Update on SEED: SEED-192/256*

Kitae Jeong¹, Joongeun Choi¹, Yuseop Lee¹, Changhoon Lee², Jaechul Sung³, Haeryong Park⁴ and Yeonjung Kang⁴

- Center for Information Security Technologies(CIST), Korea University, Korea {kite, joongeun, yusubi}@cist.korea.ac.kr
 - ² School of Computer Engineering, Hanshin University, Korea chlee@hs.ac.kr
 - ³ Department of Mathematics, University of Seoul, Korea jcsung@uos.ac.kr
 - ⁴ Korea Information Security Agency(KISA), Korea {hrpark,yjkang}@kisa.or.kr

Abstract. SEED is a 128-bit block cipher with a 128-bit secret key. Since it supports only a 128-bit secret key, it is difficult to apply this algorithm to various environments. In this paper, we propose SEED-192/256 which support 192/256-bit secret keys, respectively. Also we evaluate the security these algorithms against well-known attacks and the software performance of them on PC environments.

Keywords: Block Ciphers, SEED.

1 Introduction

Block ciphers are usually designed to have fixed block sizes, key sizes and round numbers. However, the computing landscape is being considerably extended to various environments, nowadays. Thus block ciphers like AES [6], Camellia [1] and ARIA [5] which use fixed block sizes but flexible key sizes had been proposed. Since these algorithms support 128/192/256-bit secret keys, they are implementable on a wide variety of platforms and applications.

SEED is a 128-bit block cipher with a 128-bit secret key and 16 iterative rounds of Feistel structure [3]. It has been adopted as a national industrial association standard (TTAS KO-12.0004) at 1999 and ISO/IEC 18033-3 and IETF RFC 4269 at 2005. This algorithm has been adopted to most of security systems in Korea. It is designed to utilize S-boxes and permutations that balance with the current computing technology. It has the Feistel structure with 16 rounds, and is strong against differential cryptanalysis and linear cryptanalysis balanced with security/efficiency trade-offs. However, differently from AES, Camellia and ARIA, SEED supports only a 128-bit secret key. Thus it does not satisfy the flexibility on various platforms. So it is meaningful to design SEED which supports 192/256-bit secret keys.

^{*} This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. 2009-0060420).

2 Kitae Jeong et al.

Table 1. Comparison between key schedules of SEED-128 and SEED-192/256

	SEED-128	SEED-192(SEED-256)
Number of registers	4	6(8)
Operations	32-bit modular	32-bit modular
to generate	addition/subtraction,	addition/subtraction,
round keys	G function	G function, 32-bit XOR
Rotation unit	64 bits	96(128) bits

In this paper, we update SEED block ciphers which support 192/256-bit secret keys. We call these algorithms SEED-192 and SEED-256, respectively. SEED with a 128-bit secret key is denoted by SEED-128 in this paper. Similarly to other block ciphers which use secret keys of various lengths, SEED-192/256 are the same as SEED-128 except the number of rounds (SEED-192/256 are 20/24-round block cipher) and the key schedule. The key schedules of SEED-192/256 are designed by extending that of SEED-128. Compared to the key schedule of SEED-128, those of SEED-192/256 use only additional XOR operations. In [2], it had been shown that the key schedule of SEED-128 has a particular property which did not degenerate the security so far. So the additional operations improve the security without significantly degenerating the efficiency. We remove this property by adding XOR operations in SEED-192/256. Table 1 presents the comparison between key schedules of SEED-128 and SEED-192/256.

2 Description of SEED-128

In this section, we present SEED-128 briefly. Throughout this paper, the following notations are used.

- &: bitwise AND
- $\boxplus (\boxminus)$: addition(subtraction) in modular 2^{32}
- $\ll (\gg)n$: left(right) circular rotation by n bits
- ||: concatenation

The structure of SEED-128 is shown in Figure 1-a). A 128-bit plaintext is divided into two 64-bit sub-blocks (L^0 , R^0) and the right 64-bit block is an input to the round function F (See Figure 1-b)) with a 64-bit round keys generated from the key schedule.

As shown in Figure 2, G function has two layers: a layer of two 8×8 S-boxes, S_1 and S_2 , and a layer of block permutation of sixteen 8-bit sub-blocks. For detailed descriptions of S-boxes, see [3]. In the second layer, $m_0 = 0 \text{xfc}$, $m_1 = 0 \text{xf3}$, $m_2 = 0 \text{xcf}$ and $m_3 = 0 \text{x3f}$.

