Computer Science Journal of Moldova, vol.**10**, no.2(29), 2002

Fault-based analysis of flexible ciphers

V.I.Korjik, A.Mukherjii, M.A.Eremeev, N.A.Moldovyan

Abstract

We consider security of some flexible ciphers against differential fault analysis (DFA). We present a description of the fault-based attack on two kinds of the flexible ciphers. The first kind is represented by the fast software-oriented cipher based on data-dependent subkey selection (DDSS), in which flexibility corresponds to the use of key-dependent operations. The second kind is represented by a DES-like cryptosystem GOST with secrete S-boxes. In general, the use of some secrete operations and procedures contributes to the security of the cryptosystem, however degree of this contribution depends significantly on the structure of the encryption mechanism. It is shown how to attack the DDSS-based flexible cipher using DFA though this cipher is secure against standard variants of the differential and linear cryptanalysis. We also give an outline of ciphers RC5 and GOST showing that they are also insecure against DFA-based attack. We suggest also a modification of the DDSS mechanism and a variant of the advanced DDSS-based flexible cipher that is secure against attacks based on random hardware faults.

KEY WORDS: Flexible cipher, block cipher, differential fault analysis.

1 Introduction

There are a lot of block ciphers known as strong enough against brute force attack by total exhaustion of keys and against more sophisticated attacks. First of all these are DES, IDEA, RC5 and GOST [1]. Recently a number of fast software encryption algorithms have been proposed,

for example, Blowfish [1] and CIKS-1 [2]. An important design criterion for an encryption algorithm is its performance. Designers search for round functions that allow to reduce the number of rounds and to simplify a block processing on each round. The cryptosystem RC5 can be seen as attempt along this line. The main idea is to produce the key scheduling which expands the initial secret key to a long enough one. In ciphers with extended key it is attractive to use data-dependent subkey selection (DDSS) as basic cryptographic primitive. In particular it is attractive to combine DDSS with key-dependent operations. Several variants of the DDSS-based ciphers are presented in [3] and it is shown that they are secure against known plaintext attack and against chosen plaintext attack. A new crucial type of attack based on differential fault analysis (DFA) has been proposed by D.Boneh et. al. to break RSA [4] and by A.Shamir and E.Biham to break symmetric ciphers [5]. This type of analysis can be applied to the case when an adversary has a physical access to the encryption devices to induce faults.

In section 3 we consider fault-based cryptanalysis of the DDSS-based cipher called DDSS-1. We show that it can be broken if a malicious adversary may cause and collect hardware faults corresponding to the different stages of the encryption. In contrast to fault-based cryptanalysis of DES we need many pairs of plaintexts/ciphertexts to break DDSS-1 but it requires significantly less operations than the total exhaustion of 256-bit key.

In Section 4 we consider briefly the block ciphers RC5 with data-dependent rotation operations and GOST with secrete S-boxes. It is shown that they are also insecure against DFA-based attack.

In Section 5 we consider advanced version of the DDSS-based block ciphers with key-dependent operations. Some peculiarities of the modified DDSS mechanism provide security against DFA.

In section 6 we summarize the main results and propose some modification of DDSS-1 to be more secure against DFA-based cryptanalysis.

2 Description of DDSS-1

DDSS-1 is a R-round iterated cipher with block size of 64 bits and secret key size of 256 bits. First, the secret key is expanded into 256 subblocks providing that each of them consists of 32 bits according to a scheduling scheme (see [3] for details). Since this scheme is very complicated it is reasonable to consider the expanded key as the real key in the attack.

Fig. 1 shows the structure of one round of DDSS-1. Here "$> c_i <$" denotes right circularly shift by number of digits c_i depending on i. "$*_j$" is one of the three algebraic operations: XOR, addition modulo 2^{32} or subtraction modulo 2^{32} depending on the secret key. The abbreviation SS stands for the function "Selection of subblock". The output of this function is the 32-bit expanded key subblock Q_j which number j coincides with the input of this function. The operations "$> c_i <$" and "$*_j$" depend not only on the indices i and j but on the secret key as well. This is the reason to define this cipher as an *operation key dependent* one. We can also see that a selection of the expanded key subblock is determined not deterministically (by the number of step) but by intermediate states of transformed data blocks (X^i and Y^j) depending on the initial plaintext P. In figure 1 a single line denotes a transmission of a subblock with the size of 8 bits whereas a double line means a transmission of 32 bits except the single lines close to operations "$> c_{10} <$" and "$> c_{11} <$", where they correspond to a transmission of subblocks with the total size of 16 bits.

