Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Differential attack on nine rounds of the SEED block cipher

Jiqiang Lu[a,*], Wun-She Yap[b,c,1], Matt Henricksen[a], Swee-Huay Heng[c]

[a] *Institute for Infocomm Research, Agency for Science, Technology and Research, 1 Fusionopolis Way, Singapore 138632, Singapore*
[b] *Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Kuala Lumpur 53300, Malaysia*
[c] *Faculty of Information Science and Technology, Multimedia University, Melaka 75450, Malaysia*

A R T I C L E   I N F O

Article history:
Received 22 April 2013
Received in revised form 1 October 2013
Accepted 4 November 2013
Available online 6 November 2013
Communicated by D. Pointcheval

Keywords:
Cryptography
Cryptology
Block cipher
SEED
Differential cryptanalysis

A B S T R A C T

The SEED block cipher has a 128-bit block length, a 128-bit user key and a total number of 16 rounds. It is an ISO international standard. In this letter, we describe two 7-round differentials with a trivially larger probability than the best previously known one on SEED, and present a differential cryptanalysis attack on a 9-round reduced version of SEED. The attack requires a memory of $2^{69.71}$ bytes, and has a time complexity of $2^{126.36}$ encryptions with a success probability of 99.9% when using 2^{125} chosen plaintexts, or a time complexity of $2^{125.36}$ encryptions with a success probability of 97.8% when using 2^{124} chosen plaintexts. Our result is better than any previously published cryptanalytic results on SEED in terms of the numbers of attacked rounds, and it suggests for the first time that the safety margin of SEED decreases below half of the number of rounds.

1. Introduction

The SEED [9] block cipher was designed by a group of Korean cryptographers in 1998. It has a 128-bit block length, a 128-bit user key and a total number of 16 rounds. SEED became a Korean national industrial association standard [16] in 1999, and was adopted as an ISO international standard [8] in 2005. Currently, SEED has been used in reality, mostly by banks and companies in Korea, to protect the privacy of the users and the transaction data in security applications like e-commerce and financial services [18]. Besides, it was included in PKCS #11 on Cryptographic Token Interface Standard [13], and was proposed by IETF [3] for Cryptographic Message Syntax (CMS) [4], Transport Layer Security (TLS) [5], Secure Real-time Transport Protocol (SRTP) [7] and IPsec [6]. And, Mozilla Firefox web browser supports the SEED algorithm now [11].

The SEED designers first analysed the security of SEED against differential cryptanalysis [1] as well as certain other cryptanalytic techniques, and they indicated that a 6-round differential characteristic of SEED would have a probability of at least 2^{-130}, meaning that there would not exist any effective 6-round differential characteristic for SEED. However, in 2002 Yanami and Shimoyama [17] presented three 6-round differential characteristics with probability 2^{-124}, and finally used them to conduct a differential attack on 7-round SEED (faster than exhaustive key search). In 2011, Sung [15] described a 7-round differential with probability 2^{-122} for SEED, by summing the probabilities of many 7-round differential characteristics with the same input and output differences, and finally gave a differential attack on 8-round SEED; Sung also described a 7-round differential with probability 2^{-124} of SEED. Sung's attack on 8-round SEED is the best previously published cryptanalytic result on the SEED cipher algorithm in terms of the numbers of attacked rounds.

In this letter, we further investigate the security of SEED against differential cryptanalysis. We find there exist two 7-round differentials with a probability of trivially larger than the probability of Sung's best 7-round differential, plus seventeen 7-round differentials with a probability of

[*] Corresponding author.
E-mail addresses: lvjiqiang@hotmail.com, jlu@i2r.a-star.edu.sg (J. Lu), yapws@utar.edu.my (W.-S. Yap), mhenricksen@i2r.a-star.edu.sg (M. Henricksen), shheng@mmu.edu.my (S.-H. Heng).
[1] The author was with Institute for Infocomm Research (Singapore) when the work was completed.

Table 1
Main cryptanalytic results on SEED.

Attack type	Rounds	Data	Memory	Time	Success prob.	Source
Differential	7	2^{126}	2^{67}	$2^{126.37}$	68.8%	[17]
cryptanalysis	8	2^{125}	2^{67}	$2^{125.17}$	98.1%	[15]
	9	2^{125}	$2^{69.71}$	$2^{126.36}$	99.9%	Section 4
	9	2^{124}	$2^{69.71}$	$2^{125.36}$	97.8%	Section 4

Data unit: chosen plaintexts, Memory unit: bytes, Time unit: encryptions.

trivially larger than the probability of Sung's second best 7-round differential. More importantly, we devise a differential attack on 9-round SEED, which requires a memory of $2^{69.71}$ bytes and has a time complexity of $2^{126.36}$ encryptions with a success probability of 99.9% when using 2^{125} chosen plaintexts, or a time complexity of $2^{125.36}$ encryptions with a success probability of 97.8% when using 2^{124} chosen plaintexts. This is the first published cryptanalytic attack on 9-round SEED, and it suggests that the safety margin of SEED decreases below half of the number of rounds. Table 1 summarises both previous and our main cryptanalytic results on SEED.

The remainder of this letter is organised as follows. In the next section, we give the notation and describe the SEED block cipher. In Section 3, we describe the 7-round differentials of SEED. In Section 4, we present our differential attack on 9-round SEED. Section 5 concludes the letter.

2. Preliminaries

In this section we give the notation used throughout this letter, and then briefly describe the SEED block cipher.

2.1. Notation

In all descriptions we assume that the bits of an n-bit value are numbered from 1 to n from right to left, with the most significant bit being the n-th bit, a number without a prefix expresses a decimal number, and a number with prefix 0x expresses a hexadecimal number. We use the following notation.

