Side Channel Cryptanalysis on SEED*

HyungSo Yoo[1], ChangKyun Kim[1], JaeCheol Ha[2,**],
SangJae Moon[1], and IlHwan Park[3]

[1] School of Electrical Engineering and Computer Science,
Kyungpook National Univ., Daegu, 702-701, Korea
{hsyoo,dreamps}@m80.knu.ac.kr, sjmoon@knu.ac.kr
[2] Division of Information Science, Korea Nazarene Univ.,
Cheonan, Choongnam, 330-718, Korea
jcha@kornu.ac.kr
[3] National Security Research Institute, Daejeon, Korea
ilhpark@etri.re.kr

Abstract. The Korea standard block cipher, SEED, is a 128-bit symmetric block cipher with a more complex F function than DES. This paper shows that SEED is vulnerable to two types of side channel attacks, a fault analysis attack and a power analysis attack. The first one is a fault insertion analysis which induces permanent faults on the whole left register of 15-round. This attack allows one to obtain the secret key by using only two faulty cipher texts for encryption and decryption processing respectively. The second attack is a more realistic differential power analysis. This attack requires about 1000 power traces to find the full secret key. The above two attacks use a reverse property of the F function to obtain secret key, where the reverse property is derived from the our research.

Keywords: Side channel attack, Fault insertion analysis, Differential power analysis, block cipher, SEED.

1 Introduction

In September 1996, Boneh *et al.* announced a new cryptanalytic attack which could affect security of cryptographic modules [6]. They succeeded in breaking the RSA with CRT by using one correct signature and one faulty one. In this attack, hardware faults and errors which occur during the operations of a cryptographic device might leak information about the private key. Lenstra *et al.* improved this attack by finding two secret prime number using only one faulty signature of a message [15, 19]. Consequently, many papers have been published concerning the resistance of RSA cryptosystems with CRT to fault attacks [2,

* This research has been supported by University IT Research Center Project.
** The third author was also supported in part by Korea Nazarene University research fund.

C.H. Lim and M. Yung (Eds.): WISA 2004, LNCS 3325, pp. 411–424, 2004.

26, 27]. Also the fault attack by optical illumination [23] or by spike generator [2] has been reported and is much more feasible and potential for breaking cryptosystems.

In addition, Biham and Shamir have published a paper [4] detailing a fault analysis attack which is applicable to secret key cryptosystems such as Data Encryption Standard (DES). Assuming the same register faults that Boneh *et al.* considered [6], they showed that DES could be broken. They combined two attack techniques differential cryptanalysis [3] and fault analysis. Dusart *et al.* showed how DFA (Differential Fault Analysis) works on Advanced Encryption Standard (AES) [9]. They implemented this attack on a PC, then found the full AES-128 key by analyzing less than 50 cipher texts. In 2002, Giraud also presented a paper describing two types of DFA attacks on AES [11] using between 50 and 250 faulty cipher texts.

On the other hand, Kocher *et al.* [17] firstly introduced power attacks including the simple and differential power analyses (referred to as SPA and DPA, respectively). In SPA, a single power consumption of a cryptographic execution is measured and a trace is analyzed to classify operations which are related to secret information. In DPA, an adversary measures hundreds of power signal traces, divides them into two groups using a classification criterion, and makes a differential computation between the two averaged values. Since the averaging and subtraction of two signal groups results in the amplification of small power differences which occur during the execution of an algorithm. In general, DPA is more powerful than SPA [7, 12, 25]. In fact, it has been reported that secret key cryptosystems (AES and DES) as well as public key cryptosystems (RSA and ECC) are vulnerable to DPA [5, 17, 20, 21].

In this paper, we show that the SEED which is a national industrial association standard algorithm in Korea (TTAS.KO-12.0004, 1999) and an international standard candidate for ISO/IEC SC27 CD 18033-3 [14, 18] is vulnerable to both fault attack and power attack. This paper is mostly divided in two parts. In the first part, the SEED is vulnerable to a fault insertion analysis which induces permanent faults on a whole left register of 15-round. This attack allows us to obtain the secret key by using only two faulty cipher texts for encryption and decryption processing respectively. The first reason having vulnerable property is that F function of SEED is recoverable, that is, if input and output are known, then adversary can find round key. The second reason is that if 1-round and 16-round key are known, then he can completely find full secret key. In the second part, the SEED is also vulnerable to a DPA. By using this attack, we can get a output value of a F function of 1-round during a decryption processing. The rest of this attack is similar to fault attack. This paper shows our experimental results of DPA on SEED.

This paper is organized as follows. In section 2, we give a brief overview of SEED. In section 3, we present our fault analysis, including our assumptions and the theory behind the attack. Section 4 shows the DPA method on SEED and our experimental results. Finally in section 5 we briefly discuss the countermeasure against the two types of attack on SEED and make a conclusion.