Figure 3-a) presents the key schedule of SEED-128. It uses G function, addition, subtraction and left/right circular rotation. A 128-bit secret key is divided into four 32-bit blocks (A, B, C, D) and 64-bit round keys of the first round,

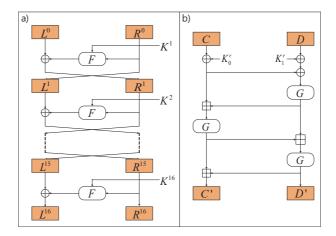


Fig. 1. a) The structure of SEED-128 and b) F function

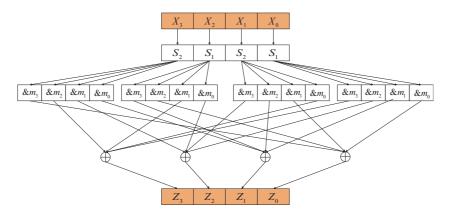


Fig. 2. G function

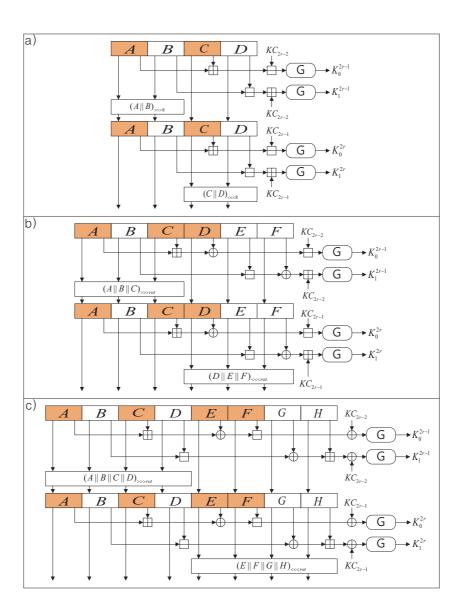
 (K_0^1,K_1^1) are generated as follows. Here, round constants KC_i are generated as follows: $KC_0=0$ x9e3779b9, $KC_i=(KC_0)_{\ll i}$ $(i=1,\cdots,15)$.

$$K_0^1 = G(A + C - KC_0), \quad K_1^1 = G(B - D + KC_0).$$

Round keys of the second and third rounds, (K_0^2, K_1^2) and (K_0^3, K_1^3) are generated as follows, respectively. The other round keys are generated iteratively.

$$A||B \leftarrow (A||B)_{\gg 8}.$$
 $K_0^2 = G(A + C - KC_1), \quad K_1^2 = G(B - D + KC_1).$
 $C||D \leftarrow (C||D)_{\ll 8}.$
 $K_0^3 = G(A + C - KC_2), \quad K_1^3 = G(B - D + KC_2).$

4 Kitae Jeong et al.



 $\bf Fig.\,3.$ The key schedules of a) SEED-128, b) SEED-192 and c) SEED-256

Table 2. The key schedule for the encryption process of SEED-192

```
K = (k_{191}, k_{190}, \cdots, k_0) = (A||B||C||D||E||F);
for (r = 1; r \le 10; r + +) {
K_0^{2r-1} = G(((A+C) \oplus D) - KC_{2r-2});
K_1^{2r-1} = G(((B-E) \oplus F) - KC_{2r-2});
if (r\%3 == 1) A||B||C = (A||B||C)_{\gg 9};
else if (r\%3 == 2) A||B||C = (A||B||C)_{\gg 8};
else A||B||C = (A||B||C)_{\gg 12};
K_0^{2r} = G(((A+C) \oplus D) - KC_{2r-1});
K_1^{2r} = G(((B-E) \oplus F) - KC_{2r-1});
if (r\%3 == 1) D||E||F = (D||E||F)_{\ll 9};
else if (r\%3 == 2) D||E||F = (D||E||F)_{\ll 8};
else D||E||F = (D||E||F)_{\ll 12};
```

3 SEED-192/256

We only focus on the encryption process in the description of SEED-192/256, because the decryption process is explained in the similar way to the encryption process. SEED-192/256 encrypt 128-bit data blocks by iterating the round function F, which is the same as that of SEED-128, 20 and 24 times, respectively (See Figure 1). Considering the trade-off of security and efficiency as like as AES, we adopt 20/24 rounds for SEED-192/256, respectively.

