SEED Evaluation Report

CRYPTREC Evaluation Committee Chair Hideki Imai
Symmetric-Key Cryptography Subcommittee Chair Toshinobu Kaneko

February 28, 2002

1 Our Evaluation Methodology

This report presents the evaluation results of a block cipher SEED [1] from the two viewpoints: one is the security and the other is performance of the PC software.

In order to evaluate the security, two researchers for symmetric cryptography were assigned to assess the security level of SEED based on the submitted specification document and the self-evaluation document. The reviewers were allowed to refer to any published documents such as proceedings of a conference. Their assessment processes were completely independent to each other. Members of CRYPTREC symmetric-key cryptography subcommittee reviewed the assessment reports and summarized their point as this evaluation report.

As for performance evaluation, CRYPTREC secretariat, with the members of CRYPTREC committee, compiled the source program prepared by the submitter and measured the performance.

2 Evaluation Results

2.1 Executive Summary

Strength of SEED against differential cryptanalysis, linear cryptanalysis, and higher-order differential cryptanalysis is evaluated. It is also considered whether interpolation attack, non-surjective attack, and slide attack are applicable. Our analysis does not show any important flaws nor weaknesses in SEED. The best known attack at this point of time is an exhaustive search for the key.

The software performances of SEED on the PC environment is moderately slow among the 64-bit or 128-bit block ciphers evaluated by CRYPTREC [2].

During the evaluation process, it is found that the key-schedule of SEED has a peculiar property described later in this report. This property, however, does not seem to turn to an immediate threat to the security of SEED.

2.2 Security Evaluation Results

(1) Security of the Data Randomization Part

Differential cryptanalysis

As an approximation, exclusive OR operations are taken in place of modulo 2^{32} additions in the round function F. In this case, at least four S-boxes are active in every round and we can find a chracteristic in which at least two round functions are active in every 3 rounds. Since each S-box has the best characteristic probability 2^{-6}, the maximum differential characteristic probability of 13-round SEED is estimated as 2^{-192}. Here, it is assumed that 8 round functions are active in 13 rounds.

It seems difficult, if not impossible, to find the best characteristic probability of original SEED with modular addition operations. However, it seems highly unlikely that one can find differentials with only 3 active S-boxes per round for an r-round differential for large values of r. With a similar discussion to the above appoximated case, a differential over 13 rounds will have a probability of at most 2^{-144}.

In the self-evaluation report, the authors argue that the maximum differential characteristic probability for 6-round reduced SEED is estimated as 2^{-130}. Based on this value, it would be concluded that the differential cryptanalysis be applicable only to 6 rounds. Recently, in [3], a new maximum differential characteristic probability is found to be 2^{-124}. Using this characteristic, 7-round SEED is analyzed with 2^{126} chosen plaintext-ciphertext pair and 2^{126} computation of F function. This is the best differential attack against SEED ever reported.

Any of the above discussion shows a differential attack is unlikely to be found on SEED with full 16 rounds.

Linear cryptanalysis
Self-evaluation document reported that the best-known linear probability becomes less than 2^{-128} if 6 or more rounds are iterated. Though this value does not take multiple-path into account, this is an evidence that 16-round SEED has sufficient margin against linear cryptanalysis.

For a simplified version of SEED whose modulo 2^{32} addition is replaced by exclusive OR, the maximum linear characteristic is estimated to have a probability less than 2^{-192}. Since replacing back modular addition does not seem to increase the probability, it is unlikely to apply a linear attack to 16-round SEED.

Higher order differential attack
Round function F is composed of 32-bit functions G, which is constructed with S-boxes. The S-boxes have an algebraic degree of seven, which is the maximum degree for such a bijective S-box on 8 bits. All output bits of the function G have an algebraic degree of seven. Further, all output bits of the F function have an algebraic degree of 63, which is the maximum degree achievable since F function is a bijective on 64 bit. It is, therefore, very

likely that for 16-round SEED, the algebraic degree of the ciphertexts as function of plaintexts is high enough to prevent a higher order differential attack from being practical.