$\oplus$	bitwise logical exclusive OR (XOR) operation
&	bitwise logical AND operation
$\boxplus$	addition modulo 2^{32}
$\boxminus$	subtraction modulo 2^{32}
$\lll$	left rotation of a bit string
$\ggg$	right rotation of a bit string
$\|\|$	string concatenation
e	the base of the natural logarithm ($\mathrm{e} = 2.71828\ldots$)
$\star$	an arbitrary value of some length, where two values represented by the $\star$ symbol may be different

2.2. The SEED block cipher

SEED [9] employs a typical Feistel structure with a 128-bit block size and a 64-bit round subkey; however, the round function **F** of SEED is very complex, which is built mainly on the **G** function. Below we describe the **G** and **F** functions specifically.

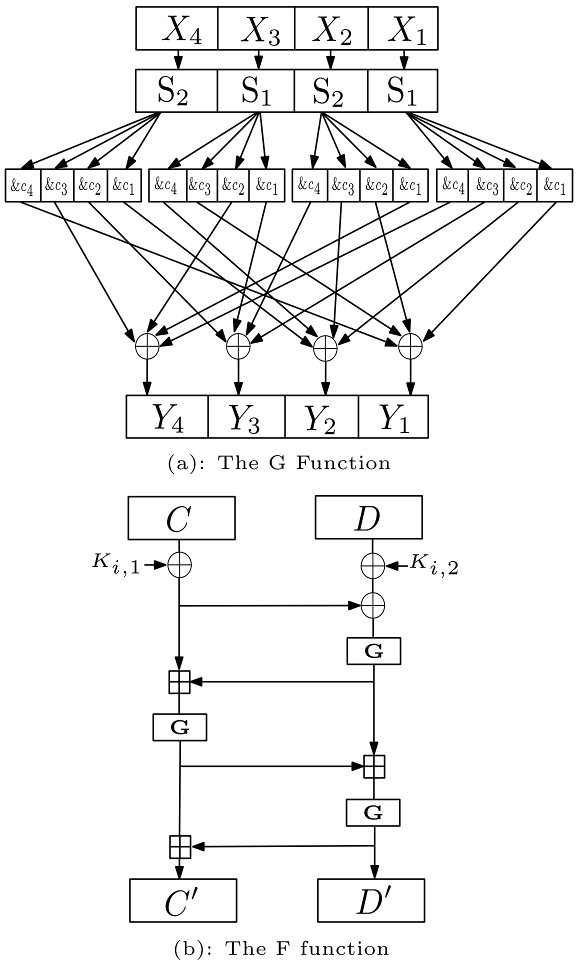

Fig. 1. The structures of the **F** and **G** functions.

– **The G function.** There are two layers in the **G** function, see Fig. 1(a). The first layer involves two S-boxes S_1 and S_2, which are constructed from two different Boolean functions. The second layer is a permutation which processes the output of the first layer, made up of an AND operation with four specific values c_1, c_2, c_3 and c_4 (see [9]), followed by an XOR operation on the expanded 16 blocks. Given a four-byte input (X_4, X_3, X_2, X_1), the **G** function generates a four-byte output (Y_4, Y_3, Y_2, Y_1), as follows.

$$Y_1 = \big(S_1(X_1)\&c_1\big) \oplus \big(S_2(X_2)\&c_2\big) \oplus \big(S_1(X_3)\&c_3\big) \oplus \big(S_2(X_4)\&c_4\big),$$

$$Y_2 = \big(S_1(X_1)\&c_2\big) \oplus \big(S_2(X_2)\&c_3\big) \oplus \big(S_1(X_3)\&c_4\big)$$

$$\oplus \big(S_2(X_4)\&c_1\big),$$

$$Y_3 = \big(S_1(X_1)\&c_3\big) \oplus \big(S_2(X_2)\&c_4\big) \oplus \big(S_1(X_3)\&c_1\big)$$
$$\oplus \big(S_2(X_4)\&c_2\big),$$
$$Y_4 = \big(S_1(X_1)\&c_4\big) \oplus \big(S_2(X_2)\&c_1\big) \oplus \big(S_1(X_3)\&c_2\big)$$
$$\oplus \big(S_2(X_4)\&c_3\big).$$

– **The F function.** As depicted in Fig. 1(b), a 64-bit input block of the **F** function is divided into two 32-bit blocks C and D. After being XORed with the two 32-bit subkeys $K_{i,1}$ and $K_{i,2}$ respectively, the two blocks pass through three layers of **G** function, and finally the two 32-bit output blocks C' and D' of the **F** function are

$$C' = \mathbf{G}\Big(\mathbf{G}\Big(\mathbf{G}\big((C \oplus K_{i,1}) \oplus (D \oplus K_{i,2})\big) \boxplus (C \oplus K_{i,1})\Big)$$
$$\boxplus \mathbf{G}\big((C \oplus K_{i,1}) \oplus (D \oplus K_{i,2})\big)\Big)$$
$$\boxplus \mathbf{G}\Big(\mathbf{G}\big((C \oplus K_{i,1}) \oplus (D \oplus K_{i,2})\big) \boxplus (C \oplus K_{i,1})\Big),$$
$$D' = \mathbf{G}\Big(\mathbf{G}\Big(\mathbf{G}\big((C \oplus K_{i,1}) \oplus (D \oplus K_{i,2})\big) \boxplus (C \oplus K_{i,1})\Big)$$
$$\boxplus \mathbf{G}\big((C \oplus K_{i,1}) \oplus (D \oplus K_{i,2})\big)\Big).$$

SEED uses a total of sixteen 64-bit subkeys K_i for the round functions $(i = 1, 2, \ldots, 16)$, all derived from a 128-bit user key K; and each round subkey K_i is made up of two 32-bit subkeys $K_{i,1}$ and $K_{i,2}$. The key schedule is as follows.

(i) Represent K as four 32-bit words $K = (K_d, K_c, K_b, K_a)$.
(ii) The subkeys are generated as follows, where $\widehat{c_i}$ are specific constants (see [9]).
For $i = 1, 2, \ldots, 16$:
– $K_{i,1} = \mathbf{G}(K_b \boxplus K_d \boxminus \widehat{c_i})$;
– $K_{i,2} = \mathbf{G}(K_c \boxminus K_a \boxplus \widehat{c_i})$;
– If $i \bmod 2 = 1$, then $(K_d||K_c) = (K_d||K_c) \ggg 8$; else $(K_b||K_a) = (K_b||K_a) \lll 8$.