We followed the overall design rationale of original SEED-128 key schedule. The key schedule of SEED-192 generates 64-bit round keys (K_0^r, K_1^r) for total 20 rounds by using a 192-bit secret key $(r=1,\cdots,20)$. Figure 3-b) shows two consecutive rounds in the key schedule of SEED-192. Here, "rot" means a rotation parameter. Table 2 presents the procedure of generating 20 round keys of SEED-192. At first, a 192-bit secret key $K=(k_{191},\cdots,k_0)$ is loaded to six registers (A,B,C,D,E,F). Secondly, a round key (K_0^1,K_1^1) for the first round is generated by using a round constant $KC_0(=0$ x9e3779b9) and G function. Generally, a round key (K_0^r,K_1^r) for the r-th round is generated by using a round constant KC_{r-1} and G function. Here, $KC_i=(KC_0)_{\ll i}$ $(i=0,\cdots,19)$. Six registers are updated by rot-bit circular rotations. In odd rounds, registers A,B,C are updated and registers D,E,F are updated in even rounds. A rotation parameter rot(=9,8,12) is used one and another to satisfy the on-the-fly encryption.

The key schedule of SEED-256 is similar to that of SEED-192, as shown in Figure 3-c). It generates 24 round keys by using a 256-bit secret key. It consists of eight registers and a rotation parameter rot (=9,11,12), which satisfies the on-the-fly encryption. Eight registers are updated by a rot-bit circular rotation of four registers. A round constant KC_i is generated as follows; $KC_i = (KC_0)_{\ll i}$ ($KC_0 = 0$ x9e3779b9 and $i = 0, \cdots, 23$).

4 Security Analysis

We analyze the security of SEED-192/256 against well-known cryptanalysis. As a result, we claim that SEED-192/256 is secure enough for cryptographic applications.

4.1 Existing attacks on SEED-128 without using its key schedule algorithm

Until now, several attacks on SEED-128 such as differential cryptanalysis and linear cryptanalysis have been proposed. However they did not use the weakness of key schedule and were applied only the reduced version of SEED-128. For example, a differential attack on a seven-round SEED-128 was proposed in [7]. It uses a 6-round differential characteristic with probability 2^{-124} . Since SEED-192/256 are the same as SEED-128 except the number of rounds and the key schedule, we claim that SEED-192/256 are secure against attacks which do not use the weakness of key schedule.

4.2 Security of Key Schedule

Weak Keys In [2, 4], the security on key schedule of SEED-128 has been evaluated. They proved that SEED-128 does not have weak keys and equivalent keys. In the similar reason, we can prove that there do not exist them in SEED-192/256. Key schedules of SEED-192/256 are constructed by using the similar design rationale on that of SEED-128. That is, key schedules of SEED-192/256 generate round keys by using simple operations (addition, subtraction and circular rotation) and G function. Thus we can construct equations in terms of registers, round constants and input values of G function. However, round constants for each round are different from each other and rotation parameters rot of SEED-192/256 are not 8 (SEED-128), but (9,8,12) and (9,11,12), respectively. Thus, input values of G function for every rounds are different from each other. It means that SEED-192/256 do not have weak keys and equivalent keys.

The property in [2] In [2], it has shown that it is possible to construct related-keys satisfying the following property; For $i=1,\cdots,16$, we can construct 256 related-keys satisfying that all differences of K_0^i are zero. Similarly, we can produce other 256 related-keys satisfying that all differences of K_1^i are zero. However, since a secret key is randomly chosen, the probability that this event occurs is very low. In cases of SEED-192/256, these algorithms use additional XOR operations and different rotation parameters per 3 rounds to generate round keys. Thus it can be easily shown that SEED-192/256 do not hold the above property of SEED-128.

