Improved Differential Fault Attack on the Block Cipher SPECK

Yuming Huo
State Key Laboratory of Information Security & Data Assurance and Communication Security Research Center
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
huoyuming@iie.ac.cn
Fan Zhang
Key Laboratory of Mathematics Mechanization
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
Beijing, China
tzhangfan4@163.com
Xiutao Feng
Key Laboratory of Mathematics Mechanization
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
Beijing, China
State Key Laboratory of Information Security
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
fengxt@amss.ac.cn
Li-Ping Wang
State Key Laboratory of Information Security & Data Assurance and Communication Security Research Center
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
wangliping@iie.ac.cn

***Abstract*—SPECK is a family of lightweight block ciphers published by the U.S. National Security Agency in 2013. The SPECK family consists of 10 versions, supporting a wide range of block and key sizes. Recently H. Tupsamudre *et al.* gave an approach to recovering the last round key of SPECK family with $n/3$ fault injections, where $2n$ is the block size. In this paper, we present two improved differential fault attacks on the SPECK family under different fault models. The first attack assumes a more practical random fault model and recovers the last round key with about $5 \sim 8$ fault injections on all versions with different block sizes, which is far less than that of H. Tupsamudre *et al.*'s attack. The second attack only requires 4 specific faults to recover the last round key (except the most significant bit) over an arbitrary block size under a chosen-value fault model.**

***Keywords*-Block Ciphers; SPECK; Differential Fault Attack; Gröbner Bases.**

I. Introduction

The idea of injecting faults during the execution of cryptographic algorithms to retrieve the key was first introduced by D. Boneh, R.A. DeMillo and R.J. Lipton who succeeded in breaking a CRT version of RSA [1], [2]. Shortly after, an adaptation of this idea on block ciphers was proposed by E. Biham and A. Shamir and the concept of Differential Fault Attack (DFA) was further introduced [3]. Application of this principle to block ciphers implemented on smart cards and other low-end devices is a real threat [4], [5], [6], [7]. By disturbing the power supply voltage, the frequency of the external clock, or by applying a laser beam, a fault can be induced with some probability during the computation [8].

Recent advances in tiny computing devices, such as RFID and sensor network nodes, give rise to the need of symmetric encryption with highly limited resources, called lightweight encryption. Among the best studied ciphers are PRESENT, PRINCE, KATAN and LED, etc [9], [10], [11], [12]. In 2013 the U.S. National Security Agency published the SIMON and SPECK families of lightweight block ciphers [13], where SIMON is optimized for hardware and SPECK is optimized for software implementations. Each of them contains 10 versions and supports block sizes ranging from 32 to 128 bits and key sizes ranging from 64 to 256 bits. A recent fault analysis on SPECK is presented by H. Tupsamudre *et al.* [14]. Their attack on SPECK assumes a bit-flip fault model and recovers the n-bit last round key using $n/3$ bit faults on average, where $2n$ is the block size.

In this paper we focus on the SPECK family and present two improved differential fault attacks on SPECK under different fault models. The first attack is under a random word fault model where several bits in the internal state of a cipher were changed into unknown random values at unknown random positions within a word. Inspired by [15],

978-1-4673-7579-5/15 $31.00 © 2015 IEEE
DOI 10.1109/FDTC.2015.15

[16], we transfer the addition modulo 2^n into an algebraic system of equations of maximum degree 2 without any carry variables. Then we apply the computer algebra system Singular [17] to compute a Gröbner bases of the equation system and recover the last round key. The number of injected faults in this attack is about $5 \sim 8$ over the SPECK family, which is far less than $n/3$ presented in [14]. The second attack assumes a chosen-value fault model which the injected fault is specified. Under this fault model, we demonstrate that 4 faults are enough to recover the last round key (except the most significant bit) of SPECK despite of the block size.

The rest of this paper is organized as follows. In Section 2 we introduce some preliminaries including notations, Gröbner bases and algebraic representation of the modular addition. In Section 3 we recall the block cipher SPECK briefly and describe the property of last round function. In Section 4 we elaborate on our two differential fault attacks on SPECK. Finally, we conclude the paper in Section 5.

II. Preliminaries

A. Notations

Here we first introduce some notations, which will be used in the remaining part of this paper.

T	Total number of rounds in the cipher.
(x^0, y^0)	Input to the cipher.
(x^{i+1}, y^{i+1})	Output of the i^{th} round, $0 \leq i < T$.
(x^{i+1*}, y^{i+1*})	Faulty output of the i^{th} round, $0 \leq i < T$.
x_i	The i^{th} least significant bit of x.
xy	The logical AND operation of x and y.
k^i	The i^{th} round key of the cipher, $0 \leq i < T$.
$\lll$	Circular left rotation operation.
$\ggg$	Circular right rotation operation.
$\boxplus$	Addition in modulo 2^n.
$\oplus$	Bitwise logical XOR operation.
$+$	Addition in modulo 2.

