Differential Fault Attack on the PRINCE Block Cipher

Ling Song, Lei Hu

State Key Laboratory of Information Security, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing 100093, China
{lsong,hu}@is.ac.cn

Abstract. PRINCE is a new lightweight block cipher proposed at the ASIACRYPT'2012 conference. In this paper two observations on the linear layer of the cipher are presented. Based on the observations a differential fault attack is applied to the cipher under a random nibble-level fault model. The attack uniquely determines the 128-bit key of the cipher using less than 7 fault injections averagely. In the case with 4 fault injections, the attack limits the key to a space of size less than 2^{18} statistically.

Key words: lightweight cipher, PRINCE block cipher, differential fault attack

1 Introduction

The idea of injecting faults during the execution of cryptographic algorithms to retrieve the key was first introduced by Boneh, DeMillo, and Lipton who succeeded in breaking a CRT version of RSA [4]. Later, Biham and Shamir adapted this idea to differential analysis on block ciphers and introduced the concept of Differential Fault Attack (DFA) [2]. Block ciphers implemented on smart cards and other low-end devices are vulnerable to such attacks, which exploit the links between right ciphertexts and the faulty counterparts. Usually the faults are injected by disturbing the power supply voltage, the frequency of the external clock, or by applying a laser beam, etc [1].

Recent years, many lightweight block ciphers have been proposed in the literature to provide cryptographic building blocks for resource constrained devices such as RFID tags. Among the best studied ciphers are PRESENT, KATAN, LED and PRINTCipher [3, 5, 6, 8]. PRINCE is a novel lightweight block cipher proposed in 2012 [7], which is optimized with respect to latency when implemented in hardware. This is the first lightweight block cipher that takes latency as main priority.

In this paper, we present a differential fault attack on PRINCE under a random fault model which adds a random disturbance to a nibble of the state whose position cannot be predicted in advance. Our attack mainly exploits the diffusion property of the linear layer of the PRINCE cipher. With the knowledge of the differential distribution table of the Sbox it used, the number of

survival keys using one fault injection can be evaluated statistically. However, the data complexity of finding the unique key is difficult to estimate due to the uncertainty of the diffusion property of the linear layer. Hence we obtain related results through experiments. The experiments show 4 faults are enough to break PRINCE practically and 7 fault injections uniquely determine the 128-bit key of PRINCE.

This paper is organized as follows: Section 2 briefly describes the PRINCE block cipher; in Section 3 we elaborate on our differential fault attack on PRINCE; Section 4 discusses the results; and finally, we conclude the paper in the last section.

2 Brief Description of PRINCE

PRINCE is a 64-bit block cipher with a 128-bit key. The key schedule is very simple, namely, the 128-bit key is split into two 64-bit parts:

$$k = k_0||k_1,$$

and extended to 192 bits by the following mapping:

$$(k_0||k_1) \rightarrow (k_0||k_0'||k_1) := (k_0||(k_0 \ggg 1) \oplus (k_0 \gg 63)||k_1).$$

During the encryption the first two subkeys k_0 and k_0' are used as whitening keys, while the third subkey k_1 is the key for a 12-round block cipher refered to as PRINCE$_{core}$. The highlevel structure of PRINCE is demonstrated in Fig. 1.

Fig. 1. The highlevel structure of PRINCE

The 12-round process of PRINCE$_{core}$ is depicted in Fig. 2. A typical round of PRINCE$_{core}$ consists of an Sbox layer, a linear layer and an addition layer. The intermediate computation result, called state is usually represented by a 64-bit vector or a 16-nibble vector.

Sbox-layer. The cipher uses a 4-bit Sbox which is given as in Table 1. We denote the Sbox and its inverse by S and S^{-1} respectively.