Interpolation attack
The S-boxes in SEED are constructed from the inverse function in a finite field, which has a simple description. However, the fact that both the inputs and outputs are mixed with affine mappings makes the description much more complex. This together with the mixed use of exclusive-ORs and modular additions make the interpolation attacks very unlikely to be applicable.

SQUARE attack (Integral attack)
It is conjectured that reduced versions of SEED with up to six rounds is vulnerable to these attacks, but not versions with more than six rounds.

Non-surjective attack
The non-surjective attacks are not applicable as the round function is bijective and thus not vulnerable to these attacks.

(2) Security of the Key-schedule Part
Exhaustive search
Exhaustive search for a key space is a trivial attack. This attack is not practical against the specified 128-bit key for SEED.

Slide attack
The slide attacks apply best to ciphers with very simple key-schedules. However, the key-schedule of SEED uses both the S-boxes and different round constants, which are good means to prevent these attacks from being effective.

New property of key-schedule
The key-schedule of SEED takes a 128-bit user-selected key and returns 16 pairs of round keys each of two times 32 bits, in total 1024 bits.

Let the round keys in round i be denoted $k_{i,0}$ and $k_{i,1}$. They are generated as follows. The user-selected key is divided into four pieces, a, b, c, d, each of 32 bits.

- for i:=1 to 16 do
  - $k_{i,0} = G(a + c - kc_i)$

– $k_{i,1} = G(b - d + kc_i)$
– if i odd do $b||a = (b||a)^{>>8}$
else do $d||c = (d||c)^{<<8}$,

where kc_i for $i = 1, \ldots, 16$ are round constants derived in a pseudo random fashion. We refer to [1] for the notation used.

At a first glance it appears that there are no related keys for SEED because of the use of the highly nonlinear function G in the key-schedule. However, there are keys whose subkeys are related. Notice that the generation of $k_{i,0}$ depends only on the rotated versions of a and c together with the round constant. Thus if it holds for two user-selected keys K and K^* that $a + c$ is always a constant, then the subkeys $k_{i,0}$ for K and the subkeys $k^*_{i,0}$ for K^* are equal, i.e., $k_{i,0} = k^*_{i,0}$ for $i = 1, \ldots, 16$. If $a+c$ is to be constant for all values of i, then it must hold that $(b||a) = (b||a)^{<<8} = (d||c) = (d||c)^{<<8}$, which means that if $a = a_0, a_1, a_2, a_3$, where a_i are byte valued, and similar for b, c, and d, then it must hold that for some constant e that

- $a_i + c_j = e$ for all $i, j = 0, 1, 2, 3$,
- $a_i + d_j = e$ for all $i, j = 0, 1, 2, 3$,
- $b_i + c_j = e$ for all $i, j = 0, 1, 2, 3$,
- $b_i + d_j = e$ for all $i, j = 0, 1, 2, 3$,

Summing up, let the user-selected key be divided into the 32-bit words a, b, c, d. Consider the keys obtained from having $a_0 = a_1 = a_2 = a_3 = x$ for some value x, $b = a$, $c_0 = c_1 = c_2 = c_3 = y$, for some value y, and $d = c$. Then it holds that the keys for which $x + y$ is a constant will produce the same values for $k_{i,0}$ for $i = 1, \ldots, 16$. Thus, there are 2^{16} keys which can be divided into 256 classes of each 256 keys, such that in one class all 256 keys produce identical values of the subkeys $k_{i,0}$ for $i = 1, \ldots, 16$. Table 1 lists three keys and the subkeys they generate.

It follows by very similar observations that there are 2^{16} keys which can be divided into 256 classes of each 256 keys, such that in one class all 256 keys produce identical values of the subkeys $k_{i,1}$ for $i = 1, \ldots, 16$. Table 2 lists three (other) keys and the subkeys they generate.

Despite of these findings the key-schedule of SEED does not seem to allow for related-key attacks. First of all, the "related keys" reported above are few, second, the relations between them does not seem to be strong enough to allow for these kinds of attacks.