B. Gröbner bases

In this section a brief introduction about Gröbner bases is presented. Please refer to [18] for details.

Let Z be the integer ring, F be a field and $F[x_1, x_2, ..., x_n]$ be the n-variables polynomial ring over F, where n is a positive integer. For an n-tuple $\alpha = (\alpha_1, \ldots, \alpha_n)$ in Z^n, we write x^α for $x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}$ to represent a monomial. We should notice that a constant monomial is also included and it is represented by $(0, 0, ..., 0)$.

Definition 1: (monomial ordering) A monomial ordering on the set of all the monomials in $F[x_1, ..., x_n]$ is a total order on monomials that is well defined (a well-ordering) and compatible with multiplication. For example, we use the following notation to represent a lexicographical ordering: $x^\alpha <_{lex} x^\beta$ if $\alpha \neq \beta$ and the first non-zero term in $(\beta_1 - \alpha_1, \cdots, \beta_n - \alpha_n)$ is positive.

In addition, we write $<_{lex}$ as $<$ for short under the assumption of no confusion. For a polynomial p, we call the monomial in p with the highest order with regard to $<$ the leading monomial of p.

Definition 2: [18] (Gröbner bases) Let $<$ be a monomial ordering on $F[x_1, x_2, ..., x_n]$ and $\mathcal{I} \subset F[x_1, x_2, ..., x_n]$ be an ideal. A Gröbner bases of $\mathcal{I}$ is a finite set of generators $g_1, g_2, ..., g_m$ of $\mathcal{I}$ such that the leading monomial of every polynomial $p \in \mathcal{I}$ is a multiple of the leading monomial of some generator g_k, where $1 \leq k \leq m$.

Theorem 1: [18](Elimination property) Let $<$ be the lexicographical ordering on $F[x_1, x_2, ..., x_n]$ with $x_1 < x_2 < \cdots < x_n$. Let $\mathcal{I}$ be an ideal and $\mathcal{G}$ be a Gröbner bases of $\mathcal{I}$ for $<$. Then for any $1 \leq i \leq n$, $\mathcal{G} \cap F[x_1, ..., x_i]$ is a Gröbner bases of the ideal $\mathcal{I} \cap F[x_1, ..., x_i]$.

Singular [17] is a computer algebra system to compute the Gröbner bases of the algebraic system. In this paper we will use the function "$slimgb$" and "$elim$" (which needs the library "$elim.lib$"). Let $\mathcal{I}$ be an ideal. The $slimgb$ and $elim$ are used to computed the Gröbner bases $\mathcal{G}$ and $\mathcal{G} \cap F[x_1, ..., x_i]$ for some $0 \leq i \leq n$ respectively.

C. Algebraic representation of the modular addition

Let $X = (x_{n-1}, ..., x_0)$, $Y = (y_{n-1}, ..., y_0)$ and $Z = (z_{n-1}, ..., z_0)$ be the n-bit words, where 0 indicates the index of the least significant bit. The addition modulo a power of 2 noted by $X \boxplus Y = Z$ can be converted to an algebraic representation over the binary field GF(2), which is a bitwise equation system without carry variables [15]. The specific algebraic representation of $X \boxplus Y = Z$ over GF(2) is

$$\begin{cases} z_0 = x_0 + y_0 \\ z_1 = x_1 + y_1 + x_0 y_0 \\ z_2 = x_2 + y_2 + x_1 y_1 + (x_1 + y_1)(x_1 + y_1 + z_1) \\ \quad \vdots \\ z_i = x_i + y_i + x_{i-1} y_{i-1} + (x_{i-1} + y_{i-1})(x_{i-1} + y_{i-1} + z_{i-1}) \\ \quad \vdots \\ z_{n-1} = x_{n-1} + y_{n-1} + x_{n-2} y_{n-2} \\ \qquad + (x_{n-2} + y_{n-2})(x_{n-2} + y_{n-2} + z_{n-2}) \end{cases} \tag{1}$$

In the following, we define $b_i = x_i + y_i + z_i$ for simplicity.

III. Description of SPECK

In this section we give a brief description of SPECK. More details can be found in [13].

The block cipher SPECK is a Feistel-like structure with a $2n$-bits state, where n is the internal word size and can be taken 16, 24, 32, 48, or 64. SPECK$2n$ with an $m \times n$-bit key is identified with a SPECK$2n/mn$ label, where $m = 2, 3, 4$, and is defined with rotation constants α and β. There are 10 suggested versions with different numbers T of rounds. The detail versions of SPECK are listed in Table I. All versions of SPECK use a similar round function.