Table 1. The Sbox S of PRINCE

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	B	F	3	2	A	C	9	1	6	7	8	0	E	5	D	4

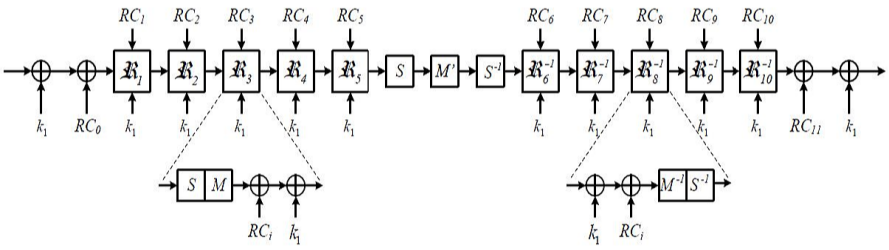

Fig. 2. PRINCE*core*

Linear layer. As can be seen from Fig. 2, the linear layer uses a matrix M or M' and is called M- or M'-mapping according to the matrix used. In the linear layer the 64-bit state is multiplied with M or M', both of which are 64×64 matrices and built from four 4×4 matrices. These four matrices are

$$M_0 = \begin{bmatrix} 0\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}, M_1 = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}, M_2 = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,1 \end{bmatrix}, M_3 = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,0 \end{bmatrix}.$$

Two block matrices $\hat{M}^{(0)}$ and $\hat{M}^{(1)}$ of size 16×16 are generated as follows:

$$\hat{M}^{(0)} = \begin{bmatrix} M_0 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \end{bmatrix}, \hat{M}^{(1)} = \begin{bmatrix} M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \\ M_0 & M_1 & M_2 & M_3 \end{bmatrix}.$$

Then, the 64×64 matrix M' is constructed as a block diagonal matrix with $\hat{M}^{(0)}, \hat{M}^{(1)}, \hat{M}^{(1)}, \hat{M}^{(0)}$ as its diagonal blocks. Note that M' is an involution matrix, namely, $M'M' = I$ is the identity matrix.

The M-mapping is the composition of the M'-mapping and a permutation SR, i.e. $M = SR \circ M'$. SR behaves like the AES shift rows and permutes the 16 nibbles of the state as $(a_0, a_1, \cdots, a_{15}) \rightarrow (a_0, a_5, \cdots, a_{11})$, where the subscripts are changed according to Table 2. Also, the inverse of SR is denoted by SR^{-1} for the sake of simplicity.

Table 2. The SR operation of PRINCE

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11

Addition. The 64-bit state is xored with the 64-bit subkey k_1 and a round-dependent constant as listed in Table 3. Note that the 12 round constants have a

symmetric property: for all $0 \leq i \leq 11, RC_i \oplus RC_{11-i} = 0\text{X}c0ac29b7c97c50dd(=: \alpha)$. With this property and together with the fact that M' is an involution matrix, the encryption and decryption of the cipher have the following relationship:

$$D_{(k_0||k_0'||k_1)}(\cdot) = E_{(k_0'||k_0||k_1 \oplus \alpha)}(\cdot).$$

Thus, a same hardware implementation can fulfill both encryption and decryption operations of the PRINCE cipher.

Table 3. Round constants

RC_0	0000000000000000
RC_1	13198a2e03707344
RC_2	a4093822299f31d0
RC_3	082efa98ec4e6c89
RC_4	452821e638d01377
RC_5	be5466cf34e90c6c
RC_6	7ef84f78fd955cb1
RC_7	85840851f1ac43aa
RC_8	c882d32f25323c54
RC_9	64a51195e0e3610d
RC_{10}	d3b5a399ca0c2399
RC_{11}	c0ac29b7c97c50dd

More details about this cipher can be found in [7].

3 Attacking PRINCE

In this section the fault model is stated first. Then we describe our two observations. Exploiting these observations, our attack is elaborated afterwards.

3.1 Fault Model

We are dealing with random faults on a single nibble whose position cannot be predicted in advance. The effect of the introduced fault is to add an arbitrary nonzero nibble disturbance to the state. Thus, faults can be induced within nibble-wise operations including Sbox substitution, SR operation and addition, which is beneficial to fault injections.

Although PRINCE may not be implemented in a round-based fashion, we assume an attacker can typically predict when a particular round happens and induce a nibble fault at a specific round. Moreover, the time that certain events take place can often be determined by analyzing a suitable side channel leakage. Furthermore, we assume that an attacker can repeat the experiments with the same plaintext and key without applying external physical effects.

In the remaining part of this paper, a 16-nibble X is represented with $(X_0, X_1, \cdots, X_{15})$ and we always denote by (C, C^*) a pair of a right ciphertext C and its corresponding faulty ciphertext C^* for the same plaintext and key.

