from a table and sharing the same bit values at the common determined bit positions, into a new partially-determined value of $X_{\mu_n+t_n}$ with all their determined bits, and then to include this new value of $X_{\mu_n+t_n}$ in table T_i. Note that each value of $T_{X,i}$ and each value of $T_{Y,i}$ share n determined bit positions (if the positions of Sboxes of each group defined at Step 3b are carefully chosen). Hence table T_i has on average $2^{2\cdot(2n-\frac{1}{2}4^{\mu_n})-n} = 2^{3n-4^{\mu_n}}$ values, each of which has $2 \cdot 2n - n = 3n$ bits.

ii. Merge T_0 and T_1. Notice that T_0 and T_1 share $2n$ determined bit positions of $X_{\mu_n+t_n}$. Hence we obtain $2^{2\cdot(3n-4^{\mu_n})-2n} = 2^{4n-2\cdot4^{\mu_n}}$ values on average, each of which has $2 \cdot 3n - 2n = 4n$ bits consisting of the full bits of $X_{\mu_n+t_n}$.

The algorithm is to find a set of internal state values of $X_{\mu_n+t_n}$, whose corresponding plaintexts can satisfy the constraints on the input and output of the MitM layer. Totally, we obtain on average $2^{2\cdot(4^{\mu_n}-4)} \cdot 2^{4n-2\cdot4^{\mu_n}} = 2^{4n-8}$ plaintexts by inversely computing from the fully-determined values of $X_{\mu_n+t_n}$, which satisfies the constraints on the input and output of the MitM layer.

The Number of Rounds. The number of rounds of the generic MitM layer is determined by Step 1, Step 2, Step 3a and Step 3b. Notice that Step 1 and Step 2 are symmetric and thus involve the same number of rounds. More exactly, they both involve $\mu_n = \lfloor \log_4 n \rfloor$ rounds, where n is the total number of Sboxes. Also, Step 3a and Step 3b are symmetric and involve the same number of rounds. Denote by t_n the maximum integer such that $4^{t_n} \mid 2n = \frac{4n}{2}$. Since in Step 3 the bits of the internal state are divided into two groups, Step 3a and Step 3b both involve t_n rounds. Totally, the number of rounds of the generic MitM layer is

$$r_0 = 2(\mu_n + t_n) - 1 = 2\lfloor \log_4 n \rfloor + 2t_n - 1.$$

Taking PRESENT for example, $n = 16$, $\mu_n = 2$, $t_n = 2$, and thus $r_0 = 7$, which is exactly the number of rounds of the MitM layer for PRESENT presented in [15]. Taking SPONGENT-88/80/8 for example, $n = 22$, $\mu_n = 2$, $t_n = 1$, and thus $r_0 = 5$, which is two less than that proposed in Section 4.2. We will discuss later how to improve the generic MitM layer for same special cases like SPONGENT-88/80/8.

Complexity. The complexity of the algorithm is dominated by Step 3. Since there are $2^{4^{\mu_n}-4}$ values of X_{μ_n} from the forward computations and $2^{4^{\mu_n}-4}$ values of $Y_{r_0-\mu_n}$ from the backward computations, Step 3 is executed $2^{2\cdot(4^{\mu_n}-4)}$ times. The complexity of each execution is dominated by Step 3(c)ii, that is merging T_0 and T_1, which needs $2^{3n-4^{\mu_n}}$ table lookups. Hence the total complexity of Step 3 is $2^{2\cdot(4^{\mu_n}-4)} \cdot 2^{3n-4^{\mu_n}} = 2^{3n+4^{\mu_n}-8} \leq 2^{4n-8}$ table lookups. Once a match of the MitM layer has been found, we can encrypt this value $X_{\mu_n+t_n}$ over the $r_1+\mu_n+t_n-1$ rounds and increment the counter D given in the previous section. The memory complexity of this attack is dominated by the storage of the table T_0 and T_1 which is $2 \cdot 2^{3n-4^{\mu_n}} \cdot 3n = 3n \cdot 2^{3n-4^{\mu_n}+1}$ bits. To sum up, the total time complexity of the distinguisher is $2^{3n+4^{\mu_n}-8}$ table lookups and 2^{4n-8} encryptions, and the memory complexity is $3n \cdot 2^{3n-4^{\mu_n}+1}$ bits.