SEED takes a 128-bit plaintext as input, and its encryption procedure is as follows. A 128-bit plaintext P is divided into two 64-bit blocks $P = (P_L, P_R)$. The right 64-bit block P_R is input to the **F** function with the 64-bit round subkey K_1. The output of the **F** function is XORed with the left 64-bit block P_L, which becomes the right 64-bit input block to the second round, and P_R becomes the left 64-bit input block to the second round. After 16 similar encryption rounds, the final 128-bit output is the ciphertext of the plaintext.

3. Seven-round differentials of SEED

In this section, we first describe the 7-round differentials owing to Sung [15], and then present two 7-round differentials with a probability of trivially larger than Sung's best 7-round differential, and seventeen 7-round differentials with a probability of trivially larger than Sung's second best 7-round differential.

3.1. Sung's 7-round differentials

In 2011, Sung [15] presented a 7-round differential $(0x80808000, 0, 0x87808000, 0x80808000) \rightarrow (0x07808000, 0x80808000, 0x80808000, 0)$ with probability 2^{-122} on SEED, and a 7-round differential $(0x80808000, 0, 0x83808000, 0x80808000) \rightarrow (0x03808000, 0x80808000, 0x80808000, 0)$ with probability 2^{-124}, which were obtained by summing the probabilities of many 7-round differential characteristics with the same input and output differences. Refer to [15] for an illustration of the 7-round differentials; or see Fig. 2(a), where $\alpha = 0x80808000$ and $X = 0x87808000/0x83808000$.

It is worthy to mention that our computation shows that the probabilities 2^{-122} and 2^{-124} for Sung's 7-round differentials are actually about $2^{-121.21}$ and $2^{-122.84}$ respectively, if we keep two more significant digits. Subsequently we will use these more accurate probabilities for them.

3.2. Our 7-round differentials

We have performed a computer programming to search 7-round differentials of SEED over sixteen different differential patterns, by considering many 7-round differential characteristics with the same input and output differences. We find some valuable results in only two patterns that are depicted in Fig. 2(a) and (b), as follows.

– Sung's 7-round differential with probability $2^{-121.21}$ is one of the best 7-round differentials (i.e. 7-round differentials with the largest probability) in Pattern (I). In Pattern (I), there is another 7-round differential with probability $2^{-121.21}$ and eight additional 7-round differentials with a probability of larger than the probability of $2^{-122.84}$ of Sung's second best 7-round differential, ranging from $2^{-121.22}$ to $2^{-122.81}$.
– There are two 7-round differentials with probability $2^{-121.07}$ in Pattern (II), which is slightly larger than the probability of $2^{-121.21}$ of Sung's best 7-round differential; and there are eight additional 7-round differentials with a probability of larger than the probability of $2^{-122.84}$ of Sung's second best 7-round differential, ranging from $2^{-121.22}$ to $2^{-122.81}$.

We summarise all these new 7-round differentials in Table 2. As an example, here we give the probabilities for the rounds of one of our best 7-round differentials: $(0x80808000, 0, 0x84808000, 0x83808000) \rightarrow (0x04808000, 0x83808000, 0x80808000, 0)$, which is the first with Pattern (II) given in Table 2 and is also the one that we will use in Section 4. Following Fig. 2(b), we get that the first and third rounds have a probability of approximately $2^{-18.45}$ each, the fourth round has a probability of approximately $2^{-43.79}$, and the fifth and seventh rounds have a probability of approximately $2^{-20.19}$ each. Clearly, the second and sixth rounds have a one probability. Hence, the 7-round differential has a total probability of $2^{-18.45\times 2} \times 2^{-43.79} \times 2^{-20.19\times 2} = 2^{-121.07}$.