To decrypt the ciphertext C given the secret key we have to produce all steps in reverse order changing all the operations to reverse ones. (It is easy to verify (see Fig. 1) that the pass of this scheme from the bottom to the top can recover a plaintext P from a ciphertext C given the secret key which uniquely determines the expanded key subblocks and all operations.) In case of multi-round ($R > 1$) algorithm we repeat one round scheme R times , where the operations depend in addition on the number of the current round.

3 Fault-based attack on DDSS-1

Let us consider a case when an adversary can induce faults. For example a bit stored in a register might randomly flip or a certain gate may spontaneously produce an incorrect value. We assume that the probability of such fault is small enough so that as a rule a single error takes place in different intermediate subblocks X^i, Y^j of the encryption scheme. It is very hard (rather impossibly) to induce a fault in some prescribed block but we can control our scheme and expect a fortunate case to be so. This approach makes us to repeat such attempts many times but in any case it requires much less computations than for brute force attack.

At first, we encrypt some given plaintext P_1 and obtain the ciphertext C_{10}. Then we induce fault and encrypt the same word P_1 many times to pick up errors in the subblock x_4^7 only. As an indicator to be so we can take the situation when subblocks (x_3^7, x_2^7, x_1^7) are error free and at the same time subblock x_4^7 contains errors. It is easy to perform because all these subblocks form the ciphertext. Let us denote by $C_{11}, C_{12}, ..., C_{1n}$ the ciphertext blocks corresponding to the first plaintext P_1 providing that there are errors in subblock x_4^7 for each of these ciphertexts and only in such subblock of X^7. Then we can compute the differences of the expanded key subblocks

$$\delta_{x_4^7 \tilde{x}_4^7} = Q_{x_4^7} \,\overline{*}_8\, Q_{\tilde{x}_4^7}, \tag{1}$$

where x_4^7 is a subblock without error and $\tilde{x}_4^7$ are subblocks containing errors (subblocks $\tilde{x}_4^7$ are parts of the erroneous ciphertexts $C_{11}, C_{12}, ..., C_{1n}$, whereas x_4^7 is a part of ciphertext C_{10}) and $\overline{*}_8$ is a group operation inverse to $*_8$. In line with our strategy all subblocks x_4^7 and $\tilde{x}_4^7$ correspond to the same plaintext P_1.

To simplify the further outline let us denote the differences by δ_{ij}, where $i = x_4^7$ and $j = \tilde{x}_4^7$. If i is fixed, then the total number of these differences is equal to $n = 255$. There is no point in expectation of all possible j given i, since we let a very small probability of fault (otherwise it would be incredible to obtain no errors in subblocks (x_3^7, x_2^7, x_1^7) of ciphertext).

Because of this we will take other plaintexts $P_2, P_3, ..., P_n$ to observe corresponding ciphertexts $C_{20}, C_{30}, ..., C_{n0}$. It is reasonable to suggest that our encryption algorithm is a "good" hash function which maps a plaintext to subblock x_4^7 of ciphertext. Hence we should take $256(1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{256}) \approx 1618$ plaintexts on the average to obtain all 256 possible subblocks x_4^7 of ciphertext. Now plaintexts $P_1, P_2, ..., P_{256}$ can be considered as blocks corresponding to different subblocks x_4^7 without the loss of generality. Then we induce fault and encrypt n times each of plaintexts $P_1, P_2, ..., P_{256}$ to obtain a batch of corresponding ciphertexts $C_{l1}, C_{l2}, ..., C_{ln}$, $l = 2, 3, ..., 256$ providing that there are errors in subblocks x_4^7 for each of these ciphertexts and only in such subblocks of X^7.