Improved Greneric Meet-in-the-Middle Layer. For even n, we always have $4 \mid 2n$ and thus $t_n \geq 1$. For the case that $t_n = 1$ and $4n > 32 = 2 \cdot 4^2$, e.g. $4n = 88$ for SPONGENT-88/80/8, it is possible to improve the greneric MitM layer by two more rounds, consisting of one round in forward direction and one round in backward direction, at the cost of increasing the complexity by using the strategy as shown in Section 4.2.

4 The Distinguishers on SPONGENT Permutations

As an application of our generic distinguisher shown in Section 3, we first analyse the internal permutation used in SPONGENT-88/80/8, and we obtain a 30-round truncated differential distinguisher including 7 rounds in MitM phase and 23 rounds in truncated differential phase. Then, we apply the similar method to get the distinguishers for the other versions of SPONGENT.

4.1 Truncated Differentials with Strong Bias for SPONGENT Permutations

Truncated Differentials with Strong Bias for SPONGENT-88/80/8 Permutation. As described in the previous section, the truncated differential with strong bias of SPONGENT-88/80/8 permutation can be converted from multidimensional linear approximation. According to Theorem 1, the greater the total correlation of multidimensional linear approximation, the stronger the bias of truncated differential. The Sbox in the SPONGENT permutation was chosen carefully to avoid many linear trails with one active Sbox in each round existing on PRESENT, see [7,8]. For instance in SPONGENT-88/80/8 permutation, there is only one trail that has one active Sbox at each round. For other versions, the number of the trails satisfying input and output mask with hamming weight 1 on each round is also less than five. So the designer claimed that there was linear distinguisher possible for not more than 22 rounds for SPONGENT-88/80/8 permutation when only input mask and output mask with hamming weight 1 were considered. Abdelraheem [12] increased the linear distinguisher to 23 rounds with capacity $2^{-87.5}$ considering linear characteristic hamming weight at most 4. As shown in [12], however, the characteristics with Hamming weight 4 contributed negatively to the total correlation. Furthermore, from their analysis, we can see that correlation was not increased much more when linear characteristics with hamming weight 3 or 4 were considered.

Our experiments show that the best capacity of multidimensional linear approximation is 2^{-84}, where the number of rounds is $r_1^* = 21$ and the truncated differential characteristic is (20,19). That is, the inputs share the same values at bits $\{80, 81, 82, 83\}$, i.e. the input bits to S_{20}, and the outputs share the same values at bits $\{76, 77, 78, 79\}$, i.e. the input bits to S_{19} for the next round. In other words, bits $\{80, 81, 82, 83\}$ of the input mask are non-zeros, and bits $\{76, 77, 78, 79\}$ of the output mask are non-zeros.

The details of the procedure are given as follows.

1. Set up the correlation matrix of one round considering linear trails with input mask and output mask of hamming weight 1, and we get many 21-round linear approximation, where a linear layer is included before the first found. We choose linear approximation whose output mask reaches S_{19} after one round and input mask reaches S_{20} after one inversion round. The capacity of these linear approximation is 2^{-84} and $r_1^* = 21$.
2. According to Parseval's theorem: $\sum_{\alpha_i=0}^{15} \rho(\alpha_i, 2^u)^2 = \sum_{\beta_j=0}^{15} \rho(2^v, \beta_j)^2 = 1$ for any $u, v \in \{0, 1, 2, 3\}$, the input of S_{20} in the first round travel all the value from 0 to 15 and the output of S_{19} in the 23 round travel all the value from 0 to 15. The capacity of the $r_1 = r_1^* + 2 = 23$ is the same with the capacity of the middle r_1^* round, and the capacity of 23 round is 2^{-84}.

Then for instance such truncated differential distinguisher on 23 rounds, we can compute the probability of the distinguisher as $p = 2^{-q}(C+1) = 2^{-4}(2^{-84}+1) = 2^{-4} + 2^{-88}$ according to

Theorem 1. Combining with a 7-round MitM layer, which will be shown in Section 4.2, we can get $2^{80} \cdot (2^{80} - 1)/2 \approx 2^{159}$ pairs of messages which are under the constraint that their outputs are the same at bits (80,81,82,83). Then we can distinguish 30-round of SPONGENT-88/80/8 from a random permutation with success probability 50.22%.