3.2 Observations

Before going to the details of our observations, we split the 16 nibbles of the state of PRINCE into four groups numbered from 1 to 4 as depicted in Fig. 3.

Fig. 3. Split the nibbles into four groups

Diffusion property of the M'-mapping. Set $X = (X_0, X_1, \cdots, X_{15})$ and $Y = (Y_0, Y_1, \cdots, Y_{15})$ to be the input and the corresponding output of the M'-mapping.

First, the M'-mapping diffuses the nibbles within groups. If only a certain group of X has nonzero nibbles, then only the same group of Y has nonzero nibbles. Hence the M'-mapping of the 64-bit state can be regarded as four small separate mappings M'_1, M'_2, M'_3, and M'_4, each of which diffuses the nibbles of the corresponding group.

Second, the M'-mapping achieves an almost-MDS property. If X has only one nonzero nibble, say X_2 (belongs to Group 1), Y will have at most 4 nonzero nibbles, all of which are located in the same group (Group 1). Concretely speaking, if the Hamming weight of X_2 is greater than 1, then all the four nibbles of Group 1 of Y are nonzero; otherwise, exactly three of them are nonzero.

Diffusion property of the SR. Set $X = (X_0, X_1, \cdots, X_{15})$ and $Y = (Y_0, Y_1, \cdots, Y_{15})$ to be the input and the corresponding output of the SR operation. If X has a group of four non-zero nibbles, then Y will still have four non-zero nibbles, each of which is located in a different group, i.e. SR diffuses the nibbles over groups.

3.3 Principle of the Attack at the 11-th Round

Our attack is based on the induction of a nibble fault at the 10-th round under the model mentioned above. To explain the attack, first let us consider the scenario when there is a nibble disturbance at the 11-th round.

For the three operations marked in Fig. 4(a) where we want to inject a fault, the attack works the same in principle. Below we suppose faults are injected during the Sbox substitution of 11-th round.

Assume we get a right ciphertext C and its corresponding faulty ciphertext C^* for the same plaintext and key. The fault can happen at any position of the 16 nibbles. For the sake of simplicity, we take the first nibble as a faulty nibble and analysis for other positions are the same.

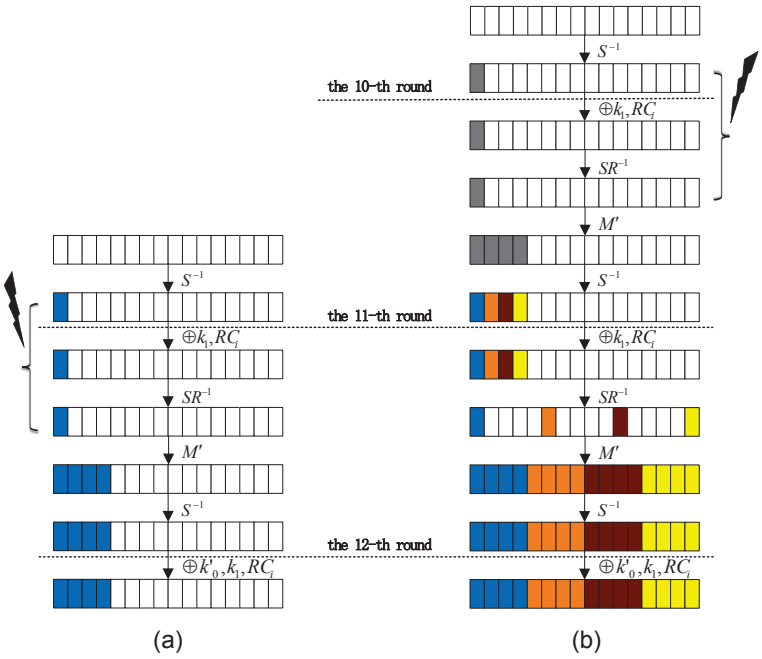

Fig. 4. Attack PRINCE with fault at 11-th round and 10-th round

As illustrated in Fig. 4(a), the fault injected in the first nibble during the Sbox substitution of 11-th round influences the first group of the 16 nibbles of the final ciphertext due to the diffusion property of M'-mapping.