Secret key differences $(A B C D E F)$	Probability
0000 0000 0000 0100 0000 0000	2^{-14}
0000 0000 0000 0000 0000 0001	2^{-18}
0000 0000 0000 0001 0000 0000	2^{-18}
0000 0000 0000 0010 0000 0000	2^{-18}
0000 0000 0000 1000 0000 0000	2^{-18}
0000 0000 0000 0000 0001 0000	2^{-20}
0000 0000 0000 0000 0100 0000	2^{-20}
0000 0000 0000 0100 0000 0100	2^{-20}
0000 0000 0000 0000 0000 0100	2^{-22}
0000 0000 0000 0000 0010 0000	2^{-22}

Table 3. Top 10 differential characteristics for Mini-192 $\,$

Difference Propagations To evaluate the difference propagation in the key schedule, as a toy example, we consider mini-key schedules of SEED-192/256, Mini-192/256 which use 4-bit registers and do not use G function.

If a rotation parameter rot of Mini-192 is 3, 2, 6 and a secret key difference is 0x000400, we can construct a differential characteristic for Mini-192 with maximal probability 2^{-14} . Here, we assume that the output difference of addition/subtraction operations has the minimal hamming weight. Table 3 presents top 10 secret key differences which construct difference characteristics for Mini-192 with highest probability. As shown in Table 3, secret key differences, where the difference of registers (A, B, C) is zero and registers (D, E, F) have 1-bit active differences, construct difference characteristics with high probability. Since registers D and F are used in XOR operations, active differences of these registers do not degenerate probabilities of difference characteristics. Thus, if there exists 1-bit active differences in registers (D, E, F), we can construct a difference characteristic with high probability.

Table 4 presents input differences of each round, given that secret key differences are 0x000400 and 0x000001, respectively. From this table, we propose the following two conditions in order to construct difference characteristics with high probability.

- 1. Many active differences happen in most significance bits of each register.
- 2. Few active differences happen in the register E.

As shown in Table 4, 0x000400 and 0x000001 have active differences in most significance bits in six rounds. However, in the case of 0x000001, active differences in the register E happen in four rounds. On the other hand, in the case of 0x000400, active differences in the register E happen in two rounds and these active differences happen in only most significance bits. Thus, 0x000400 constructs a difference characteristic with higher probability.

Since experimental results on Mini-192 are deduced by assuming that the output difference of addition/subtration operations has the minimal hamming weight, above two conditions are dependent on only bit positions happening

Table 4. Differential propagations of Mini-192

	C	(A D C D E E)	
Round	Secret key differences $(A B C D E F)$		
	0000 0000 0000 0100 0000 0000	0000 0000 0000 0000 0000 0001	
1	0000 0000 0000 0100 0000 0000	0000 0000 0000 0000 0000 0001	
2	0000 0000 0000 0100 0000 0000	0000 0000 0000 0000 0000 0001	
3	0000 0000 0000 0000 0000 0010	0000 0000 0000 0000 0000 1000	
4	0000 0000 0000 0000 0000 0010	0000 0000 0000 0000 0000 1000	
5	0000 0000 0000 0000 0000 1000	0000 0000 0000 0000 00 1 0 0000	
6	0000 0000 0000 0000 0000 1000	0000 0000 0000 0000 00 1 0 0000	
7	0000 0000 0000 0010 0000 0000	0000 0000 0000 1000 0000 0000	
8	0000 0000 0000 0010 0000 0000	0000 0000 0000 1000 0000 0000	
9	0000 0000 0000 0000 0000 0001	0000 0000 0000 0000 0000 0100	
10	0000 0000 0000 0000 0000 0001	0000 0000 0000 0000 0000 0100	
11	0000 0000 0000 0000 0000 0100	0000 0000 0000 0000 000 1 0000	
12	0000 0000 0000 0000 0000 0100	0000 0000 0000 0000 000 1 0000	
13	0000 0000 0000 0001 0000 0000	0000 0000 0000 0100 0000 0000	
14	0000 0000 0000 0001 0000 0000	0000 0000 0000 0100 0000 0000	
15	0000 0000 0000 1000 0000 0000	0000 0000 0000 0000 0000 0010	
16	0000 0000 0000 1000 0000 0000	0000 0000 0000 0000 0000 0010	
17	0000 0000 0000 0000 0000 0010	0000 0000 0000 0000 0000 1000	
18	0000 0000 0000 0000 0000 0010	0000 0000 0000 0000 0000 1000	
19	0000 0000 0000 0000 1000 0000	0000 0000 0000 0010 0000 0000	
20	0000 0000 0000 0000 1000 0000	0000 0000 0000 0010 0000 0000	
Prob.	2^{-14}	2^{-18}	