Applictions to Other Vesions of SPONGENT Permutations. Similar, we apply the method to other vesions of SPONGENT permutations for finding the best truncated differentials. The results are listed in Table 2, respectively for different versions, where b is the size of internal state, R is the number of full rounds, r_1 is the number of rounds of the truncated differential, C is the capacity of the best multidimensional approximation which equals the probability of the corresponding truncated differential according to Theorem 1, and P_S is the success probability of the truncated differential distinguisher. We also compare our results with the previous distinguishers in Table 1. As shown in this table, we can reach one, two or three rounds more than the results shown by the designers.

Table 2. Differential Distinguishers of SPONGENT

Version	b	R	r_1	C	$Charac$	Ps
SPONGENT-88/80/8	88	45	23	-84	(20,19)	50.22%
SPONGENT-88/176/88	264	135	67	-259	(44,65)	50.44%
SPONGENT-88/176/88	264	135	68	-263	(44,65)	50.03%
SPONGENT-128/128/8	136	70	35	-131	(22,31)	50.44%
SPONGENT-128/128/8	136	70	36	-135	(20,31)	50.03%
SPONGENT-128/256/128	384	195	97	-376.678	(64,95)	52.20%
SPONGENT-128/256/128	384	195	98	-380.678	(64,95)	50.14%
SPONGENT-160/160/16	176	90	45	-170.415	(20,31)	50.66%
SPONGENT-160/160/16	176	90	46	-174.415	(20,40)	50.04%
SPONGENT-160/160/80	240	120	61	-233.415	(58,58)	51.32%
SPONGENT-160/160/80	240	120	62	-238	(58,59)	50.06%
SPONGENT-160/320/160	480	240	122	-475.3	(80,118)	50.36%
SPONGENT-160/320/160	480	240	123	-478.83	(80,112)	50.03%
SPONGENT-224/224/16	240	120	61	-233.415	(58,58)	51.32%
SPONGENT-224/224/16	240	120	62	-238	(58,59)	50.06%
SPONGENT-224/224/112	336	170	85	-329.415	(28,83)	51.32%
SPONGENT-224/224/112	336	170	86	-335	(60,78)	50.03%
SPONGENT-224/448/224	672	340	171	-666.95	(56,146)	50.46%
SPONGENT-224/448/224	672	340	172	-670.947	(56,146)	50.03%
SPONGENT-256/256/16	272	140	69	-268	(44,66)	50.22%
SPONGENT-256/256/128	384	195	97	-376.678	(64,95)	52.20%
SPONGENT-256/256/128	384	195	98	-380.678	(64,95)	50.14%
SPONGENT-256/512/256	768	385	193	-762.415	(128,191)	50.66%
SPONGENT-256/512/256	768	385	194	-766.415	(128,191)	50.04%

4.2 The Meet-in-the-Middle Layer for SPONGENT Permutations

The family of SPONGENT hash functions has totally 13 variants, which use 11 different sizes of internal states. In this section, we first show an improved meet-in-the-middle approach on SPONGENT-88/80/8 and then apply it to other variants.

The MitM Layer for SPONGENT-88/80/8. Next we illustrate the MitM layer for SPONGENT-88/80/8. Notice that the size of its internal state is 88, which is not a power of 2 and makes the