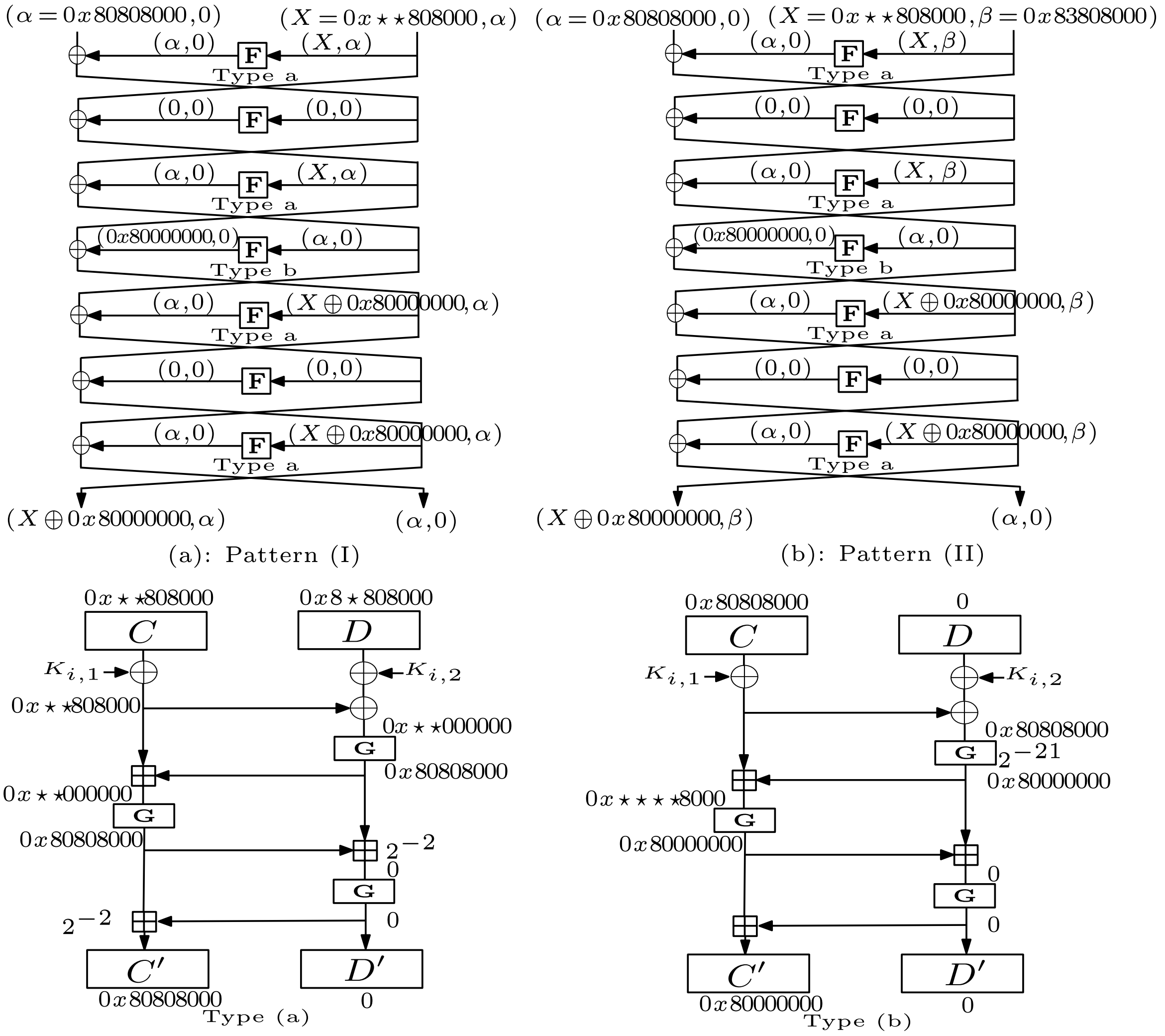

Fig. 2. 7-round differentials of SEED.

Table 2
Our 7-round differentials of SEED.

Pattern	X	$X \oplus$ 0x8000000	Probability
(I)	0x07808000	0x87808000	$2^{-121.21}$
	0x44808000	0xC4808000	$2^{-121.22}$
	0xC4808000	0x44808000	$2^{-121.22}$
	0x45808000	0xC5808000	$2^{-121.96}$
	0xC5808000	0x45808000	$2^{-121.96}$
	0x4C808000	0xCC808000	$2^{-122.55}$
	0xCC808000	0x4C808000	$2^{-122.55}$
	0x47808000	0xC7808000	$2^{-122.81}$
	0xC7808000	0x47808000	$2^{-122.81}$
(II)	0x84808000	0x04808000	$2^{-121.07}$
	0x04808000	0x84808000	$2^{-121.07}$
	0x44808000	0xC4808000	$2^{-121.22}$
	0xC4808000	0x44808000	$2^{-121.22}$
	0x45808000	0xC5808000	$2^{-121.96}$
	0xC5808000	0x45808000	$2^{-121.96}$
	0x00808000	0x80808000	$2^{-122.54}$
	0x80808000	0x00808000	$2^{-122.54}$
	0x47808000	0xC7808000	$2^{-122.81}$
	0xC7808000	0x47808000	$2^{-122.81}$

It is interesting to see from Table 2 that the differentials appear pairwise with the same probability, because of the symmetry of the Feistel structure for forward and backward directions; and our best 7-round differential in Pattern (I) is the counterpart of Sung's best 7-round differential. We would like to mention that there exist a large number of 7-round differentials whose probability is smaller than or equal to the probability of $2^{-122.84}$ of Sung's second best 7-round differential but is still larger than 2^{-128}.

4. Differential attack on 9-round SEED

In this section we devise a differential attack on 9-round SEED, building it on the best 7-round differential with $\alpha = 0x80808000$, $\beta = 0x83808000$, $X = 0x84808000$ (that has a probability of $2^{-121.07}$) we have described in Section 3.2. Thus, $\widehat{X} = X \oplus 0x80000000 = 0x04808000$. The attack consists of an offline precomputation phase and an online attack phase, and without loss of generality we assume the attacked rounds are the first 9 rounds of SEED, that is to say, from Rounds 1 to 9. It is noteworthy that similar cryptanalytic results can be obtained by using certain other 7-round differentials, including Sung's 7-round differentials.

4.1. Precomputation

Suppose x is the 64-bit state immediately after the XOR operations with the round subkey K_9 in the **F** function of Round 9. We precompute a table $\mathcal{T}_x$, so that, given an output difference of the **F** function we only need a single table lookup (memory access) to $\mathcal{T}_x$ to retrieve the pair of values at state x that have the difference $(\alpha, 0)$ and generate the given output difference after the **F** function. We generate the table $\mathcal{T}_x$ as follows.

- For every possible 64-bit value x such that $x < [x \oplus (\alpha, 0)]$:
  - Compute the output difference immediately after the **F** function corresponding to the pair of values $(x, x \oplus (\alpha, 0))$; and we denote it by Δy.
  - Store x into Table $\mathcal{T}_x$ indexed by y.

There are 2^{63} possible values for x and at most 2^{63} possible values for y. Thus, the precomputation requires a memory of $2^{63} \times \frac{64}{8} = 2^{66}$ bytes, and has a time complexity of about $2^{63} \times 2 = 2^{64}$ computations of the **F** function (which are approximately equivalent to $2^{64} \times \frac{1}{9} \approx 2^{60.84}$ 9-round SEED encryptions), and 2^{63} memory accesses (which are equivalent to $\frac{2^{63}}{9} \approx 2^{59.84}$ 9-round SEED encryptions by our estimate given in Section 4.3). There are a total of 2^{63} possible combinations of (x, y), and on average there is only one value of x in every entry y of table $\mathcal{T}_x$ (occasionally, there may actually exist more than one values of x in an entry of table $\mathcal{T}_x$, but it does not affect the correctness of our attack, and it is the case that the number is one on average).

4.2. Attack procedure

Now we can give the following attack procedure for breaking the first 9 rounds of SEED. Recall that $\alpha = 0x80808000$, $\beta = 0x83808000$, $X = 0x84808000$ and $\widehat{X} = 0x04808000$, all of which are fixed for this specific attack. The attack is illustrated in Fig. 3.