In the same manner as before we can compute the differences of the expanded key subblocks $Q_{x_4^7} \overline{*}_8 Q_{\tilde{x}_4^7}$, where x_4^7 are error free subblocks corresponding to $P_2, P_3, ..., P_{256}$ and $\tilde{x}_4^7$ are subblocks containing errors. Unfortunately this set of differences can be insufficient to obtain a full set of 256 differences δ_{ij} for the same i. To solve this problem we can use a group property of any operation "$*_8$" used in encryption algorithm. In fact, the following equality will be true

$$\delta_{ij} = \delta_{ik} *_8 \delta_{kj}. \tag{2}$$

It is easy to see that the use of relation (2) in combine with full set of subblocks x_4^7 allow us to find some i_0 which has a full set of differences $\delta_{i_0 j}$, $j = 0, 1, 2, ..., 255$.

This leads to an opportunity to recover all the subblocks of the expanded key if we guess the key subblock Q_{i_0}. There are 2^{32} such subblocks only and we could try all of them in a reasonable time. But we do not know yet which operations "$*_1$"..."$*_8$" and "$> c_1 >$" ..."$> c_{12} >$" were used in encryption algorithm because they are known as some deterministic function of the secret key but not of the expanded one. To solve this problem we remark that the following generic property of operations "$*_8$" and "$> c_{12} >$" is true: if we guess these operations correctly, then the differences δ_{ij} do not depend on the plaintext but depend on (i, j) only. In the case of a false guess this property can be true with a small enough probability. Then we can find a pair of different

plaintexts corresponding to the same x_4^7 (may be it has taken several attempts to encrypt different plaintexts) and induce faults to obtain at least two identical subblocks $\tilde{x}_4^7$ corresponding to these plaintexts. When corresponding differences δ_{ij} coincide it shows that operations "$*_8$" and "$> c_{12} >$" have been taken correctly with significant probability. Since we have only 93 possible combinations of operations "$*_8$" and "$> c_{12} >$", this test does not require much time.

After recognition of the operations "$*_8$" and "$> c_{12} >$" we can pass on to the selection of the correct expanded key. Let us take one of 2^{32} possible expanded key subblocks Q_{i_0}. It results at once in a knowledge of the whole expanded key $(Q_0, Q_1, ..., Q_{255})$ (may be incorrectly) because we already had the complete set of differences $\delta_{i_0 j}$, $j = 0, 1, 2, ..., 255$. Since expanded key and operations "$*_8$" and "$> c_{12} >$" are known, we can compute Y^4 (also the subblock y_4^4) for any given plaintext P and consequently compute the expanded key subblock $Q_{y_4^4}$. Then we induce a fault and recompute $\tilde{Y}^4$ for the observed modified ciphertext $(\tilde{X}^7|\tilde{Y}^6)$. Single errors in block $\tilde{Y}^4$ are more likely to occur than multiple ones if our guess about subblock Q_{i_0} is correct. Otherwise we will have multiple errors rather than single ones. This is a criterion to select the correct expanded key.

With the knowledge of the correct expanded key we can pass on to the specification of the operation "$*_7$". To perform it let us induce a fault in y_4^4 given error free subblocks (y_3^4, y_2^4, y_1^4) (we select such event because the block Y^4 can be recomputed given block Y^6 which is a part of the ciphertext). If we guess the operation "$*_7$" correctly then the input X^6 recomputed for known X^7 and $Q_{y_4^4}$ remains the same for both the absence of errors and the presence of them given the same plaintext P. This is a criterion to recognize the real operation "$*_7$". Similarly we can specify the operations "$*_6$", "$*_5$" and so on up to the top of the scheme shown in Fig. 1. Remark that we do not have to induce more errors when the all operations are determined. In fact one can store the ciphertexts obtained in the stage of the expanded key computation and use them for selection of faults which took place at previous encryptions.

If the encryption algorithm has several rounds we should begin with

the bottom of the last round and go to the beginning of the algorithm passing all rounds one by one. It is easy to see that the most time consuming stage of this fault-based attack is the procedure of selection one of 2^{32} expanded key subblock Q_{i_0}. It requires $\sim 2^{32} \cdot l_0$ operations, where l_0 is the number of attempts to be necessary on the average to obtain errors in the prescribed subblock (register) of the encryption scheme. The complexity of the proposed attack seems to be quite significant but much less than the complexity of a brute force attack which requires the total exhaustion of 2^{256} secret keys.