Key =	9b9b9b9b	9b9b9b9b	11111111	11111111
Round key no.				
1	4124db1d	3451bd29		
2	9a0f9a3a	4b127456		
3	79efee8e	273d39c9		
4	57215006	b12689b3		
5	03c24bbc	5f7092c7		
6	c0a53c4c	2b831b79		
7	cf3ebb62	d29fac9a		
8	2a14ef6c	a2c6cfe2		
9	7b85aa09	07894284		
10	f527f311	9100f2f9		
11	4ee60e85	14546a91		
12	26d5c935	864101db		
13	803e5e92	34e0e2c0		
14	c91d482b	2b10ede5		
15	0788fd30	2d60d71e		
16	f92d78ce	2bd7ef41		
Key =	3a3a3a3a	3a3a3a3a	72727272	72727272
Round key no.				
1	4124db1d	e0ef1874		
2	9a0f9a3a	711b066c		
3	79efee8e	5c178ff9		
4	57215006	0b809197		
5	03c24bbc	26afe9b0		
6	c0a53c4c	3c1b8a18		
7	cf3ebb62	573ddbb6		
8	2a14ef6c	c0be0d10		
9	7b85aa09	75080ba7		
10	f527f311	56ab375e		
11	4ee60e85	39e99972		
12	26d5c935	1591baad		
13	803e5e92	0ffc828b		
14	c91d482b	2d9680fc		
15	0788fd30	8e5a5bd0		
16	f92d78ce	5e235141		
Key =	2a2a2a2a	2a2a2a2a	82828282	82828282
Round key no.				
1	4124db1d	07460ff4		
2	9a0f9a3a	b82298f4		
3	79efee8e	7ee3b13e		
4	57215006	46c3d6b0		
5	03c24bbc	4af65578		
6	c0a53c4c	1bb446d4		
7	cf3ebb62	0b5a1d9e		
8	2a14ef6c	bfaa5324		
9	7b85aa09	4c16e012		
10	f527f311	1d68f56f		
11	4ee60e85	7def7131		
12	26d5c935	52eff20b		
13	803e5e92	3c3c924e		
14	c91d482b	e02f858f		
15	0788fd30	74fd6be4		
16	f92d78ce	a9ccd586		

Table 1: Examples of keys which produce equal first subkeys in every round.

Key =	9b9b9b9b	9b9b9b9b	efefefef	efefefef
Round key no.				
1	68c9edf1	7d28cbaf		
2	7e8e4d27	f9c76fad		
3	aa37e9ee	f59dd258		
4	5da694ad	7605924a		
5	c61b186a	b3c83014		
6	45dc4ae5	bfb0fcbe		
7	05ce5df3	fd6a1882		
8	9ab323b3	6ef967c7		
9	a08e3ccc	d883dcd7		
10	8c92b184	13ddd10c		
11	77553f19	af7cecc4		
12	24e69b24	e007b43e		
13	bca52806	5f7651a0		
14	dd2474e9	1e09a2f2		
15	0eeecd5b	9c28a623		
16	3685e91e	bcad5740		

Key =	3a3a3a3a	3a3a3a3a	8e8e8e8e	8e8e8e8e
Round key no.				
1	0d92c044	7d28cbaf		
2	de60205a	f9c76fad		
3	d9258549	f59dd258		
4	9d84df1f	7605924a		
5	ea0a79a8	b3c83014		
6	638fd5fa	bfb0fcbe		
7	470a077c	fd6a1882		
8	c252c5d8	6ef967c7		
9	a6b5f762	d883dcd7		
10	a55b43b7	13ddd10c		
11	ca3a056e	af7cecc4		
12	a678af9c	e007b43e		
13	4aa21758	5f7651a0		
14	a0ab171a	1e09a2f2		
15	1d432710	9c28a623		
16	ad80bb01	bcad5740		

Key =	2a2a2a2a	2a2a2a2a	7e7e7e7e	7e7e7e7e
Round key no.				
1	f23c7655	7d28cbaf		
2	0b5f9dbd	f9c76fad		
3	656eb6da	f59dd258		
4	886b8015	7605924a		
5	caac1ba9	b3c83014		
6	54d62348	bfb0fcbe		
7	a9bdeb44	fd6a1882		
8	8bb07ddf	6ef967c7		
9	661831f3	d883dcd7		
10	05090fea	13ddd10c		
11	60f094cc	af7cecc4		
12	3393a0f5	e007b43e		
13	770ab190	5f7651a0		
14	10702afd	1e09a2f2		
15	0ef8e298	9c28a623		
16	7c8e917d	bcad5740		

Table 2: Examples of keys which produce equal second subkeys in every round.