Table I
THE SPECK FAMILY

SPECK $2n/mn$	Word Size n	Key Words m	Round T	α	β
32/64	16	4	22	7	2
48/72	24	3	22	8	3
48/96	24	4	23	8	3
64/96	32	3	26	8	3
64/128	32	4	27	8	3
96/96	48	2	28	8	3
96/144	48	3	29	8	3
128/128	64	2	32	8	3
128/192	64	3	33	8	3
128/256	64	4	34	8	3

A. Round function

For providing excellent performance in both hardware and software, SPECK utilizes an extremely simple round function which is iterated over many rounds. The encryption function of SPECK makes use of the bitwise XOR, addition modulo 2^n and left and right circular shifts on a plaintext of two n-bit words $P = (x^0, y^0)$ to encrypt into a ciphertext $C = (x^T, y^T)$. The detail of the encryption function is shown below:

for $i = 0...T-1$ **do**
$x^{i+1} \leftarrow ((x^i \ggg \alpha) \boxplus y^i) \oplus k^i$
$y^{i+1} \leftarrow (y^i \lll \beta) \oplus x^{i+1}$
end for

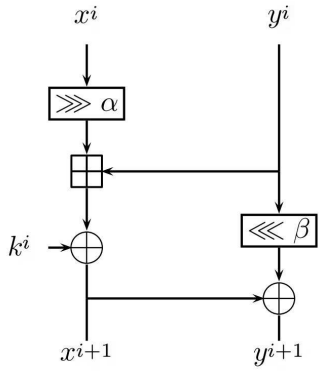

Figure 1. The round function of SPECK

B. The property of the last round function

In Figure 1, the last round function of SPECK consists of the bitwise XOR, the addition modulo 2^n and rotations on two n-bit words. The addition modulo 2^n has two n-bit inputs and one n-bit output, denoted by X, Y and Z respectively. Thus, we have the following equation:

$$X \boxplus Y = Z \tag{2}$$

It is noticed that $Y = y^{T-1}$ and $X = x^{T-1} \ggg \alpha$ in the last round. With the structure of the last round function, we have

$$y^{T-1} = (x^T \oplus y^T) \ggg \beta \tag{3}$$
$$k^{T-1} = x^T \oplus Z \tag{4}$$

Since (x^T, y^T) is the ciphertext, y^{T-1} is known by (3). If we know Z or X, we can obtain the last round key k^{T-1} by (4). In the next section, we will describe two fault attacks that target the output of the penultimate round and retrieve X in order to recover k^{T-1}.

IV. IMPROVED FAULT ATTACKS ON THE SPECK FAMILY

In this section we will present two fault attacks on SPECK under different fault models. The two fault models require that the injected faults lie in the beginning of the last round, that is to say, the faults occur in x^{T-1} or y^{T-1}.

A. The first fault attack under the random word fault model

Our fault model only requires that the fault injection occurs in y^{T-1}. In particular, we induce r random faults at y^{T-1} and get the fault ciphers, then construct the algebraic system with these fault ciphers and the correct cipher. By converting each equation of the algebraic system into the algebraic representation over $GF(2)$ described in section 2.3, the algebraic system can be solved by Singular.

1) Construction of the algebraic system: If r faults are injected into y^{T-1} under the same plaintext and the key, the r corresponding fault values are denoted by $y_k^{T-1^*}$, $1 \le k \le r$. For the sake of simplicity, we write Y_0 for y^{T-1} and Y_k for $y_k^{T-1^*}$, $1 \le k \le r$, in the rest of this paper. So we have the following system:

$$\begin{cases} X \boxplus Y_0 = Z \\ X \boxplus Y_1 = Z \oplus \Delta Z_1 \\ X \boxplus Y_2 = Z \oplus \Delta Z_2 \\ \quad \cdots \ \cdots \\ X \boxplus Y_r = Z \oplus \Delta Z_r \end{cases} \tag{5}$$

where $Y_k = (x_k^{T^*} \oplus y_k^{T^*}) \ggg \beta$ and $\Delta Z_k = x_k^T \oplus x_k^{T^*}$ are known for $1 \le k \le r$. Note that if we can solve X from the above system, then the last round key k^{T-1} is directly determined by (4).