2 SEED Algorithm

The Korea standard block cipher, SEED is a secret key block cipher with a 128-bit data block and a 128-bit secret key. This algorithm has a Feistel structure with 16 rounds and a 128-bit input-output data block. The following notations are used throughout this paper.

$\boxplus$: addition in modular 2^{32}
$\boxminus$: subtraction in modular 2^{32}
$\oplus$: bitwise exclusive OR(XOR)
& : bitwise AND
$<< n$: left circular rotation by n bits
$>> n$: right circular rotation by n bits
$\|$: concatenation

2.1 Structure of SEED

A input plain text of a 128-bits is divided into two 64-bit blocks. The right 64-bit block R_0 is an input to the F function with a first 64-bit round subkey which is generated from the round key generation processing. The output of F function is XORed with the left 64-bit block L_0. After 16 round encryption processings, the final 128-bit output is a cipher text. The overview of SEED structure is shown in Figure 1.

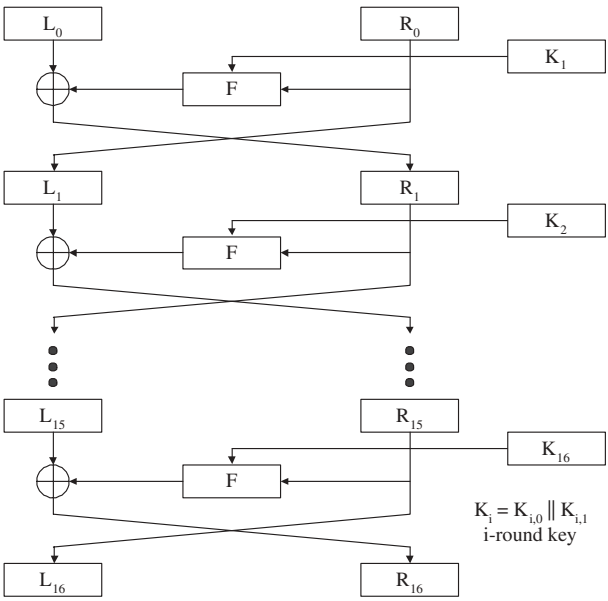


Fig. 1. Structure of SEED.

The F function also has a 64-bit Feistel structure. The input block is divided into two blocks (C, D) and XORed with two 32-bit subkeys ($K_{i,0}, K_{i,1}$). After mixing subkeys, two blocks are passed through three layers of G function with addition in modular 2^{32}. The structure of F function is shown in Figure 2.

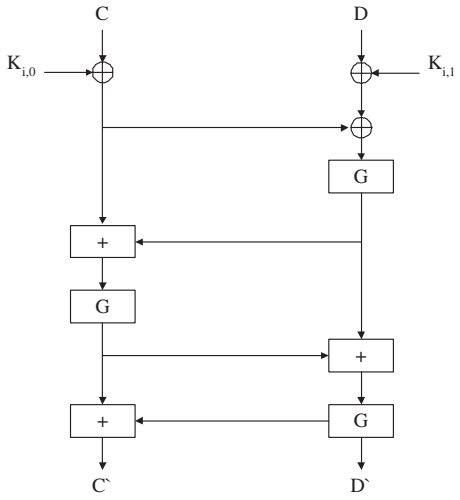


Fig. 2. The F function.

As shown in Figure 3, the G function used in F function has two security layers. The first layer consists of two S-boxes generated from boolean functions X^{247} and X^{251}. Here, S boxes can be represented by two lookup tables. The 32-bit input of G function is divided into 4 blocks. Each 8-bit block pass through the 8×8 S-boxes, S_2 and S_1. The second layer consists of permutation processing of S-box outputs which is a computation by AND operation with four specific values from m_0 to m_3. After XOR processing of expanded 16 blocks, G function generates a final 32-bit output.

$$
\begin{aligned}
&Y_3 = S_2(X_3), Y_2 = S_1(X_2), Y_1 = S_2(X_1), Y_0 = S_1(X_0),\\
&Z_3 = (Y_0 \& m_3) \oplus (Y_1 \& m_0) \oplus (Y_2 \& m_1) \oplus (Y_3 \& m_2),\\
&Z_2 = (Y_0 \& m_2) \oplus (Y_1 \& m_3) \oplus (Y_2 \& m_0) \oplus (Y_3 \& m_1),\\
&Z_1 = (Y_0 \& m_1) \oplus (Y_1 \& m_2) \oplus (Y_2 \& m_3) \oplus (Y_3 \& m_0),\\
&Z_0 = (Y_0 \& m_0) \oplus (Y_1 \& m_1) \oplus (Y_2 \& m_2) \oplus (Y_3 \& m_3).
\end{aligned}
$$

2.2 Round Key Generation

The round key generation function uses the G function, addition in modular 2^{32}, subtraction in modular 2^{32}, and left (right) circular rotation by 8 bits.