active differences. Thus, we expect that these conditions are also applied to the key schedule of SEED-192. To evaluate the difference propagation on the key schedule of SEED-192, we consider total 192 cases where the difference of registers has only a 1-bit active difference. As the result, assuming that only registers (A,B,C) have a 1-bit active difference and G function is not considered, there exist 30 difference characteristics with probability 2^{-36} and the probability of other difference characteristics is 2^{-40} . Similarly, assuming that only registers (D,E,F) have a 1-bit active difference and G function is not considered, there exist 30, 44 and 22 difference characteristics with probability 2^{-24} , 2^{-26} and 2^{-28} , respectively. Table 5 presents input differences of each round for secret key differences (1) and (2) which construct difference characteristics with probability 2^{-24} and 2^{-26} , respectively.

As shown in Table 5, two conditions of Mini-192 are also applied to the key schedule of SEED-192. That is, two secret key differences have the same active differences in the register E (the second condition) but the first secret key difference (1) has more active differences in most significant bits than the second one (the first condition).

	Secret key differences $(D E F)$		
Round	$0000004_{\rm x} 0000000_{\rm x} 0000000_{\rm x} $	$ 00000000_{x} 00040000_{x} 00000000_{x} $	
1	$0000004_{\rm x} 0000000_{\rm x} 0000000_{\rm x}$	$ 00000000_{x} 00040000_{x} 00000000_{x} $	
2	$00000004_{x} 00000000_{x} 00000000_{x}$	$00000000_{x} 00040000_{x} 00000000_{x}$	
3	$00000800_{x} 00000000_{x} 00000000_{x}$	$00000000_{\rm x} 08000000_{\rm x} 00000000_{\rm x} $	
4	$00000800_{\rm x} 00000000_{\rm x} 00000000_{\rm x}$	$00000000_{\rm x} 08000000_{\rm x} 00000000_{\rm x}$	
5	$00080000_{\rm x} 00000000_{\rm x} 00000000_{\rm x}$	$00000008_{\rm x} 00000000_{\rm x} 00000000_{\rm x}$	
6	$00080000_{\rm x} 00000000_{\rm x} 00000000_{\rm x}$	$ 00000008_{\rm x} 00000000_{\rm x} 000000000_{\rm x}$	
7	$80000000_{\rm x} 00000000_{\rm x} 00000000_{\rm x}$	$ 00008000_{\rm x} 00000000_{\rm x} 00000000_{\rm x} $	
8	$80000000_{\rm x} 00000000_{\rm x} 00000000_{\rm x} $	$00008000_{\rm x} 00000000_{\rm x} 00000000_{\rm x}$	
9	$00000000_{\rm x} 00000000_{\rm x} 00000100_{\rm x} $	$01000000_{\rm x} 00000000_{\rm x} 00000000_{\rm x} $	
10	$00000000_{\rm x} 00000000_{\rm x} 00000100_{\rm x} $	$01000000_{\rm x} 00000000_{\rm x} 00000000_{\rm x}$	
11	$00000000_{\rm x} 00000000_{\rm x} 00010000_{\rm x} $	$00000000_{\rm x} 00000000_{\rm x} 00000001_{\rm x}$	
12	$00000000_{\rm x} 00000000_{\rm x} 00010000_{\rm x}$	$ 00000000_{\rm x} 00000000_{\rm x} 00000001_{\rm x}$	
13	$00000000_{\rm x} 00000000_{\rm x} 10000000_{\rm x}$	$ 00000000_{\rm x} 00000000_{\rm x} 00001000_{\rm x}$	
14	$00000000_{\rm x} 00000000_{\rm x} 10000000_{\rm x}$	$ 00000000_{x} 00000000_{x} 00001000_{x} $	
15	$ 00000000_{x} 00000020_{x} 00000000_{x} $	$ 00000000_{x} 00000000_{x} 00200000_{x} $	
16	$00000000_{\rm x} 00000020_{\rm x} 00000000_{\rm x} $	$ 00000000_{\rm x} 00000000_{\rm x} 00200000_{\rm x}$	
17	$ 00000000_{x} 00002000_{x} 00000000_{x} $	$ 00000000_{x} 00000000_{x} 20000000_{x} $	
18	$00000000_{\rm x} 00002000_{\rm x} 00000000_{\rm x} $	$ 00000000_{x} 00000000_{x} 20000000_{x} $	
19	$00000000_{\rm x} 02000000_{\rm x} 00000000_{\rm x} $	$00000000_{\rm x} 00000200_{\rm x} 00000000_{\rm x}$	
20	$ 00000000_{x} 02000000_{x} 00000000_{x} $	$ 00000000_{x} 00000200_{x} 00000000_{x} $	
Prob.	2^{-24} (Not including G function)	2^{-26} (Not including G function)	
1 100.	2^{-144} (Including G function)	2^{-146} (Including G function)	