4 Fault-based attack on RC5 and GOST

4.1 RC5 against DFA

Cipher RC5 includes procedures expressed by the following pseudocode

$$A := A + S_0 \pmod{2^n}, \tag{3}$$

$$B := B + S_1 \pmod{2^n}, \tag{4}$$

for $i = 1$ **to** R **do:**

$$A := (A \oplus B)^{<B<} + S_{2i} \pmod{2^n}, \tag{5}$$

$$B := [(B \oplus A)^{<A<}] + S_{2i+1} \pmod{2^n}, \tag{6}$$

where A and B are the left and the right half of the input message, respectively (for details see [7]).

Let us assume the attacker can induce one random fault on the average while encrypting one data block. After a number of attempts one can induce a fault in given register at given step of data transformation, for example in the register containing data subblock A after execution of the transformation according to formula (5) in the round $i = R$. Such faults are easily recognized observing ciphertexts produced from the same plaintext block T without faults ($C = A|B$) and with faults ($\tilde{C} = \tilde{A}|\tilde{B}$).

From (6) it is easy to derive the expression

$$A \oplus \tilde{A} = (\tilde{B} - S_{2R+1})^{>\tilde{A}>} \oplus (B - S_{2R+1})^{>A>} \pmod{2^n}. \quad (7)$$

In (7) only S_{2R+1} is unknown. With high probability one can obtain $A \bmod 2^5 = \tilde{A} \bmod 2^5$. In this case equation (7) is transformed to

$$(A \oplus \tilde{A})^{<A<} = (\tilde{B} - S_{2R+1}) \oplus (B - S_{2R+1}) \pmod{2^n}. \quad (8)$$

Using last equation it is easy to calculate a part of subkey S_{2R+1}. Inducing different faults one can determine the value of subkey S_{2R+1}. After the subkey S_{2R+1} is found one can compute the subkey S_{2R}. To do this we can use faults which occurred in data subblock B in the round $i = R - 1$ after the transformation given by formula (6). Using subkey S_{2R+1}, such an event can be easily recognized by recovering the value of subblock B after $(R-1)$-th round. After several attempts one can determine the subkey S_{2R-1}. In a similar way it is easy to determine S_{2R-2}, ...,S_0. For $R = 10 - 30$ the work effort to compute the all subkeys does not exceed 10^8 operations. Model of this attack against RC5 has been elaborated and tested experimentally. A simulation of given above attack confirmed our estimation of the complexity of RC5 against DFA.

4.2 GOST against DFA

Russian encryption standard GOST [1] is an example of the practically used ciphers security of which is based on the both the secrete key and the secrete S-boxes. GOST is a DES-like 32-round cryptosystem with the 256-bit secrete key and eight secrete S-boxes having 4x4 size. While attacking one round of GOST with DFA technique one should calculate one 32-bit round subkey and to determine the secrete set of eight S-boxes from $(16!)^8$ possible sets. We have applied DFA based on random hardware faults to this cipher. Our results have confirmed the possibility to reconstruct the complete specification of the DES-like unknown ciphers [5]. The weaknesses of the GOST encryption round consist in (1) the use of the modulo 2^{32} addition operation between the

left data subblock and the round subkey and (2) small size of S-boxes. Our variant of the DFA attack against GOST is based on the use of the avalanche caused by carry bits. To perform such attack it is sufficient to have two output texts (one text with errors and one free of them) resulted from the same source text with the size about 10^5 byte. The work effort of this attack is $< 10^{12}$ operations. The model of the DFA-based attack on GOST has been elaborated. This model has confirmed our estimation of the security of GOST against DFA based on random faults.

5 Cipher based on the advanced DDSS

Below we present another fast software-oriented cipher from the family of the DDSS-based cryptosystems. It has encryption speed about 70 Mbit/s for PentiumII-266. It uses 2051-byte expanded key presented as a sequence $\{q_j\}$, $j = 0, 1, 2, ..., 2051$, one-byte words. This expanded key is formed following a scheduling scheme from the secret key of 256 bit. We believe that this scheme is also very complicated and hence it is reasonable to consider the expanded key as the real key in the attack.