1. Choose 2^{ϕ} structures $\mathcal{S}_i$ (a specific value of ϕ will be given below, $i = 1, 2, \ldots, 2^{\phi}$), where a structure is defined to be a set of 2^{64} plaintexts $P_{i,j}$ with the left half taking all the possible 64-bit values, bits (48, 56, 64) of the right half fixed to a certain value such that its complement is not used as bits (48, 56, 64) of the right half of any other structure, and the other 61 bits fixed ($j = 1, 2, \ldots, 2^{64}$). In a chosen-plaintext attack scenario, obtain all the ciphertexts for the 2^{64} plaintexts in each of the 2^{ϕ} structures; we denote by $C_{i,j}$ the ciphertext for plaintext $P_{i,j}$.
2. Choose 2^{ϕ} structures $\widehat{\mathcal{S}_i}$ ($i = 1, 2, \ldots, 2^{\phi}$), where a structure $\widehat{\mathcal{S}_i}$ is obtained by taking the complement of bits (48, 56, 64) of the right half of all the plaintexts $P_{i,j}$ in $\mathcal{S}_i$. In a chosen-plaintext attack scenario, obtain all the ciphertexts for the 2^{64} plaintexts in each $\widehat{\mathcal{S}_i}$.
3. For each pair of structures $(\mathcal{S}_i, \widehat{\mathcal{S}_i})$, perform the following three sub-steps:
   (a) Identify all plaintext–ciphertext quartets $(P_{i,j}, C_{i,j}, \widehat{P}_{i,l}, \widehat{C}_{i,l})$ (or more precisely, the quartets $(P^L_{i,j}, C^R_{i,j}, \widehat{P}^L_{i,l}, \widehat{C}^R_{i,l})$) such that $C^L_{i,j} \oplus \widehat{C}^L_{i,l}$ is equal to $(\alpha, 0)$, using a sorting (or hash) method, for example, by storing $(P_{i,j}, C_{i,j})$ (respectively $(P^L_{i,j}, C^R_{i,j})$) indexed by $C^L_{i,j}$ and storing $(\widehat{P}_{i,j}, \widehat{C}_{i,j})$ (respectively $(\widehat{P}^L_{i,j}, \widehat{C}^R_{i,j})$) indexed by $\widehat{C}^L_{i,j} \oplus (\alpha, 0)$, where $P^L_{i,j}$ and $\widehat{P}^L_{i,l}$ represent the left halves of $P_{i,j}$ and $\widehat{P}_{i,l}$, respectively, $C^R_{i,j}$ and $\widehat{C}^R_{i,l}$ represent the right half of $C_{i,j}$ and $\widehat{C}_{i,l}$, respectively, and $C^L_{i,j}$ and $\widehat{C}^L_{i,l}$ represent the left halves of $C_{i,j}$ and $\widehat{C}_{i,l}$, respectively ($1 \leqslant l \leqslant 2^{64}$).
   (b) Store the satisfying plaintext–ciphertext quartets $(P_{i,j}, \widehat{P}_{i,l}, C_{i,j}, \widehat{C}_{i,l})$ (or more concisely, the pairs $(C^L_{i,j}, C^R_{i,j} \oplus \widehat{C}^R_{i,l})$) into a table $\mathcal{T}_P$ indexed by $P^L_{i,j} \oplus \widehat{P}^L_{i,l}$.
   (c) Guess a value for the round subkey K_1, and perform the following sub-steps:
      (i) Partially encrypt the two right halves respectively from the plaintexts in $\mathcal{S}_i$ and $\widehat{\mathcal{S}_i}$ through the **F** function of Round 1, and we denote by Δ the output difference of the **F** function.
      (ii) Access entry $(\Delta \oplus (X, \beta))$ in table $\mathcal{T}_P$ to get the plaintext–ciphertext quartet, and we assume it is $(P_{i,j}, \widehat{P}_{i,l}, C_{i,j}, \widehat{C}_{i,l})$ (respectively the pair $(C^L_{i,j}, C^R_{i,j} \oplus \widehat{C}^R_{i,l})$).
      (iii) Given the selected plaintext–ciphertext quartet $(P_{i,j}, \widehat{P}_{i,l}, C_{i,j}, \widehat{C}_{i,l})$ (respectively the selected pair $(C^L_{i,j}, C^R_{i,j} \oplus \widehat{C}^R_{i,l})$), access entry $((\widehat{X}, \beta) \oplus C^R_{i,j} \oplus \widehat{C}^R_{i,l})$ in the precomputation table $\mathcal{T}_x$ to get the intermediate value(s) immediately after the XOR operations with the round subkey K_9 in the **F** function of Round 9, and we denote it by z. Then, compute two possible values for K_9 as $z \oplus C^L_{i,j}$ and $z \oplus C^L_{i,j} \oplus (\alpha, 0)$. (Note that $C^L_{i,j} \oplus (\alpha, 0) = \widehat{C}^L_{i,l}$.)

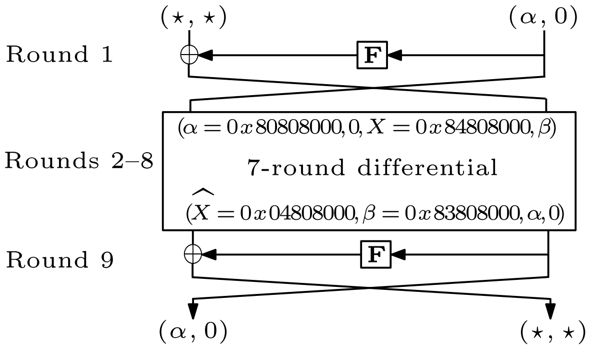
$(\star, \star)$
$(\alpha, 0)$
Round 1
F
$(\alpha = 0x80808000, 0, X = 0x84808000, \beta)$
Rounds 2–8
7-round differential
$(\widehat{X} = 0x04808000, \beta = 0x83808000, \alpha, 0)$
Round 9
F
$(\alpha, 0)$
$(\star, \star)$

Fig. 3. Differential attack on 9-round SEED.