In this context, first four nibbles C_0, C_1, C_2, C_3 and $C_0^*, C_1^*, C_2^*, C_3^*$ are known, and so is the fact that the the bitwise Exclusive-or (XOR) differences of them stem from a single nibble induced by the fault.

Let us look into the first nibble. The C_0 and C_0^* are known. Given the input difference of the first Sbox Δ_0^{in}, with the knowledge of the differential distribution table of the S^{-1} of PRINCE (see Table 4) in mind, the first nibble of $K = k_0' \oplus k_1$ will be limited to one of 0, 2, or 4 choices by the following equation [9]:

$$S(C_0 \oplus K_0) \oplus S(C_0^* \oplus K_0) = \Delta_0^{\text{in}}.$$

To get information about all the first four-nibble of $K = k_0^{'} \oplus k_1$, we can guess $(\Delta_0^{\text{in}}, \Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_3^{\text{in}})$, the input difference of the first four Sboxes and then search the subkey information. Before searching, it is necessary to check whether the guesses satisfy the following two conditions which we call the ***M'-Mapping Conditions***.

- Non-zero Δ_i^{in}s are valid differences that can lead to the right output differences.
- The preimage of $(\Delta_0^{\text{in}}, \Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_3^{\text{in}})$ under the corresponding submapping of M'-mapping has only one nonzero nibble.

Table 4. Differential distribution table of the S^{-1} used by PRINCE.

Δin	Δout															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	2	0	2	0	0	0	0	0	0	2	4	2	0	0
2	0	0	0	0	2	2	2	2	2	2	0	0	0	0	2	2
3	0	0	4	0	4	2	2	0	0	2	2	0	0	0	0	0
4	0	2	0	0	2	0	0	0	4	0	2	4	0	0	0	2
5	0	0	0	2	2	2	2	0	2	0	4	0	2	0	0	0
6	0	2	0	2	0	0	2	2	0	0	0	0	2	0	4	2
7	0	0	2	0	0	2	0	0	0	0	4	2	0	2	2	2
8	0	4	2	2	2	0	0	2	2	0	2	0	0	0	0	0
9	0	2	0	2	0	2	2	0	2	2	0	0	0	4	0	0
A	0	0	0	2	2	0	0	4	0	2	0	0	2	2	0	2
B	0	2	0	2	0	2	2	0	2	0	0	2	2	0	2	0
C	0	0	0	2	0	2	0	0	0	4	0	2	2	0	2	2
D	0	0	4	0	0	2	0	2	2	2	0	0	0	2	2	0
E	0	0	2	0	0	0	4	2	0	0	0	2	2	2	0	2
F	0	0	0	2	0	0	0	2	0	2	2	2	0	2	2	2

A guess cannot be called a right guess until it passes the M'-mapping Conditions. Below a right guess's four-nibble preimage under the corresponding submapping of M'-mapping is denoted by P.

Now consider the number of suggested four-nibble key values given one right four-nibble guess. Note that a pair of nonzero input/output differences of a single Sbox suggests 2 or 4 values (16/7 in average) for a subkey nibble, and zero differences suggest all possible values for a subkey nibble. As a result, one pair of input/output differences of four Sboxes suggests $2^4 \sim 2^{10}$ values for the four-nibble subkey used, where $2^{10} = 4 \cdot 4 \cdot 4 \cdot 16$ since there is at least three nonzero differential nibbles. According to the property of the M'-mapping, the number of values suggested for a four-nibble subkey in average is

$$\frac{11}{15} \times \left(\frac{16}{7}\right)^4 + \frac{4}{15} \times \left(\frac{16}{7}\right)^3 \times 16 \approx 71.$$

Therefore, such a pair (C, C^*) reduces the size of the four-nibble subkey space from 2^{16} to $t_1 \cdot 71$, where t_1 is the number of right four-nibble guesses of the input difference of the Sbox layer.

If we want to uniquely determine the first four nibbles of $k_0^{'} \oplus k_1$, at least another fault leading to non-zero ciphertext difference at first four nibbles is needed. Hence, the whole nibbles of $k_0^{'} \oplus k_1$ can be uniquely determined with at lest $4 \times 2 = 8$ fault injections, each of which leaks some information of a group of nibbles of the 64-bit key.