A plaintext has to be divided into blocks consisting of 128 subblocks and each of them is a 32-bit subblock. These 32-bit subblocks are transformed one by one into the blocks of ciphertext using different key-dependent operations and "accumulating" variables V, U, and Y obtained as a function of both the expanded key and the plaintext. A typical number of rounds is 4. An algorithm suitable to the use as disk encryption system (about problem of the disk sector encryption see [8]) is presented below, where $* \in \{\oplus, + \pmod{2^{32}}, - \pmod{2^{32}}\}$ and "$> c <$" denotes right circularly shift by c digits in line with notations for previous cipher.

Disk sector encryption algorithm

INPUT: 512-byte plaintext is represented as a sequence of 32-bit words P_h, $h = 0, 1, 2, ..., 127$.

1. Set $r = 1$, $R = 4$ and define: $L_h = P_h$, $h = 0, 1, ..., 127$.

2. Set counter $i = 1$ and compute initial values of the variables U, V, Y, G, and n: $U \leftarrow Q(1)\ \ , V \leftarrow Q(2)$, $Y \leftarrow Q(3)$, $G \leftarrow Q(4)\ \ ,$ $n \leftarrow Q(5)\ mod\ 2^{11}$, where expanded key subblocks $Q(i)$ are taken according to the formula $Q(i) = q_{i+3}|q_{i+2}|q_{i+1}|q_i$.

3. Perform computations:

$$n \leftarrow \{[n \oplus (G \bmod 2^{11})] - U\} \bmod 2^{11},$$

$$V \leftarrow (V + G)^{>11>}\ *_{7r-6}\ Q(n),$$

$$n \leftarrow n\ *_{7r-5}\ (V \bmod 2^{11}),$$

$$U \leftarrow [U\ *_{7r-4}\ Q(n)]^{>c_{3r-2}>} - (G^{>22>}),$$

$$n \leftarrow n + U \bmod 2^{11},$$

$$Y \leftarrow Y\ *_{7r-3}\ Q(n) + G \bmod 2^{32}.$$

4. Compute the index of the current input word: $h = 128 - i$, if $r = 2, 4$, or $h = i - 1$, if $r = 1, 3$.

5. Perform the current encryption step:

$$C_h \leftarrow [(L_h\ *_{7r-2}\ V)^{<c_{3r-1}<}\ *_{7r-6}\ Y]^{>c_{3r}>}\ *_{7r}\ U.$$

6. Save the value C_h.

7. If $i < 128$, then increase i and go to step 3.

8. If $r < R$, then increase r, update $L_h \leftarrow C_h$, $h = 0, 1, ..., 127$, and go to the step 2.

Otherwise – STOP.

OUTPUT: 512-byte ciphertext $\{C_h\}$, $h = 0, 1, 2, ..., 127$.

We have no intention to describe a fault-based attack on this kind of cipher in details. We note only that the cipher described above has the following features which probably make it secure against this attack:

1. The key indices are never presented as some observed binary blocks.

2. The expanded key subblocks are not used directly as inputs of some operations. They are saved only to form the intermediate "virtual" keys.

3. This algorithm prevents a possibility of separate consideration for different possible combinations of operations used in the same encryption round.

These properties result in the implication that it is very hard to compute the differences of the expanded key subblocks. Indeed, in order to change the number of subkey at some DDSS step of the last round we should induce an error into n, however this changing cannot be recognized exactly given ciphertext only.

6 Conclusion

We have described an application of the DFA attack based on random faults to break the cryptosystems DDSS-1, RC5, and GOST. These ciphers can be modified to be more resistant to such an attack. For example we can build into the algorithm DDSS-1 a key-dependent 32x32 substitutions just after the operation "$> c_{12} >$" and also before the operations "$> c_4 >$", "$> c_3 >$", "$> c_2 >$", and "$> c_1 >$" (because an attacker may also induce fault into initial blocks and use the known decryption algorithm to break this cryptosystem). The availability of key-dependent substitutions will make a computation of differences δ_{ij} very hard because the number of possible operations (including substitutions) is very large. On the other hand such modification will result in a decreased speed of encryption. (That is why substitutions are not used in DDSS-1.)

We have also presented another version of the fast DDSS-based ciphers. In the proposed cipher selected subkeys are not combined directly with transformed data subblocks. Some sets of subkeys are mixed together forming some accumulating key variables combined with data. Such advanced DDSS mechanism contributes significantly against DFA attack based on random hardware faults.