2) Solving the algebraic system by Gröbner bases: Similarly to Equation (1), we first convert each equation of (5) into the algebraic representation over $GF(2)$, and denote by $\mathcal{S}$ the new algebraic system obtained finally. In $\mathcal{S}$ the unknown variables include z_i and x_i, $i = 0, 1, \cdots, n-1$. We define the ordering $<$ as follows:

$$x_0 < x_1 < \cdots < x_{n-1} < z_0 < z_1 < \cdots < z_{n-1}$$

Let $U = (z_{n-1}, \cdots, z_0, x_{n-1}, \cdots, x_0)$, and GF(2)$[U]$ be the polynomial ring over $GF(2)$. Here the monomial ordering is the lexicographical ordering of U. Let G be the Gröbner bases of $\mathcal{S}$ with regard to $<$ and $G' = \mathcal{S} \cap$ GF(2)$[x_0, x_1, \cdots, x_{n-1}]$. Now G and G' can be computed

by Singular. In [16], B. Debraize *et al.* counted the average number of linear polynomials occurring in G' under the random fault model, and gave the number of fault injections. In this paper, both linear and nonlinear polynomials can be used to decrease the size of the possible value space of unknown bits, and the remain uncertain bits can be exhausted.

For a given plaintext (x^0, y^0) and the corresponding ciphertext (x^T, y^T), the specific procedure to recover the last round key is given below:

1. Inject r faults at y^{T-1} under the same plaintext and secret key so that it results in r faulty ciphertexts $(x_k^{T^*}, y_k^{T^*})$, where $1 \leq k \leq r$,
2. Construct the equation system (5) by computing Y_k and ΔZ_k,
3. Convert (5) to the algebraic system $\mathcal{S}$ according to (1),
4. Use the function "slimgb" and "elim" in Singular to get G',
5. Determine the bits in X by solving G',
6. The remain uncertain bits in X are determined by exhausting and checking with the equation system (5),
7. With the knowledge of X, k^{T-1} is recovered by (4).

3) Simulation experiments: In this section we will simulate the solution of the algebraic system $\mathcal{S}$ on a PC(CPU:Intel i7, Memory: 8G). The determination of r is under a large number of experiments. The result of each experiment is the number of linear polynomials in G'. As the number of linear polynomials is large, the uncertainty of X is small. For each experiment, we set $r = \lfloor \log_2 n \rfloor$ and choose random n-bit X, Y and $Y_1, \cdots, Y_r$. Then $Z, \Delta Z_1, \cdots, \Delta Z_r$ are computed directly. It is followed by the construction of system $\mathcal{S}$. If the computation of G' is slow or the number of linear polynomials in G' is less than $n/2$, we increase r by 1 and repeat the above process. In the end, the relationship between r and n is shown in Table II, and the results of 1000 experiments of each (n, r) are shown in Figure 2. One experiment instance is shown in Appendix A.

Table II
THE VALUES OF n AND r

word size n	number of faults r
16	5
24	5
32	6
48	7
64	8

For each instance of (n, r) in Table II, it is noticed that the time of solving $\mathcal{S}$ by Singular is less than 5 seconds, even when $n = 64$. We compute the mean and variance of these data, which is shown in Table III. From Table III, we can see the mean is close to n and the variance is low. It implies that the selection of r is appropriate. Hence, the number of faults r is far less than $n/3$ in [14].

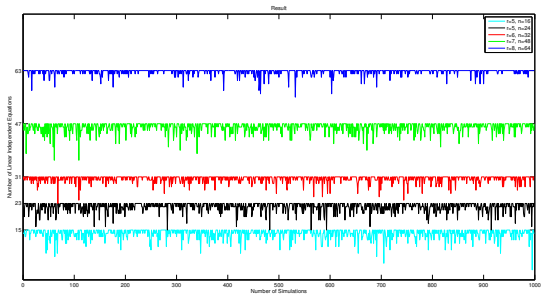

Figure 2. Results of 1000 experiments, the vertical axis represents the number of linear polynomials in Gröbner bases G'.

Table III
THE MEAN AND VARIANCE OF THE RESULTS OF 1000 EXPERIMENTS

Fault r , Word size n	average number of linear independent equations	Variance of linear independent equations
r=5,n=16	13.9220	2.1899
r=5,n=24	21.9740	2.0153
r=6,n=32	30.4030	1.1346
r=7,n=48	45.9310	2.2462
r=8,n=64	62.6370	0.8372

B. The second fault attack under the chosen-value fault model

At present, our fault model requires that the injected fault is chosen at the beginning of last round, i.e., x^{T-1} or y^{T-1}. The effect of the injected fault is to add a difference to x^{T-1} or y^{T-1}. Below we assume that X, Y, Z are the same as defined above, and consider a general system with r fault injections in the output of the penultimate round.