(iv) Recover the corresponding user key from each of the obtained candidate values on (K_1, K_9), and check whether the user key is correct with one or more plaintext–ciphertext pairs. If it passes this test, then it is very likely to be the correct user key, and we terminate the procedure; otherwise, repeat Step 3 with another pair of structures.

Notice that there are only two possible 64-bit values for the right halves of the plaintexts in a pair of structures $(\mathcal{S}_i, \widehat{\mathcal{S}_i})$, and it is simple to recover the user key from a candidate (K_1, K_9) by the key schedule of SEED. The overall idea used in this attack is similar to that used by Biham and Shamir [2] in 1992 to break the full DES [12] block cipher.

4.3. Attack complexity

The attack requires $2 \times 2^{\phi} \times 2^{64} = 2^{\phi+65}$ chosen plaintexts. Steps 1 and 2 have a time complexity of $2^{\phi+65}$ 9-round SEED encryptions. Observe that we collect another pair of plaintext structures after testing a pair of plaintext structures, so that we can reuse the memory for storing the pair of plaintext structures; further, for a structure $\mathcal{S}_i$ of 2^{64} plaintexts $P_{i,j}$, we store the (identical) right halves $P^R_{i,j}$ of the plaintexts only once, then store $(P^L_{i,j}, C^R_{i,j})$ indexed by $C^L_{i,j}$, and similar for the other structure $\widehat{\mathcal{S}_i}$. Thus, the memory complexity of the online attack is dominated by the space for storing the pair of plaintext structures and table $\mathcal{T}_P$, which is approximately $2^{64} \times (8+8) \times 2 + 2^{64} \times (8+8) = 3 \times 2^{68}$ bytes.

Next we analyse Steps 3(a)–3(c) for a pair of structures $(\mathcal{S}_i, \widehat{\mathcal{S}_i})$. Step 3(a) has a time complexity of about $2^{64} \times 2 = 2^{65}$ memory accesses. There is a 64-bit filtering condition in Step 3(a), and thus it is expected that there are $2^{64} \times 2^{64} \times 2^{-64} = 2^{64}$ satisfying ciphertext pairs $(C_{i,j}, \widehat{C}_{i,l})$ after Step 3(a). Thus, Step 3(b) has a time complexity of about 2^{64} memory accesses. Since there are only two possible 64-bit values for the right halves of the plaintexts in a pair of structures $(\mathcal{S}_i, \widehat{\mathcal{S}_i})$, Step 3(c)(i) has a time complexity of approximately $2^{64} \times 2 \times \frac{1}{9} = \frac{2^{65}}{9}$ 9-round SEED encryptions. For a guessed K_1, there is about only one memory access in Step 3(c)(ii)/3(c)(iii), and Step 3(c)(iv) has a time complexity of about $4 \times 2 = 8$ computations of the **G** function to recover the two possible user keys by the key schedule of SEED, which are approximately equal to three computations of the **F** function, plus 2 trial 9-round SEED encryptions. Hence, for all 2^{ϕ} pairs of structures, Step 3 has a total time complexity of approximately $2^{\phi} \times \frac{2^{65}}{9} + 2^{\phi} \times 2^{64} \times \frac{3}{9} + 2^{\phi} \times 2^{64} \times 2 = \frac{23}{9} \times 2^{\phi+64}$ 9-round SEED encryptions and $2^{\phi} \times 2^{65} + 2^{\phi} \times 2^{64} + 2^{\phi} \times 2^{64} \times (1+1) = 5 \times 2^{\phi+64}$ memory accesses.

In total, the online attack has a time complexity of approximately $2^{\phi+65} + \frac{23}{9} \times 2^{\phi+64} = \frac{41}{9} \times 2^{\phi+64}$ 9-round SEED encryptions and $5 \times 2^{\phi+64}$ memory accesses.

Just like what was mentioned in [10], the question that how many memory accesses (table lookups) are equivalent to one SEED encryption in terms of time depends closely on the used platform and SEED implementation as well as the storage location of the sorted table and, in theoretical block cipher cryptanalysis, it is usually assumed by default that a table is stored in an ideal place, RAM say, like an S-box table; and it takes an almost constant time to access an entry in a sorted table, independently of the number of entries. Thus, an extremely conservative estimate is: 9 memory accesses equal a 9-round SEED encryption in terms of time, assuming that the **F** function without the XOR operations with a round subkey is precomputed in a table and is equivalent to one memory access (neglecting the computational complexity for other operations and the key schedule); that is, one round is equivalent to one memory access. As a consequence, the total time complexity of the online attack is $\frac{41}{9} \times 2^{\phi+64} + \frac{5 \times 2^{\phi+64}}{9} = \frac{46}{9} \times 2^{\phi+64}$ 9-round SEED encryptions.

Take the complexity for the precomputation into consideration, so the attack requires a total memory of approximately $3 \times 2^{68} + 2^{66} \approx 2^{69.71}$ bytes and a total time complexity of approximately $\frac{46}{9} \times 2^{\phi+64} \approx 2^{126.36}$ 9-round SEED encryptions, when we let $\phi = 60$.

The attack succeeds if there is at least one right plaintext pair. When $\phi = 60$, for each candidate (K_1, K_9) there are $2^{\phi} \times 2^{64} = 2^{124}$ candidate plaintext pairs that have an input difference $(\alpha, 0, \star, \star)$ to Round 2 and an output difference $(\star, \star, \alpha, 0)$ immediately after Round 8 (that is, after Step 3(a)), and thus the attack has an expected success probability of approximately $1 - (1 - 2^{-121.07})^{2^{124}} \approx 1 - e^{-2^{2.93}} \approx 99.9\%$.