After $k_0^{'} \oplus k_1$ has been recovered, the last round can be peeled off, and the attack is repeated on the reduced cipher to reveal k_1.

3.4 Attack Strategy at the 10-th Round

In this subsection our target fault attack on PRINCE is described on the basis of the principle elaborated in the previous subsection, aiming to use less faults as possible.

The attack scenario remains except that faults are injected one round earlier, as depicted in Fig. 4 (b). Set the first nibble to be the faulty nibble again (other cases work the same).

Suppose that the induced fault difference has a Hamming weight greater than 1 and the opposite case will be discussed later. As demonstrated by Fig. 4 (b), the difference keeps until it goes into the M'-mapping of the 11-th round. The M'-mapping spreads the difference to the whole group, and then the four nibble differences are changed by the S^{-1}. In order to make a distinction among the four nibble differences, we color them differently. The SR^{-1} of the 12-th round splits the four nibble differences into different groups, making each group have one and only one nonzero nibble of difference. After that, M'-mapping propagates differences within groups, resulting full difference in the ciphertext.

Given a pair (C, C^*) whose fault difference propagation follows the pattern depicted in Fig. 4(b), the analysis is sketched below.

1. For group $i, 1 \leq i \leq 4$, guess $(\Delta^{\text{in}}_{4i}, \Delta^{\text{in}}_{4i+1}, \Delta^{\text{in}}_{4i+2}, \Delta^{\text{in}}_{4i+3})$. For those that satisfy the M'-Mapping Conditions, store $(P_i, (\Delta^{\text{in}}_{4i}, \Delta^{\text{in}}_{4i+1}, \Delta^{\text{in}}_{4i+2}, \Delta^{\text{in}}_{4i+3}))$ in table T_i, where P_i is the four-nibble preimage of the corresponding guess.
2. After we get such four tables, search four-nibble subkey values using the items in Table T_i, $1 \leq i \leq 4$ as we do in Section 3.3.
3. Using the P_is in four tables $T_i, 1 \leq i \leq 4$, check whether the concatenations of $P_1||P_2||P_3||P_4$ satisfy the SR **Condition**, which is defined as the four non-zero nibbles need to gather together in a single group after the SR operation. For those concatenations that pass the SR Condition, record (P_1, P_2, P_3, P_4) in table D.
4. To get candidates of 64-bit key, concatenate the four-nibble subkey values suggested by (P_1, P_2, P_3, P_4), the items of D.
5. Inject more faults and repeat previous steps to reduce the space of the 64-bit key.

Since 71 four-nibble subkey values in average is returned by a pair of input/output differences of a group, $t_2 \cdot 71^4 = t_2 \cdot 2^{24.60}$ values of 64-bit key will be obtained with one fault injected at round 10, where t_2 is the number of items in list D. In this context, 64-bit key information are interrelated with one fault, and hence at least 2 fault are needed to recover 64-bit subkey information.

For the fault difference with Hamming weight equal to 1, less information can be obtained, since the fault difference propagates to only thress nibbles within the same group after the M'-mapping of 11-th round. The following SR^{-1} operation then scatters the three non-zero nibbles into different groups, resulting differences in only three groups of nibbles in the ciphertext. In this case, the 64-bit key space is reduced approximately to a size of $t_2 \cdot 71^3 \cdot 2^{16} = t_2 \cdot 2^{34.45}$.

Note that the induced fault difference has a Hamming weight greater than 1 with probability 11/15 and for the other case the probability is 4/15. Considering these two cases together, the number of possible values for the 64-bit key is reduced to

$$t_2 \cdot \left(\frac{11}{15} \cdot 71^4 + \frac{4}{15} \cdot 71^3 \cdot 2^{16} \right) = t_2 \cdot 2^{32.55}$$

with one fault injection.

Once the outer 64-bit subkey has been recovered, the last round can be peeled off and the inner 64-bit subkey will be retrieved in a more efficient way.

For the recovery of the inner 64-bit key k_1, the M'-Mapping Conditions can be applied to the items in D after Step 3. In other words, the difference of the states after decrypting two rounds back needs to have only one non-zero nibble[1]. In this way more wrong keys can be excluded.