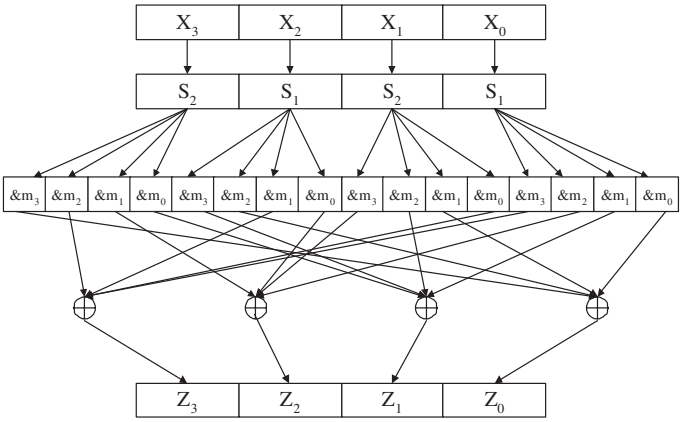


Fig. 3. The G function.

A 128-bit input key is separated into four 32-bit blocks (A, B, C, D). The two divided blocks perform addition (or subtraction) in modular 2^{32} and subtraction (addition) by the constant KC_1. The 1-round keys $K_{1,0}$ and $K_{1,1}$ are generated using final G function operations. After an 8-bit right rotation of the left 64-bits the second round keys $K_{2,0}$ and $K_{2,1}$ are generated by addition, subtraction by the constant KC_2, and G functions. The rest of the subkeys are generated in the same way, as shown in Figure 4.

3 Fault Analysis Attack on SEED

On a smart card, a fault may be induced in many ways, such as by a power glitch, by a clock pulse, or by radiation from a laser, etc. In this paper, the fault attack assumes that the permanent fault occurs on whole bit of a register. This attack is similar to one of two assumptions made by Biham and Shamir in fault attack for DES [4]. The other fault attack assumption is difficult to apply to our fault attack and is difficult to implement experimentally. One criticism against this second model, a differential fault analysis attack, is that the transient fault attack assumed by Biham and Shamir [4] is not realistic. Therefore, we assume a more practical fault model that will be less controversial. Our fault attack assumes that we can cut a wire or destroy a memory cell in a cryptoprocessor such as a smart card. As a result, the values in affected location can be considered to be permanently fixed. We assume that an adversary can insert these permanent faulty values into some memory cells. Some papers make similar assumptions in their work [1, 4, 22].

3.1 Fault Analysis Attack

We assume that SEED is implemented in hardware as 16 unrolled hardware rounds. For this attack, it suffices to destroy all the bits of the LSB of register L_{15}

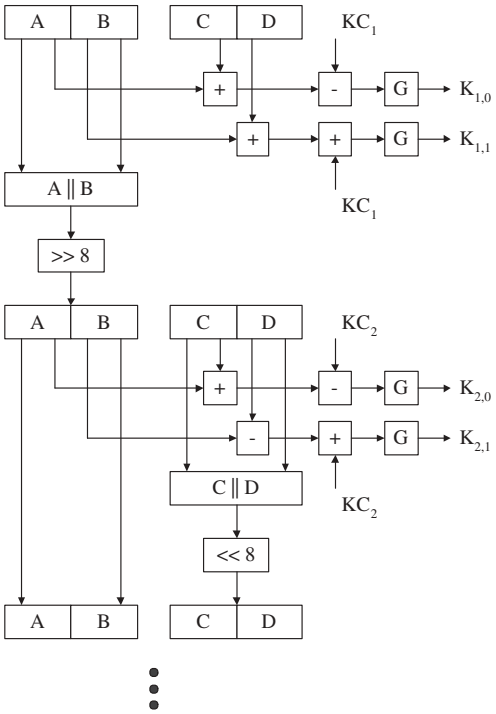


Fig. 4. Round key generation.

or set them to known values. This attack assumes that we can destroy a memory cell in a register or cut a wire. As a result, we consider the memory cells to be fixed to known values. This attack is a pure cipher text only attack, which does not require any information about plain texts. Based upon this assumption, we can find the 16-round keys, $K_{16,0}$ and $K_{16,1}$. The 16-round in the implementation of SEED is shown in Figure 5.

Our attack to find out the 16-round subkeys is composed of 4 steps as follows.

Step 1

We induce the faults to destroy all the bits of the LSB register L_{15} or set them to known values. In addition, we can see the final output of SEED, L_{16} and R_{16}. Therefore, we can find the input and outputs of the F function. In the end, we want to find the 16-round keys $K_{16,0}$ and $K_{16,1}$ when we know the input and output of the F function.