We can obtain a faster attack with a smaller success probability by using a smaller number of data, for example, if we let $\phi = 59$ (that is, 2^{59} pairs of plaintext structures — a total of 2^{124} chosen plaintexts), then the resulting attack requires the same amount of memory of $2^{69.71}$ bytes, but has a total time complexity of $\frac{46}{9} \times 2^{\phi+64} \approx 2^{125.36}$ 9-round SEED encryptions, with an expected success probability of approximately $1 - (1 - 2^{-121.07})^{2^{123}} \approx 1 - e^{-2^{1.93}} \approx 97.8\%$.

Observe that all those table lookups are operated on either 64-bit or 128-bit data with a 64-bit index, and we assume each table lookup is done with a single memory access, as widely adopted in theoretical analysis. However, on a 64-bit computer in reality, it takes two memory accesses to retrieve 128-bit data with a 64-bit index, thus the resulting number of memory accesses will be $2^{\phi} \times 2^{65} \times 2 + 2^{\phi} \times 2^{64} \times 2 + 2^{\phi} \times 2^{64} \times (2+1) = 9 \times 2^{\phi+64}$, and the resulting total time complexity of the attack will be $\frac{41}{9} \times 2^{\phi+64} + \frac{9 \times 2^{\phi+64}}{9} \approx 2^{\phi+66.48}$ 9-round SEED encryptions, still smaller than that for exhaustive key search when $\phi = 60$ or 59.

It is worthy to notice that the structure pairs $(\mathcal{S}_i, \widehat{\mathcal{S}_i})$ can be generated by using different keys, like what Biham and Shamir [2] mentioned for their DES attack. Under this scenario, we can find one or more of the used keys, as long as there is a right plaintext pair.

4.4. Notes

In this subsection we describe two time–memory trade-offs to the above attack and compute the success probabilities of Yanami and Shimoyama's 7-round attack and Sung's 8-round attack.

4.4.1. Note 1

Observe that the round functions of Rounds 1 and 9 have the same input difference $(\alpha, 0)$, thus we can obtain a time–memory trade-off by making the following revisions to the above attack: For every satisfying plaintext–ciphertext quartet after Step 3(a), with a single lookup to entry $((X, \beta) \oplus P^L_{i,j} \oplus \widehat{P}^L_{i,l})$ in table $\mathcal{T}_x$, we retrieve intermediate value(s) immediately after the XOR operations with the round subkey K_1 in the $\mathbf{F}$ function of Round 1, and we denote it by $\widehat{z}$, then compute the two possible values for K_1 as $K_1 = \widehat{z} \oplus P^R_{i,j}$ and $K_1 = \widehat{z} \oplus \widehat{P}^R_{i,l}$; then retrieve the two possible values for K_9 with a single lookup to $\mathcal{T}_x$, and finally check whether one of the four combinations on (K_1, K_9) produces the correct user key.

For a structure, we only need to store $(P^L_{i,j}, C^R_{i,j})$ indexed by $C^L_{i,j}$. As a result, this time–memory trade-off requires a total memory of approximately $2^{64} \times (8 + 8) \times 2 + 2^{66} \approx 2^{69.17}$ bytes. This time–memory trade-off has a total time complexity of approximately $2^{125} + \frac{2^{60} \times 2^{64} \times (1+1+1)}{9} + 2^{60} \times 2^{64} \times \frac{4\times 4}{9\times 3} + 2^{60} \times 2^{64} \times 4 \approx 2^{126.8}$ 9-round SEED encryptions when using 2^{60} pairs of plaintext structures, with the same success probability of 99.9%; and it has a total time complexity of approximately $2^{124} + \frac{2^{59} \times 2^{64} \times (1+1+1)}{9} + 2^{59} \times 2^{64} \times \frac{4\times 4}{9\times 3} + 2^{59} \times 2^{64} \times 4 \approx 2^{125.8}$ 9-round SEED encryptions when using 2^{59} pairs of plaintext structures, with the same success probability of 97.8%. Each version is slightly slower than the corresponding attack version described in the last subsection, but requires a slightly smaller memory.

4.4.2. Note 2

Another time–memory trade-off can be obtained by precomputing a table so that we can retrieve the candidate subkey K_9 with a single table lookup to this table, given a pair of input blocks to the $\mathbf{F}$ function with difference $(\alpha, 0)$ and their output difference immediately after the $\mathbf{F}$ function. A straightforward way to generate such a table is as follows.

For every possible 64-bit value u such that $u < [u \oplus (\alpha, 0)]$:

For every possible value of the round subkey K_9:

- Compute $w = \mathbf{F}(u, K_9) \oplus \mathbf{F}(u \oplus (\alpha, 0), K_9)$.
- Store K_9 into the table indexed by (u, w).

There are 2^{63} possible values for u and at most $2^{64} - 1$ possible values for w. Thus, the precomputation requires a memory of $2^{63} \times (2^{64} - 1) \times \frac{64}{8} \approx 2^{130}$ bytes (this is smaller than, more specifically, a quarter of, a memory of $2^{132} (= 2^{128} \times 16)$ bytes required by the dictionary or codebook attack), and has a time complexity of about $2^{63} \times 2^{64} \times 2 = 2^{128}$ computations of the $\mathbf{F}$ function (which are approximately equivalent to $2^{128} \times \frac{1}{9} \approx 2^{124.84}$ 9-round SEED encryptions), and $2^{63} \times 2^{64} = 2^{127}$ memory accesses (which are equivalent to $\frac{2^{127}}{9} \approx 2^{123.84}$ 9-round SEED encryptions by our estimate given in Section 4.3). There are a total of about $2^{63} \times 2^{64} = 2^{127}$ possible combinations of (u, K_9, w), and thus on average there is about only one value of K_9 in every single entry (u, w) of the table.

We can have a slightly more efficient way. Recall that x is the 64-bit state immediately after the XOR operation with the round subkey K_9 in the $\mathbf{F}$ function. We generate the table as follows.

For every possible 64-bit value x such that $x < [x \oplus (\alpha, 0)]$:

- Compute the output difference of the $\mathbf{F}$ function corresponding to the pair of intermediate values $(x, x \oplus (\alpha, 0))$; and we denote it by y.
- For every possible value of the round subkey K_9, store K_9 into the table indexed by $(\min\{x \oplus K_9, x \oplus (\alpha, 0) \oplus K_9\}, y)$.