$$\begin{cases} X \boxplus Y = Z \\ (X \oplus A^{(1)}) \boxplus (Y \oplus B^{(1)}) = Z \oplus Z^{(1)} \\ \cdots \quad \cdots \\ (X \oplus A^{(r)}) \boxplus (Y \oplus B^{(r)}) = Z \oplus Z^{(r)} \end{cases} \tag{6}$$

where $(A^{(k)}, B^{(k)}, Z^{(k)})$ is a chosen 3-tuple difference, $1 \leq k \leq r$. $(A^{(k)}, B^{(k)})$ are the chosen fault injections, and $Z^{(k)} = x_k^{T^*} \oplus x_k^T$. We claim that 4 specific differences are enough to solve X, Y and Z despite of the word length n. Before the proof of the claim, we first introduce a lemma.

Lemma 1: Assume that the least i bits of X, Y and Z are known, for a given i with $0 \leq i \leq n-2$. Consider the i-th bit of X, Y and Z, i.e., x_i, y_i and z_i, if we choose

$$\begin{aligned} A_i^{(1)} A_{i-1}^{(1)} &= 10,\ B_i^{(1)} B_{i-1}^{(1)} = 00 \\ A_i^{(2)} A_{i-1}^{(2)} &= 00,\ B_i^{(2)} B_{i-1}^{(2)} = 10 \\ A_i^{(3)} A_{i-1}^{(3)} &= 01,\ B_i^{(3)} B_{i-1}^{(3)} = 00 \\ A_i^{(4)} A_{i-1}^{(4)} &= 00,\ B_i^{(4)} B_{i-1}^{(4)} = 01 \end{aligned}$$

and a fixed $Z_i^{(k)} Z_{i-1}^{(k)}$ for $k = 1, 2, 3, 4$, then x_i, y_i and z_i are determined uniquely.

The proof of Lemma 1 will be given in Appendix B.

Theorem 2: For any n bit X, Y and $Z = X \boxplus Y$, if we choose

$$A_i^{(1)} = \begin{cases} 1, & i = 0 \\ 1 + A_{i-1}^{(1)}, & 1 \le i < n \end{cases}, \quad B_i^{(1)} = 0, 0 \le i < n$$

$$A_i^{(2)} = 0, 0 \le i < n, \quad B^{(2)} = \begin{cases} 0, & i = 0 \\ 1 + B_{i-1}^{(2)}, & 1 \le i < n \end{cases}$$

$$A^{(3)} = B^{(1)}, \quad B^{(3)} = A^{(1)}, \quad A^{(4)} = B^{(2)}, \quad B^{(4)} = A^{(2)}$$

then the equation system (6) has four solutions, and these four solutions are different only in the most significant bit.

Proof: The theorem holds by induction on n and Lemma 1.

In fact, we randomly choose X, Y, and compute Z. The four parameters $(A^{(i)}, B^{(i)})$ are the same as in Theorem 2, and then the corresponding $Z^{(i)}$ is computed. The possible solution of (6) is shown in below for $n = 32$.

- $n = 32$:
  Randomly choose X=0x69db6af3, Y=0xc717d7d6, Z=0x30f342c9, and then the four parameters are

  $(A^{(1)}, B^{(1)}, Z^{(1)})$ = (0x55555555, 0x0, 0x335555b5),
  $(A^{(2)}, B^{(2)}, Z^{(2)})$ = (0x0, 0x55555555, 0xcceeafbf),
  $(A^{(3)}, B^{(3)}, Z^{(3)})$ = (0xaaaaaaaa, 0x0, 0xba7adae6),
  $(A^{(4)}, B^{(4)}, Z^{(4)})$ = (0x0, 0xaaaaaaaa, 0xe76baaa6).

  Then all possible solutions are

  X=0xe9db6af3, Y=0xc717d7d6, Z=0xb0f342c9,
  X=0xe9db6af3, Y=0x4717d7d6, Z=0x30f342c9,
  X=0x69db6af3, Y=0xc717d7d6, Z=0x30f342c9,
  X=0x69db6af3, Y=0x4717d7d6, Z=0xb0f342c9.

It is easy to see that the non-uniqueness lies in the uncertain of the most significant bits. Thus k^{T-1} is recovered by (4) except the most significant bit.

V. Conclusion

In this paper we analyze the structure of round functions in the SPECK family, and present two improved differential fault attacks under different fault models. The first attack is under a practical, random word fault model. After the certain number of fault injection, an algebraic system of addition modulo 2^n is established. Then we apply Gröbner bases algorithm to recover the last round key of SPECK family. Experiments show that the round key of the SPECK family is recovered by using about $5 \sim 8$ fault injections, which is far less than that in [14]. The second attack assumes a chosen-value fault model, where w e inject the specific faults, and can retrieve the last round key (except the most significant bit) with only 4 faults over all 10 versions of SPECK.