Step 2

Given that inputs and outputs in F function of Figure 2 (C, D, C', and D') are known, we want to find the keys $K_{16,0}$ and $K_{16,1}$. As for the addition in modular 2^{32}, the inverse operation is subtraction in the same modular. So, based on the assumption that the output of G function is known, if we can find the input of it, we safely say that we can extract the 16-round keys.

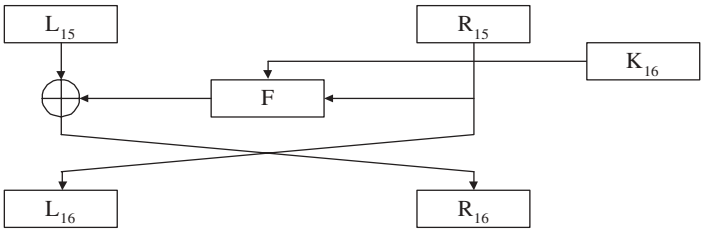


Fig. 5. 16-round of SEED.

Step 3

Let us attempt to find the inputs of G function given the outputs. In Figure 3, we denote the 32-bit inputs as X_0, X_1, X_2 and X_3, and the outputs as Z_0, Z_1, Z_2 and Z_3. It is enough to know the inverse values of function S by S-box tables S_1 and S_2. Consequently, the outputs of S-boxes are the result of $S_2(X_3)||S_1(X_2)||S_2(X_1)||S_1(X_0)$. An important point to emphasize is that the least significant bits Z_{00}, Z_{10} , Z_{20} and Z_{30} are determined by 4 bit inputs $S_{20}(X_3)||S_{10}(X_2)||S_{20}(X_1)||S_{10}(X_0)$. Here, Z_{ij} is a jth bit of Z_i. By selecting 8 out of 16 inputs combined by $S_{20}(X_3)$, $S_{10}(X_2)$, $S_{20}(X_1)$, and $S_{10}(X_0)$ we can calculate the correct output bit Z_{00}. Similarly, by selecting 4 out of 8 inputs to check bit Z_{00}, we can calculate the correct output bit Z_{10} and by selecting 2 out of 4 inputs to check Z_{10} bit, we can calculate the correct output bit Z_{20}. Finally, by selecting 1 out of 2 inputs to check bit Z_{20}, we can calculate the correct output bit Z_{30}. As a result, if we know the outputs of the G function, then we can compute its inputs, that is, the G function becomes reversible; there is a reverse computational method to find the inputs for given outputs. We will use the notation G^{-1} function to represent this inverse computational algorithm for G function.

Step 4

We know the inverse method for a G function and the addition function in modular 2^{32}. Therefore, the reverse computation for a F function shown in Figure 2 is possible. Finally, given the input and output of F function, we can find the 16-round keys, $K_{16,0}$ and $K_{16,1}$.

3.2 Secret Key Attack Using Two Round Keys

After fault insertion at 16-round for encryption of plain text, we can calculate 16-round keys by using the final cipher text. Additionally, after fault insertion at 16-round for decryption of the cipher text, we can compute 1-round keys using the final plain text. Note that it is not necessary to use a genuine cipher text as an input. It is sufficient to use random data as an input because an adversary is only interested in the decryption operation.

Now, let us examine how to search the 128-bit secret key given a 1-round and a 16-round keys. As you see in Figure 4, a 128-bit input key is divided into four 32-bit blocks, and then used to generate a 64-bit round key at each

round. The round key generation algorithm uses four computational operations: circular rotation by 8 bits, addition (subtraction) in modular 2^{32} and G function. As has been noted, we can compute inverse values of function G and addition (subtraction) in modular 2^{32}. Therefore, if we know the 1-round keys $K_{1,0}$ and $K_{1,1}$ in Figure 4, then we know two temporary values $(A + C)$ and $(B - D)$ as follows.

$$\begin{aligned} A + C &= G^{-1}(K_{1,0}) + KC_1 = T_{1,0} \\ B - D &= G^{-1}(K_{1,1}) - KC_1 = T_{1,1} \\ A_3||A_2||A_1||A_0 + C_3||C_2||C_1||C_0 &= T_{1,0} \\ B_3||B_2||B_1||B_0 - D_3||D_2||D_1||D_0 &= T_{1,1} \end{aligned}$$

Note that the A and B used in 16-round key generation are the same that were used in 1-round due to 8 right circular rotations by 8 bits. Furthermore, C and D used in the 16-round key generation are the same that operated by only one right circular rotations by 8 bits for 1-round C and D.