This latter precomputation also takes the same amount of memory of 2^{130} bytes, however, it has a time complexity of about $2^{63} \times 2 = 2^{64}$ computations of the $\mathbf{F}$ function (which is approximately equivalent to $2^{64} \times \frac{1}{9} \approx 2^{60.84}$ 9-round SEED encryptions), and about $2^{63} \times 2^{64} = 2^{127}$ memory accesses, slightly faster than the former precomputation.

Consequently, we can retrieve the candidate K_9 with a single lookup to entry $(\min\{C^L_{i,j}, \widehat{C}^L_{i,l}\}, (\widehat{X}, \beta) \oplus C^R_{i,j} \oplus \widehat{C}^R_{i,l})$ of this table in Step 3(c)(iii) of the online attack procedure given in Section 4.3. As a result, the resulting attack has a total time complexity of approximately $2^{125} + \frac{2^{60} \times 2^{64} \times (1+1)}{9} + 2^{60} \times 2^{64} \times 2 \times \frac{1}{9} + \frac{2^{60} \times 2^{64} \times (1+1)}{9} + 2^{60} \times 2^{64} \times \frac{2}{9} + 2^{60} \times 2^{64} + \frac{2^{127}}{9} \approx 2^{126.26}$ 9-round SEED encryptions when using 2^{60} pairs of plaintext structures, with the same success probability of 99.9%; and it has a total time complexity of approximately $2^{124} + \frac{2^{59} \times 2^{64} \times (1+1)}{9} + 2^{59} \times 2^{64} \times 2 \times \frac{1}{9} + \frac{2^{59} \times 2^{64} \times (1+1)}{9} + 2^{59} \times 2^{64} \times \frac{2}{9} + 2^{59} \times 2^{64} + \frac{2^{127}}{9} \approx 2^{125.51}$ 9-round SEED encryptions when using 2^{59} pairs of plaintext structures, with the same success probability of 97.8%. The first version is slightly faster than the corresponding version described in Section 4.3, and the second version is slightly slower than the corresponding version described in Section 4.3, but both require a dramatically larger memory.

4.4.3. Note 3

Yanami and Shimoyama's 7-round attack and Sung's 8-round attack used 2^{64} counters on the attacked last round subkey to filter out the candidate with the highest number as the correct subkey. Thus, we can follow Theorem 3 of Selçuk's work [14] to compute their approximate success probabilities, which are roughly 68.8% and 98.1%, respectively. Note that they can achieve a higher success probability simply by considering the counters with the highest few numbers.

5. Conclusions

SEED is a 128-bit block cipher with a 128-bit user key and a total of 16 rounds, which is an ISO international standard. In this letter, we have described some 7-round differentials that have a trivially larger probability than the previously known ones on SEED, and have presented a differential attack on 9-round SEED. The presented attack is theoretical, and it does not threaten the security of the full SEED cipher; but nevertheless, from a cryptanalytic view it suggests that the safety margin of SEED decreases below half of the number of rounds.

References

[1] E. Biham, A. Shamir, Differential cryptanalysis of DES-like cryptosystems, J. Cryptol. 4 (1) (1991) 3–72.
[2] E. Biham, A. Shamir, Differential cryptanalysis of the full 16-round DES, in: E.F. Brickell (Ed.), CRYPTO 1992, in: Lect. Notes Comput. Sci., vol. 740, Springer, Heidelberg, 1993, pp. 487–496.
[3] Internet Engineering Task Force (IETF), http://www.ietf.org.
[4] Internet Engineering Task Force (IETF), Use of the SEED encryption algorithm in cryptographic message syntax (CMS), RFC 4010, 2005.
[5] Internet Engineering Task Force (IETF), Addition of SEED cipher suites to transport layer security (TLS), RFC 4162, 2005.
[6] Internet Engineering Task Force (IETF), The SEED cipher algorithm and its use with IPSec, RFC 4196, 2005.
[7] Internet Engineering Task Force (IETF), The SEED cipher algorithm and its use with the secure real-time transport protocol (SRTP), RFC 5669, 2010.
[8] International Organization for Standardization (ISO), International Standard – ISO/IEC 18033-3, Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers, 2005.
[9] Korea Information Security Agency (KISA), SEED algorithm specification, archive available at www.seed.kisa.or.kr:8080/seed/down/SEED_Specification_english.pdf.
[10] J. Lu, W.-S. Yap, Y. Wei, Weak keys of the full MISTY1 block cipher for related-key differential cryptanalysis, in: E. Dawson (Ed.), CT-RSA 2013, in: Lect. Notes Comput. Sci., vol. 7779, Springer, Heidelberg, 2013, pp. 389–404.
[11] Mozilla Corporation, https://bugzilla.mozilla.org/show_bug.cgi?id=478839.
[12] National Bureau of Standards (NBS), U.S.A.: Data Encryption Standard (DES), FIPS-46, 1977.
[13] Public-Key Cryptography Standards (PKCS), PKCS #11 Mechanisms v2.30: Cryptoki – Draft 7, 2009.
[14] A.A. Selçuk, On probability of success in linear and differential cryptanalysis, J. Cryptol. 21 (1) (2008) 131–147.
[15] J. Sung, Differential cryptanalysis of eight-round SEED, Inf. Process. Lett. 111 (10) (2011) 474–478.
[16] Telecommunications Technology Association (TTA), South Korea, 128-bit block cipher SEED, TTAS.KO-12.0004, 1999 (in Korean).
[17] H. Yanami, T. Shimoyama, Differential cryptanalysis of a reduced-round SEED, in: S. Cimato, et al. (Eds.), SCN 2002, in: Lect. Notes Comput. Sci., vol. 2576, Springer, Heidelberg, 2003, pp. 186–198.
[18] S. Yoon, Using SEED cipher algorithm with SRTP, archive available at www.ietf.org/proceedings/69/slides/avt-11.pdf.