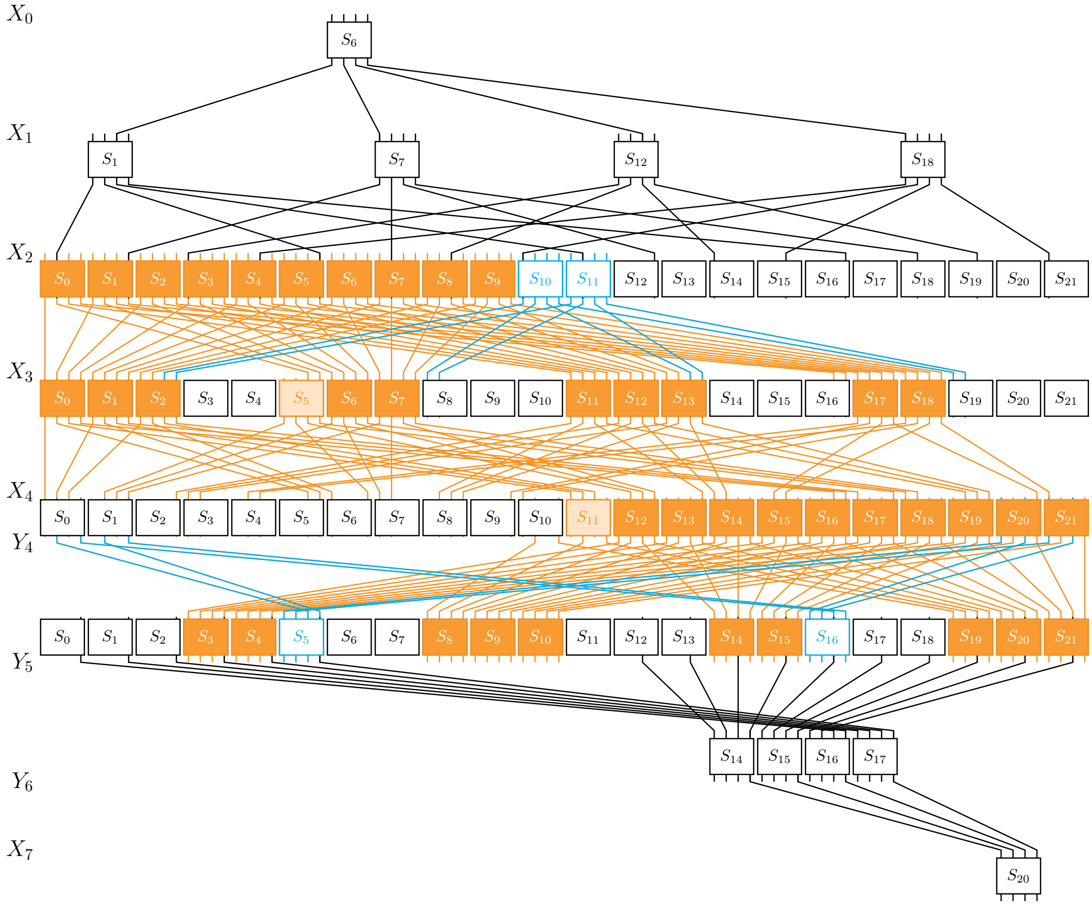

Fig. 2. MitM over the first 7 rounds of SPONGENT-88/80/8

MitM layer more complex than PRESENT. The MitM layer consists of 7 rounds. The constraints on the inputs are that they share the same values at bits $\{24, 25, 26, 27\}$, i.e. the input bits to S_6. The constraints on the outputs are that they share the same values at bits $\{80, 81, 82, 83\}$, i.e. the input bits to S_{20} for the eighth round. As in [15], we denote by X_i the internal state after i-th round of SPONGENT-88/80/8, and by Y_i the internal state after applying Sbox layer to X_i. The detailed procedure is described as below.

1. Set the bits $\{24, 25, 26, 27\}$ of plaintexts to a randomly chosen 4-bit value, and compute bits $\{6, 28, 50, 72\}$ of X_1 in the forward direction. These bits are input to Sboxes S_1, S_7, S_{12}, S_{18} in the second round. Then we guess the other 12 bits input to these Sboxes, i.e. bits $\{4, 5, 7, 29, 30, 31, 48, 49, 51, 73, 74, 75\}$ of X_1, and compute in the forward direction to get 16 bits of X_2, i.e. bits $\{1, 7, 12, 18, 23, 29, 34, 40, 45, 51, 56, 62, 67, 73, 78, 84\}$. It is also depicted as the first two rounds in Figure 2. Further, we guess the 6 bits $\{41, 42, 43, 44, 46, 47\}$ of X_2, which are input to Sboxes S_{10} and S_{11}, as shown in Figure 2 in cyan color. In total we get a set of 2^{18} such values of X_2 and each value has 22 bits determined.
2. Similarly, set the bits $\{80, 81, 82, 83\}$ of Y_7 to a randomly chosen 4-bit value, and compute bits $\{59, 63, 67, 71\}$ of Y_6 in the backward direction. These bits are input to the inversion of Sboxes $S_{14}, S_{15}, S_{16}, S_{17}$ in the sixth round. Then we guess the other 12 bits input to the inversion of these Sboxes, i.e. bits $\{56, 57, 58, 60, 61, 62, 64, 65, 66, 68, 69, 70\}$ of Y_6, and compute in the backward direction to get 16 bits of Y_5, i.e. bits $\{3, 7, 11, 15, 19, 23, 50, 54, 58, 62, 66, 70, 74, 78, 82, 86\}$. It is also depicted as the last two round-

s in Figure 2. Further, we guess the 6 bits $\{20, 21, 22, 64, 65, 67\}$ of Y_5, which are input to the inversion of Sboxes S_5 and S_{16}, as shown in Figure 2 in cyan color. In total we get a set of 2^{18} such values of Y_5 and each value has 22 bits determined.