$$\begin{aligned} A + L((C||D) >> 8) = G^{-1}(K_{16,0}) + KC_{16} &= T_{16,0} \\ B - R((C||D) >> 8) = G^{-1}(K_{16,1}) + KC_{16} &= T_{16,1} \\ A_3||A_2||A_1||A_0 + D_0||C_3||C_2||C_1 &= T_{16,0} \\ B_3||B_2||B_1||B_0 - C_0||D_3||D_2||D_1 &= T_{16,1} \end{aligned}$$

Here, $L(\ )$ means the extraction of the left 32 bits from 64 bits data and $R(\ )$ means right extraction. In order to compute the secret key, we subtract two equations as follows.

$$\begin{aligned} T_{16,1} - T_{1,1} &= D_3||D_2||D_1||D_0 - C_0||D_3||D_2||D_1 \\ T_{1,0} - T_{16,0} &= C_3||C_2||C_1||C_0 - D_0||C_3||C_2||C_1 \end{aligned}$$

As you know, if C and D are known, then A and B can easily be computed. Figure 6 describes the operation to subtract two temporary round keys. As shown in Figure 6, knowing $T_{16,1} - T_{1,1}$ and $T_{1,0} - T_{16,0}$, we can compute 256 possible secret keys. Now, let D_0 be a random 8 bit value. Then we can compute D_1 as $D_0 - R_{7-0}(T_{16,1} - T_{1,1})$ where $R_{7-0}(K)$ denotes the bits from the LSB to the 7th bit of K. Consecutively, we can compute D_2 from $D_1 - R_{15-8}(T_{16,1} - T_{1,1})$ as follows.

$$\begin{aligned} D_1 &= D_0 - R_{7-0}(T_{16,1} - T_{1,1}) \\ D_2 &= D_1 - R_{15-8}(T_{16,1} - T_{1,1}) \\ D_3 &= D_2 - R_{23-16}(T_{16,1} - T_{1,1}) \\ C_0 &= D_3 - R_{31-24}(T_{16,1} - T_{1,1}) \\ C_1 &= C_0 - R_{7-0}(T_{1,0} - T_{16,0}) \\ C_2 &= C_1 - R_{15-8}(T_{1,0} - T_{16,0}) \\ C_3 &= C_2 - R_{23-16}(T_{1,0} - T_{16,0}) \\ D_0 &= C_3 - R_{31-24}(T_{1,0} - T_{16,0}) \end{aligned}$$

Note that D_0 used to compute $D_1 = (D_0 - R_{7-0}(T_{16,1} - T_{1,1}))$ is same with the final value $C_3 - R_{31-24}(T_{1,0} - T_{16,0})$. Finally, we can compute 256 C and D from 256 D_0. If 256 C and D are known, then 256 A and B can simply be computed. These 256 secret keys can always generate secret keys which satisfy the relationship between 1-round and 16-round keys. The final step to find a complete secret key is to do an exhaustive search of the 256 possible secret keys. Known plain text/cipher text pair (using the fault-inserted cipher text), a computer can find the unique secret key satisfied the text-pair from the 256 possible secret keys by an exhaustive software search. Thus, we can find the full secret key using only two faulty cipher texts, where one is the output for encryption processing, and the other is for decryption processing.

We assume that SEED is implemented as a single round, which is used 16 times. This is an iterated hardware implementation of SEED. In this case we generate a permanent fault in the left-half register, in which all of the bits are permanently fixed. At this point, it doesn't matter whether the value of the left-half register is zero or known values. In iterated implementations, we can also apply the proposed attack which can compute the 1-round and 16-round keys.

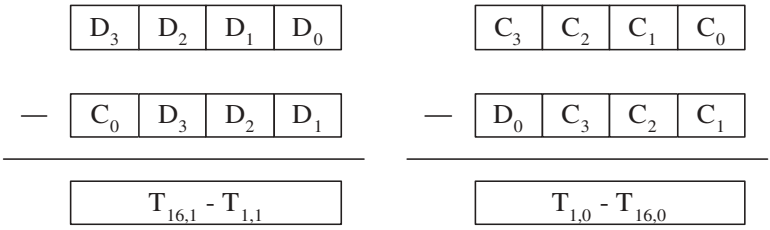


Fig. 6. Differential value between two temporary round keys.

4 Power Analysis Attack on SEED

Proposition 1. *If both the input and output value of the F function in i-round are known, i-round key can be computed using a reverse algorithm of the F function.*

DPA is a powerful attack in which an adversary collects a number of power traces from a hardware device as it repeatedly executes a cryptographic operation. In our DPA attack, an adversary must have knowledge of inputs processed by the device. Furthermore the same secret key is used in encryption (decryption) over multiple plain texts (cipher texts).

Our basic implementation of DPA is as follows. Assume that an adversary is able to input two 64-bit plain texts R_0 and L_0, and measures power consumption traces. Now, an adversary would like to attack the 1-round keys $K_{1,0}$ and $K_{1,1}$ as shown in Figure 7. Since the input and output of F function are known, according to Proposition 1, he can extract the 1-round key.