Acknowledgment

The research was supported by National Natural Science Foundation of China (No. 61121062, 11071285, 61170289), the 973 Program (2011CB302401, 2013CB834203) and the open project of the SKLOIS in Institute of Information Engineering, CAS(Grant No. 2015-MS-03).

References

[1] Dan Boneh, Richard A DeMillo and Richard J Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In Advances in Cryptology EUROCRYPT 97, volume 1233 of Lecture Notes in Computer Science, pages 37-51. Springer, 1997.

[2] Dan Boneh, Richard A DeMillo and Richard J Lipton. On the Importance of Eliminating Errors in Cryptographic Computations. Journal of cryptology 14(2), pages 101-119, 2001.

[3] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In Advances in Cryptology Crypto 97, volume 1294 of Lecture Notes in Computer Science, pages 513-525. Springer, 1997.

[4] Piret Gilles and Jean-Jacques Quisquater. A differential fault attack technique against SPN structure, with application to the AES and KHAZAD. In Cryptographic Hardware and Embedded Systems (CHES 2003), volume 2779 of Lecture Notes in Computer Science, pages 77-88, Springer, 2003.

[5] Dusart Pierre, Gilles Letourneux, and Olivier Vivolo. Differential fault analysis on AES. In Applied Cryptography and Network Security, pages 293-306, Springer, 2003.

[6] Hoch Jonathan J. and Adi Shamir. Fault Analysis of Stream Ciphers. Cryptographic Hardware and Embedded Systems-CHES 2004, volume 3156 of Lecture Notes in Computer Science, pages 240-253, Springer, 2004.

[7] Jovanovic Philipp, Martin Kreuzer and Ilia Polian. A Fault Attack on the LED Block Cipher. In W. Schindler and S. Huss (eds.) COSADE 2012, Volume 7275 of Lecture Notes in Computer Science, pages 120-134, Springer, 2012.

[8] Bar-El Hagai, Hamid Choukri, David Naccache, Michael Tunstall and Claire Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. Proceedings of the IEEE 94(2), pages 370-382, 2006.

[9] Bogdanov Andrey, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin and Charlotte Vikkelsoe. Present: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (ed.) CHES 2007. volume 4727 of Lecture Notes in Computer Science, pages 450-466, Springer, 2007.

[10] Borghoff Julia, Anne Canteaut, Tim Gneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar and Christian Rechberger. PRINCE: a Low-Latency Block Cipher for Pervasive Computing Applications. In Advances in Cryptology Asiacrypt 2012, Volume 7658 of Lecture Notes in Computer Science, pages 208-225, Springer, 2012.

[11] De Canniere Christophe, Orr Dunkelman and Miroslav Knezevic. Katan and Ktantan Family of Small and Efficient Hardware-Oriented Block Ciphers. In Cryptographic Hardware and Embedded Systems-Ches 2009, volume 5747 of Lecture Notes in Computer Science, pages 272-288, Springer, 2009.

[12] Guo Jian, Thomas Peyrin, Axel Poschmann and Matt Robshaw. The Led Block Cipher. In Cryptographic Hardware and Embedded Systems, Ches 2011, volume 6917 of Lecture Notes in Computer Science, pages 326-341, Springer, 2011.

[13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. IACR Cryptology ePrint Archive 2013/404, 2013, http://eprint.iacr.org/.

[14] Harshal Tupsamudre, Shikha Bisht and Debdeep Mukhopadhyay. Differential Fault Analysis on the Families of Simon and Speck Ciphers. In Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pages 40-48, IEEE, 2014.

[15] Nicolas T Courtois and Blandine Debraize. Algebraic Description and Simultaneous Linear Approximations of Addition in Snow 2.0. In Information and Communications Security, volume 5308 of Lecture Notes in Computer Science, pages 328-344, Springer, 2008.

[16] Blandine Debraize and Irene Marquez Corbella. Fault Analysis of the Stream Cipher Snow 3g. In Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC) , pages 103-110, IEEE, 2009.

[17] SINGULAR. A Computer Algebra System for polynomial computations with special emphasis on the needs of commutative algebra, algebraic geometry, and singularity theory. Available at http://www.singular.uni-kl.de.

[18] Bruno Buchberger and Franz Winkler. Gröbner Bases and Applications. London Mathematical Society LNS 251, Cambridge University Press, 1998.

APPENDIX

A. The specific process of one simulation

We made simulations to recover all the bits of X in the last round. Once X is known the last round key is uniquely determined. The following is an example.