Comparing results of four different ciphers against DFA one can conclude that the use of secrete algorithm is not sufficient to provide high security against DFA. It is more important to design the round structure contributing efficiently to the security against DFA. Secrecy of the algorithm is effective against DFA only if the general encryp-

tion scheme is properly composed. For example, one can use advanced DDSS-based cryptoschemes to develop flexible ciphers like that described in section 5.

References

[1] B.Schneier *Applied Cryptography, Second Edition*, John Wiley & Sons, Inc., 1996.

[2] A.A. Moldovyan, N.A. Moldovyan, *A cipher based on data-dependent permutations*, Journal of Cryptology. 2002, vol. 15, no. 1, pp.61–72.

[3] A.A.Moldovyan, N.A.Moldovyan, *Software Encryption Algorithms for Transparent Protection Technology*, Cryptologia. 1998, V.XXII, N.1, pp.56–68.

[4] D.Boneh, R.A.DeMillo, R.J.Lipton, *On the Importance of Checking Cryptographic Protocols for Faults*, Advances in Cryptology – EUROCRYPT '97 Proceedings, Springer-Verlag LNCS. 1997, pp. 37–51.

[5] E.Biham, A.Shamir, *Differential Fault Analysis of Secret Key Cryptosystems*, 17th Annual International Conference "Advances in Cryptology – CRYPTO'97". Santa Barbara, USA, August 17-21, 1997. Procedings. Springer-Verlag LNCS. 1997. V. 1294. pp. 513–525

[6] A.A.Moldovyan, N.A.Moldovyan, *Flexible Block Ciphers with Provably Inequivalent Cryptalgorithm Modifications*, Cryptologia. 1998. V.XXII. N.2. pp.134–139.

[7] R.L.Rivest, *The RC5 Encryption Algorithm*, "Fast Software Encryption", Second International Workshop Proc. Springer-Verlag LNCS. 1995. V. 1008. pp.86–96.

[8] Crowley P. Mercy, *A Fast Large Block Cipher for Disk Sector Encryption*, "Fast Software Encryption", Seventh International Workshop Proc. Springer-Verlag LNCS. 2001, V. 1978. pp.49–63.

V.I.Korjik, A.Mukherjii, M.A.Eremeev, N.A.Moldovyan, Received April 4, 2002

Korjik V.I.
CINVESTAV-IPN,Ingeneria Electrica Department,
AV.IPN No.2508 ESQ Ticoman Col. San Pedro Zacatenco
C.P.07000, Mexico, D.F., Mexico,
Tel:5747-7000, ext.3459,
Fax: 5747-7088,
E–mail: *vkorjik@mail.cinvestav.mx*

Mukherjii A.
Itel Technologies Limited,
UK, London SW11 3AD, 2 Old Garden House,
The Lanterns, Bridge Lane,
tel. (+44) 20 77878005, fax (+44) 20 77878007,
e–mail: *anindo@london.com*

Eremeev M.A., Moldovyan N.A.
Specialized Center of Program Systems SPECTR,
Kantemirovskaya Str., 10,
St-Petersburg 197342, Russia;
Tel:+7-812-2453693,
Fax:+7-812-2453743,
E–mail: *spectr@vicom.ru*

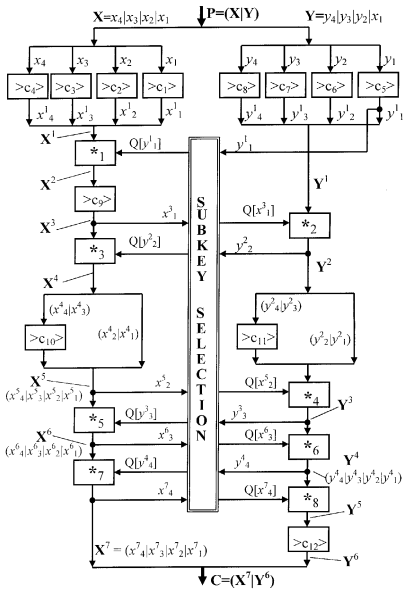

Figure 1One round of DDSS-1