3. For each partially determined value of X_2 and Y_5, repeat the following:
   (a) Divide the bits of X_2 into two overlapping groups, whose intersection is the 8 bits input to Sboxes S_{10} and S_{11}. Then for each group, we guess the 33 undetermined bits of X_2 and 2 bits ($\{10, 11\}$ or $\{66, 67\}$) of X_3, and compute in the forward direction to get 44 bits of X_4. We store in a table $T_{X,i}$, $i = 0, 1$, the values of partially determined X_4 computed from the i-th group together with the 4 bits $\{10, 11, 66, 67\}$ of X_3. See Figure 2 for an example group in orange color, where the guessed 2 bits of X_3 are bits 10 and 11, which are input to Sbox S_5.
   (b) Similarly, divide the bits of Y_5 into two overlapping groups, whose common bits are the 8 bits input to the inversion of Sboxes S_5 and S_{16}. Then for each group, we guess the 33 undetermined bits of Y_5 and 2 bits ($\{41, 43\}$ or $\{44, 46\}$) of Y_4, and compute in the backward direction to get 44 bits of X_4. We store in a table $T_{Y,i}$, $i = 0, 1$, the values of partially determined X_4 computed from the i-th group together with the 4 bits $\{41, 43, 44, 46\}$ of Y_4. See Figure 2 for an example group in orange color, where the guessed 2 bits of Y_4 are bits 44 and 46, which are input to the inversion of Sbox S_{11}.
   (c) Then merge those tables to find a set of fully-determined values of X_4.
      i. Merge the tables $T_{X,i}$ and $T_{Y,i}$ to T_i respectively for $i = 0, 1$. By merging these two tables, we mean to merge every two partially-determined values of X_4, each from a table and sharing the same bit values at the common determined bit positions, into a new partially-determined value of X_4 with all their determined bits, and then to include this new value of X_4 in table T_i. Note that each value of $T_{X,i}$ and each value of $T_{Y,i}$ share 22 determined bit positions. Hence table T_i has on average $2^{2\cdot 35-22} = 2^{48}$ values, each of which has $2 \cdot (44 + 4) - 22 = 74$ bits.
      ii. Merge T_0 and T_1. Notice that T_0 and T_1 share 44 determined bit positions of X_4, 4 determined bit positions of X_3 and 4 determined bit positions of Y_4. Hence we obtain $2^{2\cdot 48-52} = 2^{44}$ values on average, each of which has $2 \cdot 74 - 52 = 96$ bits consisting of the full 88 bits of X_4, 4 bits of X_3 and 4 bits of Y_4.

The algorithm is to find a set of internal state values of X_4, whose corresponding plaintexts can satisfy the constraints on the input and output of the MitM layer. Totally, we obtain on average $2^{2\cdot 18} \cdot 2^{44} = 2^{80}$ plaintexts by inversely computing from the fully-determined values of X_4, which satisfies the constraints on the input and output of the MitM layer.

Complexity. The complexity of the algorithm is dominated by Step 3. Since there are 2^{18} X_2 from the forward computations and 2^{18} Y_5 from the backward computations, Step 3 is executed 2^{36} times. The complexity of each execution is dominated by Step 3(c)ii, that is merging T_0 and T_1, which needs 2^{48} table lookups. Hence the total complexity of Step 3 is 2^{84} table lookups. Once a match of the MitM layer has been found, we can encrypt this value X_4 over the $r_1 + 3$ rounds and increment the counter D given in the previous section. The memory complexity of this attack is dominated by the storage of the table T_0 and T_1 which is $2 \cdot 2^{48} \cdot 74 \approx 2^{55.2}$ bits. To sum up, the total time complexity of the distinguisher is 2^{84} table lookups and 2^{80} permutation queries.