The above phenomena are not a threat for SEED when used for encryption. If a key is chosen uniformly at random, the probability to pick one of the above keys is very small. Moreover it is not clear for which applications an attacker would be able to exploit the use of such keys. On the other hand, such similarities between the subkeys of different keys do not appear in other modern block ciphers and could probably have been prevented.

(3) Comments on fixed points of the S-boxes

The designers of SEED argued in [1] that one reason for modifying the outputs of the power polynomials x^{247} and x^{251} by affine mappings was to remove the fixed points 0 and 1. However, although these affine mappings remove these two fixed points, it has also the effect that it introduces three other fixed pints, namely 23, 230, for S-box S_1 and 28 for S-box S_2. Design policy and the design result do not seem to match well with respect to fixed points, though it is unlikely that these new fixed points give rise to new threats.

2.3 Software Evaluation Results

Tables 3 and 4 show the our measurement data. We used a PC based on Pentium III (650MHz). We used the following APIs.

```
typedef unsigned char uint8;
void KSEnc(uint8 *secret, uint8 *eKey); /* key schedule for Enc() */
void KSDec(uint8 *secret, uint8 *eKey); /* key schedule for Dec() */
void Enc(uint8 *eKey, uint8 *src, uint8 *dst); /* encryption */
void Dec(uint8 *eKey, uint8 *src, uint8 *dst); /* decryption */
void Kine(uint8 *secret,uint8 *src,uint8 *dst); /* KSEnc()+Enc() */
void Kind(uint8 *secret,uint8 *src,uint8 *dst); /* KSDec()+Dec() */
```

Using the above functions (`Enc()`, `Dec()`, `Kine()`, `Kind()`), we measured 2^{17} iteration time of these functions. The output (ciphertext) was used to the next input (plaintext). We tried the above procedure 256 times, and adopted the best values, and converted them to cycles per block. Note that we show the program size as the amount of object sizes. The measurement method mentioned above is the same used in [2].

Because the measurement value is significantly affected by the execution environment, the value might not always have been realized. Errors also were caused by errors in the measurement programs. Therefore, making a final determination solely based on the values in the tables is dangerous.

Table 3: Data-randomization part speed-measurement results ([cycles/block])

Pentium III (650MHz)		
Language	C	
Program size	45056 bytes (including encryption/decryption/key schedule)	
Compiler option	VC++6.0 Win32 Release (Default)	
	Encryption (Best/Average)	Decryption (Best/Average)
First round	846 / 871	846 / 873
Second round	846 / 867	846 / 867
Third round	846 / 866	846 / 867

Table 4: Key schedule + data-randomization part speed-measurement results ([cycles/block])

Pentium III (650MHz)		
Language	C	
Program size	49152 bytes (including encryption/decryption/key schedule)	
Compiler option	VC++6.0 Win32 Release (Default)	
	Encryption (Best/Average)	Decryption (Best/Average)
First round	1233 / 1255	1235 / 1256
Second round	1233 / 1255	1235 / 1257
Third round	1233 / 1255	1235 / 1257

References

[1] Korean National Body, "Contribution for Korean Candidates of Encryption Algorithm (SEED)," related to ISO/IEC JTC1 SC27 N2563, 2000. http://www.kisa.or.kr/seed/data/algorithm/seed_english.doc.

[2] Information-technology Promotion Agency, Japan, "CRYPTREC Report 2000," March, 2001. http://www.ipa.go.jp/security/enc/workshop2000/report-e.html

[3] H.Yanami and T.Shimoyama, "Differential Cryptanalysis of SEED," in Japanese, Proc. of the 2002 Symposium on Cryptography and Information Security (SCIS2002), Jan.29-Feb.1, 2002.