Let $r = 6, n = 32$, choose randomly

$$X = 00000011001100000111111011110110,$$
$$Y = 11000111010010001110100010010011,$$

Then $Z = 11001010011110010110011110001001$. The r faults deduce Y_i and ΔZ_i, where $i = 1, \cdots, r$. The random Y_i and the corresponding ΔZ_i are

$$\begin{aligned}
Y_1 &= 11011001001111111111110110100110,\\
\Delta Z_1 &= 00010110000010010001101100010101\\
Y_2 &= 00001101001111100101100000000000,\\
\Delta Z_2 &= 11011010000101111011000101111111\\
Y_3 &= 01010110110010000000011111110000,\\
\Delta Z_3 &= 10010011100000011110000101101111\\
Y_4 &= 00010111010010001111110010011011,\\
\Delta Z_4 &= 11010000000000000001110000011000\\
Y_5 &= 11011111000111010000011011110011,\\
\Delta Z_5 &= 00101000001101001110001001100000\\
Y_6 &= 01101010100011111010001011011000,\\
\Delta Z_6 &= 10100111101110010100011001000111
\end{aligned}$$

Then we use the function "slimgb" and "elim" to compute the Gröbner bases with respect to X. G' consists of

$$\begin{gathered}
x_1,\ x_2+1,\ x_3+1,\ x_4,\ x_5+1,\ x_6+1,\ x_7+1,\ x_9,\ x_{10}+1,\\
x_{11}+1,\ x_{12}+1,\ x_{13}+1,\ x_{14}+1,\ x_{15}+1,\ x_{16},\ x_{17},\ x_{18},\\
x_{19},\ x_{20},\ x_{21}+1,\ x_{22}+1,\ x_{24}+x_{23},\ x_{26}+1, x_{27},\ x_{28},\\
x_{29},\ x_{30},\ x_{31},\ x_{25}+x_{23}^2+1,\ x_{25}^2+x_{25}.
\end{gathered}$$

Therefore, the directly determined bits are:

$$\begin{gathered}
x_1 = 0,\ x_2 = 1,\ x_3 = 1,\ x_4 = 0,\ x_5 = 1,\ x_6 = 1,\ x_7 = 1,\\
x_9 = 0,\ x_{10} = 1,\ x_{11} = 1,\ x_{12} = 1,\ x_{13} = 1,\ x_{14} = 1,\\
x_{15} = 1,\ x_{16} = 0,\ x_{17} = 0,\ x_{18} = 0,\ x_{19} = 0,\ x_{20} = 0,\\
x_{21} = 1,\ x_{22} = 1,\ x_{26} = 1,\ x_{27} = 0,\ x_{28} = 0,\ x_{29} = 0,\\
x_{30} = 0,\ x_{31} = 0.
\end{gathered}$$

The remain uncertain bits are $x_8, x_{23}, x_{24}, x_{25}$ and x_{32}. It is noticed that $x_{24} + x_{23}, x_{25} + x_{23}^2 + 1$ and $x_{25}^2 + x_{25}$ can decrease the uncertainty of these 5 bits, but the most significant bit x_{32} can not be determined.

B. The Proof of Lemma 1

Proof: Define

$$b_j^{(k)} = x_j + A_j^{(k)} + y_j + B_j^{(k)} + z_j + Z_j^{(k)}$$

where $j = i-1, i, i+1$, $1 \le k \le 4$.
For the specific $A_i^{(k)} A_{i-1}^{(k)}$, we have

$$b_{i+1}^{(0)} = x_i y_i + (x_i + y_i) b_i^{(0)} \tag{7}$$
$$b_{i+1}^{(1)} = (x_i + 1) y_i + (x_i + 1 + y_i) b_i^{(1)} \tag{8}$$
$$b_{i+1}^{(2)} = x_i (y_i + 1) + (x_i + y_i + 1) b_i^{(2)} \tag{9}$$
$$b_{i+1}^{(3)} = x_i y_i + (x_i + y_i) b_i^{(3)} \tag{10}$$
$$b_{i+1}^{(4)} = x_i y_i + (x_i + y_i) b_i^{(4)} \tag{11}$$

and

$$b_i^{(0)} = x_{i-1}y_{i-1} + (x_{i-1} + y_{i-1})b_{i-1}^{(0)} \quad (12)$$

$$b_i^{(1)} = x_{i-1}y_{i-1} + (x_{i-1} + y_{i-1})b_{i-1}^{(1)} \quad (13)$$

$$b_i^{(2)} = x_{i-1}y_{i-1} + (x_{i-1} + y_{i-1})b_{i-1}^{(2)} \quad (14)$$

$$b_i^{(3)} = (x_{i-1} + 1)y_{i-1} + (x_{i-1} + 1 + y_{i-1})b_{i-1}^{(3)} \quad (15)$$

$$b_i^{(4)} = x_{i-1}(y_{i-1} + 1) + (x_{i-1} + y_{i-1} + 1)b_{i-1}^{(4)} \quad (16)$$

As the i least bits of X, Y, Z are known, and the i least bits of $A^{(k)}, B^{(k)}, Z^{(k)}$ are given, it implies that $b_{i-1}^{(k)}$ is unique for $k = 0, 1, 2, 3, 4$. Furthermore, $b_i^{(k)}$ is unique with above equations.