The MitM Layer for All Versions of `SPONGENT`. The 13 versions of `SPONGENT` hash functions use 11 different sizes of internal states. We have shown the MitM layer for `SPONGENT`-88/80/8. The approach also applies to the other versions. Due to the similarity of the idea, we directly provide our results while omitting the details. These results are listed in Table 3, respectively for different sizes of internal states. The numbers of rounds for generic approach are directly derived from the results of Section 3.2, and the corresponding complexities can also be obtained. Notice that by the improved approach, besides the size 88 we also increase two more rounds for the sizes $136, 176, 240, 264, 272, 336$, which are not divided by $2 \cdot 4^2 = 32$. The complexities of these cases are 2^{n-4} table lookups and 2^{n-8} permutation queries.

Table 3. The number of rounds of the MitM layer for SPONGENT

Size of internal state	88	136	176	240	264	272	336	384	480	672	768
#Rounds (generic)	5	5	5	5	7	7	7	11	9	9	11
#Rounds (improved)	7	7	7	7	9	9	9	11	9	9	11

4.3 Summary

We have shown the meet-in-the-middle layers and the truncated differential distinguishers for all versions of SPONGENT permutations. In the truncated differential phase, we take advantage of the strong relation between truncated differential probability and capacity of multidimensional linear approximation to derive the best differential distinguishers, and as a result we reach one, two or three rounds more than the results shown by the designers. In the meet-in-the-middle phase, we get up to 11 rounds to pre-add to the differential distinguishers. Totally, we improve the previous distinguishers on all versions of SPONGENT permutations by up to 13 rounds. The full results are summarized in Table 1, compared with the previous distinguishers.

5 Conclusion

In this paper, we present a general method to distinguish a PRESENT-like permutation with a random permutation. This generic method is a truncated differential distinguisher which includes two layers: a truncated differential layers for describing the collision bias on some predetermined output bits and a MitM layer for extending the number of the attacked rounds without changing the probability of truncated differential. We also estimated the number of attacked rounds of the MitM layer. For a concrete permutation, the estimated bound can possibly be further improved. For example, for SPONGENT-88/80/8 it can be improved to 7 rounds from 5 rounds. As an application, we further show the distinguishers for all the versions of SPONGENT permutations, which improve the previous results by up to 13 rounds.

Acknowledgement

The authors are grateful to Lei Wang for inspiring this work and many helpful suggestions. The authors would also like to thank Jérémy Jean for providing TikZ code of PRESENT block cipher [29]. This work was supported by the National Natural Science Foundation of China (Grant Nos. 61303258, 61379139 and 11526215) and the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06010701.

References

1. De Cannere, Christophe. Trivium: A stream cipher construction inspired by block cipher design principles. Information Security, Springer Berlin Heidelberg, 2006, 4176:171-186.