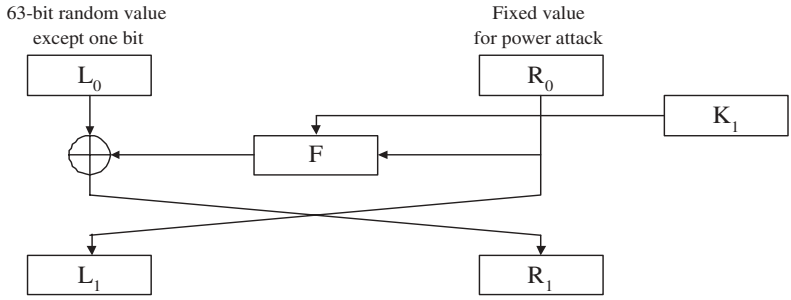


Fig. 7. The DPA attack in 1-round of SEED.

For example, we will find the least significant bit (LSB) of the output of F function at the 1-round. First, the value of the right half input R_0 is fixed during attack processing. Then, the left half input L_0 is randomly selected. We use a left input value which sets the LSB of L_0 to 0 or 1 and then measure the power signal at register R_1 during encryption processing. Let T_{it} be a sampled type of the power consumed. The i index corresponds to the ith power signal and t index corresponds to the time of the sample. Given the random L_0, the T_{it} are split into two sets according to the LSB of L_0 as follows.

$$T_0 = \{T_{it} \mid \text{LSB of } L_0 = 0\}$$
$$T_1 = \{T_{it} \mid \text{LSB of } L_0 = 1\}$$

Let T_0 be the set of measured traces where LSB is 0 and T_1, the set where LSB is 1. Both T_0 and T_1 will contain the same number of traces. Then we compute the average of the partitioning traces as follows:

$$A_0[t] = \frac{1}{|T_0|} \sum_{T_{it} \in T_0} T_{it}$$
$$A_1[t] = \frac{1}{|T_1|} \sum_{T_{it} \in T_1} T_{it}$$

Here, the number of measurements in a trace, $N = |T_0| = |T_1|$, depends on sampling rate and memory capacity. The differential trace of $A_0(t)$ and $A_1(t)$ is defined for $t = 1, ..., m$ as:

$$\triangle[t] = A_1[t] - A_0[t]$$

$$\lim_{N \to \infty} \triangle[t] = A_1[t] - A_0[t] = \begin{cases} 0 & \text{if } t \neq t^* \\ \varepsilon & \text{if } t = t^* \end{cases}$$

Assume that the register R_1 is stored at time t^* and t is equal to t^*. If the expected difference of power traces has a positive peak, $\varepsilon > 0$, then the LSB of F function of the 1-round is 0 because the storing power consumption for bit "1" is more than for bit "0". Similarly, if the difference between power traces has a

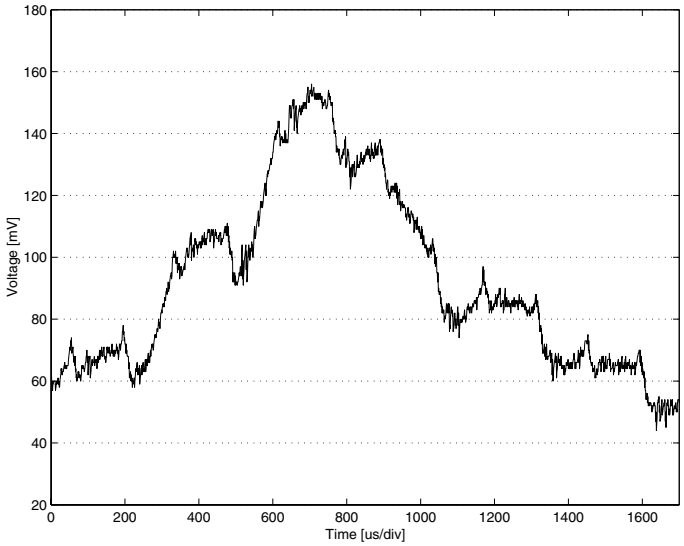


Fig. 8. Single power trace of an XOR operation which is $L_0 \oplus$ 64-bit of F function.

negative peak $\varepsilon < 0$, then the LSB of F function of the 1-round is 1. So, we can find the LSB bit of the output of F function at the 1-round. In addition, when t is not equal to t^*, the power dissipation is independent of the LSB because the smart card is manipulating bits other than the LSB.