Note that

$$b_j^{(0)} + b_j^{(k)} = A_j^{(k)} + B_j^{(k)} + Z_j^{(k)},$$

where $j = i+1, i, i-1$ and $k = 1, 2, 3, 4$. We conclude that all $b_j^{(0)} + b_j^{(k)}$ are the fixed constants.

From (7), (8) and (9), we have

$$\begin{cases} b_{i+1}^{(0)} + b_{i+1}^{(1)} = y_i + b_i^{(1)} + (x_i + y_i)(b_i^{(0)} + b_i^{(1)}) \\ b_{i+1}^{(0)} + b_{i+1}^{(2)} = x_i + b_i^{(2)} + (x_i + y_i)(b_i^{(0)} + b_i^{(2)}) \end{cases} \quad (17)$$

Below we discuss the uniqueness of x_i, y_i on the condition of $b_i^{(1)} + b_i^{(2)}$.

1. $b_i^{(1)} = b_i^{(2)}$. That is $b_i^{(0)} + b_i^{(1)} = 0$. By (17) we have

   $$b_{i+1}^{(0)} + b_{i+1}^{(1)} = y_i + b_i^{(1)}$$
   $$b_{i+1}^{(0)} + b_{i+1}^{(2)} = x_i + b_i^{(2)}$$

   As $b_i^{(1)}, b_i^{(2)}$ and $b_{i+1}^{(0)} + b_{i+1}^{(1)}, b_{i+1}^{(0)} + b_{i+1}^{(2)}$ are unique, x_i, y_i are unique. The case when $b_i^{(0)} + b_i^{(1)} = 1$ is similar.
2. $b_i^{(1)} \neq b_i^{(2)}$. It is noticed that $b_0^{(1)} = b_0^{(2)} = 0$, thus the following analysis is based on $i > 0$. With (13) and (14), we have

   $$1 = b_i^{(1)} + b_i^{(2)} = (x_{i-1} + y_{i-1})(b_{i-1}^{(1)} + b_{i-1}^{(2)}) \quad (18)$$

   It implies that $x_{i-1} + y_{i-1} = 1$. With (7), (10) and (11), we have

   $$b_{i+1}^{(0)} + b_{i+1}^{(3)} = (x_i + y_i)(b_i^{(0)} + b_i^{(3)}) \quad (19)$$
   $$b_{i+1}^{(0)} + b_{i+1}^{(4)} = (x_i + y_i)(b_i^{(0)} + b_i^{(4)}) \quad (20)$$

   If $b_i^{(0)} + b_i^{(3)}$ or $b_i^{(0)} + b_i^{(4)}$ is equal to 1, then we obtain the equation $x_i + y_i = z$, where

   $$z = \begin{cases} b_{i+1}^{(0)} + b_{i+1}^{(4)}, \text{ if } b_i^{(0)} + b_i^{(3)} = 1 \\ b_{i+1}^{(0)} + b_{i+1}^{(5)}, \text{ if } b_i^{(0)} + b_i^{(4)} = 1 \end{cases}$$

   is a fixed constant. Furthermore, x_i and y_i are uniquely determined.

   Otherwise we have

   $$b_i^{(0)} + b_i^{(3)} = b_i^{(0)} + b_i^{(4)} = 0$$

   With (12), (15) and (16), we have

   $$\begin{aligned} 0 = b_i^{(0)} + b_i^{(3)} &= y_{i-1} + b_{i-1}^{(3)} + (x_{i-1} + y_{i-1})(b_{i-1}^{(0)} + b_{i-1}^{(3)}) \\ &= y_{i-1} + b_{i-1}^{(0)} \\ 0 = b_i^{(0)} + b_i^{(4)} &= x_{i-1} + b_{i-1}^{(4)} + (x_{i-1} + y_{i-1})(b_{i-1}^{(0)} + b_{i-1}^{(4)}) \\ &= x_{i-1} + b_{i-1}^{(0)} \end{aligned}$$

   It implies that $x_{i-1} + y_{i-1} = 0$, which is a contradiction.