2. Hell, Martin and Johansson, Thomas and Maximov, Alexander and Meier, Willi. The Grain family of stream ciphers, New Stream Cipher Designs. Lecture Notes in Computer Science, 2008, 4986:179-190.
3. De Canniere, Christophe and Dunkelman, Orr and Knežević, Miroslav. KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. Cryptographic Hardware and Embedded Systems-CHES 2009, Switzerland, 2009, LNCS 5747:272-288.
4. Bogdanov, Andrey and Knudsen, Lars R and Leander, Gregor and Paar, Christof and Poschmann, Axel and Robshaw, Matthew JB and Seurin, Yannick and Vikkelsoe, Charlotte. PRESENT: An ultra-lightweight block cipher. Lecture Notes in Computer Science, 2007, 4727:450-466.
5. Guo, Jian and Peyrin, Thomas and Poschmann, Axel and Robshaw, Matttitle. The LED block cipher. International Conference on Cryptographic Hardware Embedded Systems. Springer Berlin Heidelberg, 1970:326-341.
6. Aumasson J P, Henzen L, Meier W, et al. Quark: A Lightweight Hash[J]. Journal of Cryptology, 2010, 26(2):1-15.
7. Bogdanov A, Knezevic M, Leander G, et al. SPONGENT: A Lightweight Hash Function[J]. Lecture Notes in Computer Science, 2011, 6917:312-325.
8. Bogdanov A, Knezevic M, Leander G, et al. SPONGENT: The Design Space of Lightweight Cryptographic Hashing[J]. Computers IEEE Transactions on, 2013, 62(10):2041-2053.
9. Guo J, Peyrin T, Poschmann A. The PHOTON Family of Lightweight Hash Functions[M]. Advances in Cryptology-CRYPTO 2011. Springer Berlin Heidelberg, 2011:222-239.
10. Borghoff J, Knudsen L R, Leander G, et al. Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes[J]. Lecture Notes in Computer Science, 2011, 6733(2010):270-289.
11. Lauridsen M M, Rechberger C. Linear Distinguishers in the Key-less Setting: Application to PRESENT[M]. Fast Software Encryption. Springer Berlin Heidelberg, 2015:217-240.
12. Abdelraheem M A. Estimating the probabilities of low-weight differential and linear approximations on PRESENT-like ciphers[M]// Information Security and Cryptology-ICISC 2012. Springer Berlin Heidelberg, 2012:368-382.
13. Bulygin, Stanislav. More on linear hulls of PRESENT-like ciphers and a cryptanalysis of full-round EPCBC–96. Cryptology ePrint Archive, Report 2013/028.
14. Nikolic I, Wang L, Wu S. Cryptanalysis of Round-Reduced LED[M]. Fast Software Encryption. Springer Berlin Heidelberg, 2013:112-129.
15. Blondeau C, Peyrin T, Wang L. Known-Key Distinguisher on Full PRESENT[M]. Advances in Cryptology-CRYPTO 2015. Springer Berlin Heidelberg, 2015:455-474.
16. ISO/IEC. Information technology-Security techniques-Lightweight cryptography-Part 2: Block ciphers. ISO/IEC, 2012, 29192-2:2012.
17. Cheng H, Heys H M, Wang C. PUFFIN: A Novel Compact Block Cipher Targeted to Embedded Digital Systems[C]. Proceedings of the 2008 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools. IEEE Computer Society, 2008:383-390.
18. Knudsen L, Leander G, Poschmann A, et al. PRINTcipher: A Block Cipher for IC-Printing[M]. Cryptographic Hardware and Embedded Systems, CHES 2010. Springer Berlin Heidelberg, 2010:16-32.
19. Gomathisankaran M, Lee R B. Maya: A novel block encryption function[J]. Unt Scholarly Works, 2009.
20. Yap H, Khoo K, Poschmann A, et al. EPCBC-A Block Cipher Suitable for Electronic Product Code Encryption[M]. Cryptology and Network Security. Springer Berlin Heidelberg, 2011:76-97.
21. Zhang W, Bao Z, Lin D, Rijmen V, Yang B, Verbauwhede I. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. SCIENCE CHINA Information Sciences, 2015, 58(12): 1-15.
22. ISO/IEC. Information technology-Security techniques-Lightweight cryptography-Part 5: Hash-functions. ISO/IEC DIS, 2015, 29192-5:2015.
23. Cho, Joo Yeon. Linear cryptanalysis of reduced-round PRESENT. Topics in Cryptology-CT-RSA 2010, San Francisco, CA, USA, 2010, LNCS 5985:302-317.
24. Blondeau C, Nyberg K. Links between Truncated Differential and Multidimensional Linear Properties of Block Ciphers and Underlying Attack Complexities[M]. Advances in Cryptology-EUROCRYPT 2014. Springer Berlin Heidelberg, 2014:165-182.
25. Bertoni G, Daemen J, Peeters M, et al. Sponge-Based Pseudo-Random Number Generators[M]. Cryptographic Hardware and Embedded Systems, CHES 2010. Springer Berlin Heidelberg, 2010:33-47.
26. Bertoni G, Daemen J, Peeters M, et al. On the Indifferentiability of the Sponge Construction.[J]. Lecture Notes in Computer Science, 2008, 4965:181-197.
27. Bao Z, Zhang W, Lin D. Speeding Up the Search Algorithm for the Best Differential and Best Linear Trails[M]. Information Security and Cryptology. Springer International Publishing, 2014:259-285.
28. Knudsen L R. Truncated and Higher Order Differentials[J]. Lecture Notes in Computer Science, 1995, 1008:196-211.
29. Jean J. TikZ for Cryptographers. Asiacrypt 2015 Rump Session, 2015. `http://www.di.ens.fr/~jean/latex_crypto/`