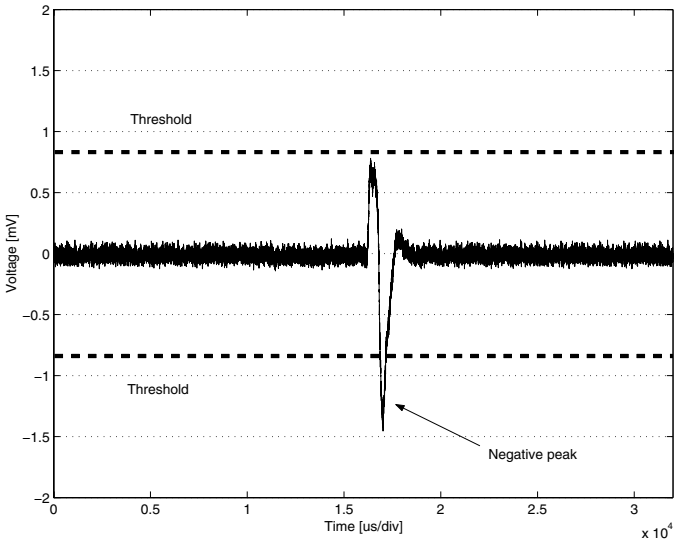
We show that SEED is vulnerable to our DPA by an experimental result. Our experiment was made on the 1-round implementation of SEED. Figure 8 shows a single power trace of an XOR operation to find the output of F function. Figure 9 illustrates the average difference between $A_0(t)$ and $A_1(t)$, in which we know whether the LSB of F function of the 1-round is 0 or 1. The signals in Figure 9 were obtained by averaging 5000 random power traces to observe a clear view of peak, but we have also been able to mount this attack with only 1000 power traces.

In our example, we have only discussed finding the LSB of F function at the 1-round. However, by a similar partition method to other bit using above measured traces, an adversary can steal all the bits of F function of the 1-round. As an above result and Proposition 1, we can extract the 1-round key.

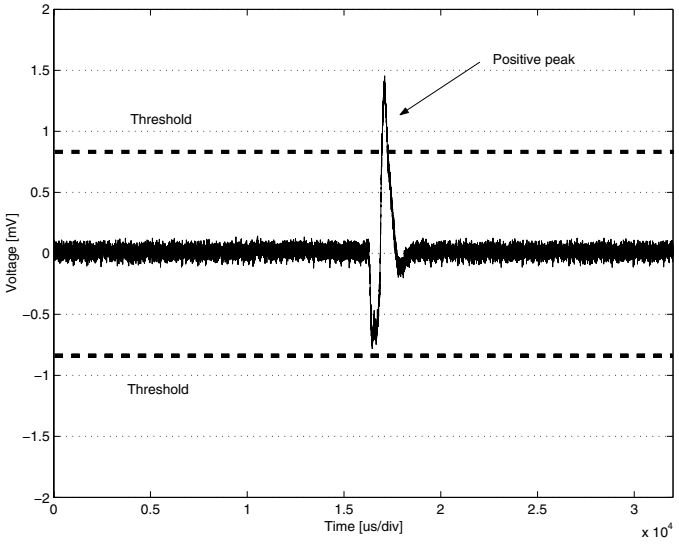
As mentioned above, if the 1-round key is vulnerable to DPA during the encryption processing, then the 16-round key can also be revealed during decryption. Therefore, an adversary can be compute a complete secret key from the 1-round and 16-round keys. The rest of this attack is similar to the fault attack described in section 3.2.

5 A Remark on Countermeasures and Conclusion

In this paper, we have shown that SEED is vulnerable to both a fault attack and a power attack. The basic assumption of the two attacks is that we can induce the



(a)



(b)

Fig. 9. Differential power traces. In the case (a), a negative peak is observed since the LSB bit of F function is 1, while in the case (b), a positive peak is observed since the LSB bit of F function is 0. We can make a decision about the threshold value by many experiments.

output values of F function through the side channel attack techniques. Since F function of SEED is recoverable if input and output are known, this assumption is quite reasonable and realistic. To achieve the real attack, we demonstrated

by power analysis experiment described in section 4. Our attacks is applicable to other block ciphers with Feistel structure. Furthermore, if target cipher has a recoverable properties for F function under our assumption, the round key is easily extracted by an adversary. Unfortunately, the SEED has a weakness to leak the full secret key by analyzing with only two round keys.

Countermeasures for resisting the fault insertion attacks can be implemented using both hardware [16] and software methods to detect any intrusions by external voltage variations, external clock variations, and physical fault induction attack. To defeat the power analysis, a power signal reduction technique, a self-timed dual-rail method, a sense amplifier based logic, a non-deterministic processor, etc., are needed [8, 10, 13, 24]. Implementers need to consider these side channel attacks when designing secure smart card systems. It is also important to prove the validity of some countermeasures not given in this paper.

Acknowledgement

The authors wish to thank the anonymous referees for their helpful remarks and suggestions.