Table 5. Differential propagations on the key schedule of SEED-192

If there exist active input differences of G function, there exists at least one active S-box in G function. The maximal differential probability on S-box is 2^{-6} [4]. Thus we should consider probability 2^{-6} for each active S-box. If the hamming weight of secret key difference is 1, there exist total 20 active input differences of G function. It means that the key schedule of SEED-128 has enough security against the related-key attack. For example, considering a secret key difference (1), the maximal differential probability on the key schedule of SEED-192 is $2^{-144} (= 2^{-24} \cdot 2^{-6 \cdot 20})$.

Two conditions on the key schedule of SEED-192 are also applied to that of SEED-256. We consider total 256 cases where the difference of registers has only a 1-bit active difference. As the result, assuming that G function is not considered, there exist 12 difference characteristics with maximal probability 2^{-16} . These 12 secret key differences have a 1-bit active difference in registers (E, F, G, H). For these secret key differences, there exist total 24 active input differences of G function. It means that the maximal differential probability on the key schedule of SEED-256 is $2^{-160} (= 2^{-16} \cdot 2^{-6\cdot24})$. Therefore, SEED-256 has enough security against the related-key attack.

^{*} Difference of registers (A, B, C) is zero.

5 Software Implementation

We performed simulations on SEED-128/192/256 (along with AES-128/192/256) with the following platform: Intel personal computer, with an Intel(R) Core(TM)2 Quad CPU Q6600, 2.40 GHz clock speed, 2GB RAM, running Windows XP Professional Edition. Using MSVC/C++ 6.0, we executed 2²⁰ simulations. Table 6 presents results of simulations. Here, in the encryption/decryption process, the part of generating round keys is not included.

Algorithm		Encryption/Decryption	
Aigortiiiii	(cycles)	cycles/byte	Mbps
AES-128	1022	16.9	1137.3
AES-192	1201	20.1	957.0
AES-256	1459	23.2	828.0
SEED-128		51.5	372.8
SEED-192	438	63.9	300.6
SEED-256	634	76.1	252.2

Table 6. Comparison of software performance between AES and SEED

6 Conclusion

In this paper, we proposed 128-bit block ciphers SEED-192/256, which support 192/256-bit secret keys, and evaluated the security of them against well-known cryptanalysis. We anticipate that SEED-192/256 can be applied to various environments together with SEED-128.

References

- 1. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima and T. Tokita, Camellia: A 128-bit Block Cipher Suitable Multiple Platforms - Design and Analysis, SAC'00, LNCS 2012, pp. 39–56, Springer-Verlag, 2000.
- 2. CRYPTREC Evaluation Committee, SEED Evaluation Report, 2002.
- 3. Korea Information Security Agency, SEED Algorithm Specification. Available at http://www.kisa.or.kr/kisa/seed/down/SEED_Specification_english.pdf.
- 4. Korea Information Security Agency, SEED Algorithm Self Evaluation. Available at http://www.kisa.or.kr/ kisa/seed/down/SEED_Self_Evaluation-English.pdf.
- D. Kwon, J. Kim, S. Park, S. Sung, Y. Sohn, J. Song, Y. Yeom, E. Yoon, S. Lee, J. Lee, S. Chee, D. Han and J. Hong, *New Block Cipher: ARIA*, ICISC'03, LNCS 2971, pp. 443–456, Springer-Verlag, 2003.
- 6. NIST, FIPS 197: Advanced Encryption Standard, 2001.
- 7. H. Yanami and T. Shimoyama, Differential Cryptanalysis of a Reduced-Round SEED, SCN'02, LNCS 2576, pp. 186–198, Springer-Verlag, 2003.