For a group of four nibbles (all of them are nonzero or three of them are nonzero), it satisfies the M'-Mapping Condition with probability of

$$\frac{4 \times 15}{\frac{11}{15} \times 15^4 + \frac{4}{15} \times 15^3} = 2^{-9.31}$$

Consequently, using one fault injection the number of suggested values for the subkey k_1 is about

$$t_2 \cdot 2^{32.55} \cdot 2^{-9.31} = t_2 \cdot 2^{23.24}.$$

This is verified by the experimental results in the next section.

4 Results

In fact, random faults introduce differences with any Hamming weight. Due to the uncertainty of the fault difference and the almost-MDS diffusion property of M'-mapping, it is difficult to estimate accurately the number of fault injections needed to uniquely determine the 128-bit key. In this section not only the estimated number of survival keys using one fault injection is verified, but also the data complexity is evaluated by experiments on a PC.

According to the experiments, t_1 ranges from 1 to 6 and has an average of 2.50, while $t_2 \in [1, 12]$ and achieves 1.88 on average. The result in Table 5, which is derived statistically from 1000 instances, shows that using 4.05 fault injections in average, the outer 64-bit key will be determined uniquely. In addition, if only 2 fault injections are used, the number of candidates for the outer 64-bit keys will be $2^{14.18}$. Similarly, the unique determination of the inner 64-bit key needs 2.56 fault injections in average and 2 fault injections reduce the size of the 64-bit key space to 2^3, as shown in Table 6. Note that $2^{34.76}$ and $2^{25.32}$, which are the experimental number of suggested values for the 64-bit subkeys using one fault injection, are very close to our estimates.

[1] This cannot be applied to the recovery of outer 64-bit subkey since the decryption needs subkey k_1 which is totally unknown.

The time complexity mainly lies in the computation of Step 2 in the algorithm of Section 3.4. Since the computations for each group are processed separately and parallel, the time complexity is is very small, which is around 2^{16}.

Table 5. Recovery of outer 64-bit key of PRINCE

#faults	1	2	3	4	4.05
#survival	$2^{34.76}$	$2^{14.18}$	28.94	3.53	1

Table 6. Recovery of inner 64-bit key of PRINCE

#faults	1	2	2.56
#survival	$2^{25.32}$	8.03	1

5 Conclusion

In this paper we broke PRINCE with differential fault attack under the random fault model which adds a nibble of disturbance to the state of the cipher. Experiments showed that the 128-bit key can be uniquely determined using 6.61 fault injections. Also, the key space can be reduced to a size of $2^{17.18}$ with 4 fault injections. To our knowledge, this is the first differential fault attack on PRINCE.

References

1. Bar-el, H., Choukri, H., Naccache, D., Tunstal, M., Whelan, C.: *The Sorcerers Apprentice Guide to Fault Attacks.* Proceedings of the IEEE, 94(2), 370-382 (2006)
2. Biham, E., Shamir, A.: *Differential Fault Analysis of Secret Key cryptosystem.* In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513-525. Springer, Heidelberg (1997)
3. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: *PRESENT: An Ultra-Lightweight Block Cipher.* In: Paillier, P., Verbauwhede, I. (ed.) CHES 2007. LNCS, vol. 4727, pp. 450-466. Springer, Heidelberg (2007)
4. Boneh, D., DeMillo, R. A., Lipton, R. J.: *On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract).* In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37-51. Springer, Heidelberg (1997)
5. De Canniére, C., Dunkelman, O., Knežević, M.: *KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers.* In: Clavier, C., Gaj, K. (ed.) CHES 2009. LNCS, vol. 5747, pp. 272-288. Springer, Heidelberg (2009)
6. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: *The LED Block Cipher.* In: Preneel, B., Takagi, T. (ed.) CHES 2011. LNCS, vol. 6917, pp. 326-341. Springer, Heidelberg (2011)

7. Julia, B., Anne, C., et. al.: *PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications*. In: Wang, X., Sako, K. (ed.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208-225. Springer, Heidelberg (2012)
8. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: *PRINTcipher: A Block Cipher for IC-Printing*. In: Mangard, S., Standaert, F.-X. (ed.) CHES 2010. LNCS, vol. 6225, pp. 16-32. Springer, Heidelberg (2010)
9. Nyberg, K.: *Differentially Uniform Mappings for Cryptography*. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 55-64. Springer, Heidelberg (1994)