References

1. R. Anderson and M. Kuhn, "Tamper resistance – a cautionary note," In *Proc. of the Second USENIX Workshop on Electronic Commerce*, pp. 1-11, November 1996. Available from http://www.usenix.org
2. C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J. Seifrert, "Fault attacks on RSA with CRT: Concrete results and practical countermeasures," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 260–275, Springer-Verlag, 2002.
3. E. Biham and A. Shmir, "Differential cryptanalysis of the full 16-round DES," In *Advances in Cryptology – CRYPTO '92*, LNCS 740, pp. 487–496, Springer-Verlag, 1992.
4. E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," In *Advances in Cryptology – CRYPTO '97*, LNCS 1294, pp. 513–525, Springer-Verlag, 1997.
5. B. Boer, K. Lemke, and G. Wieke, "A DPA attack against the modular reduction within a CRT implementation of RSA," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 228–243, Springer-Verlag, 2002.
6. D. Boneh, R.A. DeMillo, and R.J. Liption, "On the importance of checking cryptographic protocols for faults," In *Advances in Cryptology – EUROCRYPT '97*, LNCS 1233, pp. 37–51, Springer-Verlag, 1997.
7. J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," In *Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 292–302, Springer-Verlag, 1999.
8. J.F. Dhem and N. Feyt, "Hardware and software symbiosis helps smartcard evolution," In *IEEE Micro 21*, pp. 14-25, 2001.
9. P. Dusart, G. Letourneux, and O. Vivolo, "Differential fault analysis on A.E.S," In *Applied Cryptography and Network Security – ACNS '03*, LNCS 2846, pp. 293–306, Springer-Verlag, 2003.

10. N. Feyt, "Countermeasure method for a microcontroller based on a pipeline architecture," United States Patent 20030115478, June 19, 2003.
11. C. Giraud, "DFA on AES," In *IACR, Cryptology ePrint Archive,* Available from http://eprint.iacr.org/2003/008/, 2003
12. J.C. Ha and S.J. Moon, "Randomized signed-scalar multiplication of ECC to resist power attacks," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 551–563, Springer-Verlag, 2002.
13. J.I. den Hartog, J. Verschuren, E.P. de Vink, J. Vos, and W. Wiersma, "PINPAS: a tool for power analysis of smartcards," In *Information Security Conference – SEC '03*, pp. 453–457, Kluwer Academic, 2003.
14. ISO/IEC JTC 1/SC27, *"Third Party Evaluation on SEED by CRYPTEC,"* ISO/IEC JTC 1/SC27 N3213, April 23, 2002.
15. M. Joye, A.K. Lenstra, and J.-J. Quisquater, "Chinese remaindering based cryptosystems in the presence of faults," In *Journal of Cryptology*, vol. 12, no. 4, pp. 241–245, 1999.
16. R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent error detection of fault-based side-channal cryptanalysis of 128-bit symmetric block ciphers," Proc. of *IEEE DEsign Automation Conference*, pp. 579–585, 2001.
17. P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," In *Advances in Cryptology – CRYPTO '99*, LNCS 1666, pp. 388–397, Springer-Verlag, 1999.
18. Korea Information Security Agency, *Block Cipher Algorithm SEED*, Available from http://www.kisa.or.kr/seed/seed_eng.html.
19. A.K. Lenstra, "Memo on RSA signature generation in the presence of faults," September 1996.
20. T. Messerges, E. Dabbish, and R. Sloan, "Power analysis attacks of modular exponentiation in smartcards," In *Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 144–157, Springer-Verlag, 1999.
21. T. Messerges, "Securing the AES finalists against power analysis attacks," In *Fast Software Encryption – FSE '00*, LNCS 1978, pp. 150–164, Springer-Verlag, 2000.
22. J.A. Muir *Techniques of Side Channel Cryptanalysis*, masters thesis, 2001. Available from http://www.math.uwaterloo.ca/~jamuir/sidechannel.htm.
23. S.P. Skorobogatov and R.J. Anderson, "Optical fault induction attacks," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 2–12, Springer-Verlag, 2002.
24. K. Tiri, M. Akmal, I. Verbauwhede, "A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards," In *28th European Solid-State Circuits Conference – ESSCIRC '02*, September, 2002.
25. C.D. Walter, "Some security aspacts of the MIST randomized exponentiation algorithm," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 564–578, Springer-Verlag, 2002.
26. S.M. Yen, S.J. Moon, and J.C. Ha, "Hardware fault attack on RSA with CRT revisited," In *Information Security and Cryptology – ICISC '02*, LNCS 2587, pp. 374–388, Springer-Verlag, 2002.
27. S.M Yen, S.J. Moon, and J.C. Ha, "Permanent fault attack on the parameters of RSA with CRT," In *Information Security and Privacy – ACISP '03*, LNCS 2727, pp. 285–296, Springer-Verlag, 2003.