Krzysztof BUCHOLC
POZNAŃ UNIVERSITY OF TECHNOLOGY
pl. Skłodowskiej-Curie 5, 60-965 Poznań

Influence of faults insertion on software implementation of block cipher security

Ph.D. eng. Krzysztof BUCHOLC

Is a senior lecturer in Poznań University of Technology. His main research areas are: computer architecture, embedded systems, reliability and diagnosis of computer hardware. He received a Ph.D. from Poznań University of Technology in 1989. He is author or coauthor of more than 50 published papers, 2 patents and 1 textbook.

e-mail: Krzysztof.Bucholc@.put.poznan.pl

Abstract

Faults inserted deliberately may cause hazard for cipher security. In this paper there is analysed the influence of faults on cipher security. There are considered two approaches to obtain more tamper-resistant implementation. The author focuses on the PP-1 block cipher but the results are applicable to any class of ciphers in which the main key, or all the round keys, can be deduced from the round key for the first round, and the round key is applied twice in each round.

Keywords: block cipher, software implementation, fault attack.

Wpływ defektów wstawianych celowo na bezpieczeństwo programowej implementacji szyfru blokowego

Streszczenie

W pracy przeanalizowano wpływ defektów wstawianych celowo na bezpieczeństwo szyfru blokowego. Przedmiotem badań była programowa implementacja szyfru blokowego, przy czym procesor wraz z oprogramowaniem potraktowano, jako układ sprzętowy. Rozważono wpływ defektów typu sklejenie z zerem, sklejenie z jedynką i zmiana bitu na przeciwny na bezpieczeństwo szyfru. Uzyskane wyniki są ważne dla takich szyfrów blokowych, w których w każdej rundzie używane są dwa klucze rundowe. Koncepcja ataku polega na celowym wprowadzeniu błędów w procesie szyfrowania. Otrzymane błędne wyniki są następnie przetwarzane w celu znalezienia potencjalnych kluczy rundowych. W błędnych wynikach poszukujemy takich, które odpowiadają jednorundowej wersji szyfru. Kolejnym etapem jest weryfikacja wyznaczonych kluczy. W pracy pokazano, że w przypadku uzyskania błędnych wyników odpowiadających rundzie numer 1 dla 2 różnych bloków danych, możliwe jest złamanie szyfru w wyniku wypróbowania nie więcej niż 614656 różnych wersji kluczy rundowych. Oczekiwana średnia liczba kluczy do wypróbowania jest znacznie mniejsza i wynosi około 466. Eksperymenty wykazały wysoką podatność rozważanej implementacji na atak. Z tego powodu zaproponowano metodę zwiększenia odporności na atak na drodze modyfikacji programu. Uzyskano znaczącą poprawę bezpieczeństwa za cenę wydłużenia czasu przetwarzania nie przekraczającego 4%.

Słowa kluczowe: szyfr blokowy, atak z użyciem błędów.

1. Introduction

Fault attack is a technique of breaking a cipher by faults inserted on purpose. Even if the cryptographic algorithm is safe under normal conditions, it can be unsafe in the presence of faults. The influence of errors on hardware implementation of block ciphers has been an object of substantial research effort in last few years [2, 3, 4, 5, 7]. Recently Saha et al. [10] showed that by inserting multiple faults into the AES state matrix, the complexity of brute force attack can be reduced to 232. Only 400 seconds were required to break the cipher using 8-core computer.

Fault attack technique is usually used for hardware implementation of a cipher. In this research there is considered the software implementation of the PP-1 cipher treated as a piece of hardware. It means that the processor and program are treated as an encryption circuit.

The main objective was to analyse in detail how faults affect the PP-1 implementation. This research is a continuation of work presented in [4]. It is intended as an introductory step to implement more tamper resistant version of PP-1.

There are many papers describing how to insert faults into a cryptographic circuit. For example, variations of supply voltages, clock glitches, temperature variations or flashes of light [1, 11], can be used for this purpose. In this paper there is simply assumed that it is possible to insert faults and focus on the results.

The paper is organized as follows: Section 2 presents the PP-1 cipher; in Section 3 there is described the concept of fault attack. Section 4 presents the results of experiments with fault insertion. Some countermeasures against fault attack are discussed in Section 5.

2. Description of the PP-1 Algorithm

The PP-1 [6] is an n-bit (n = 64, 128, 192, ...) scalable block cipher. The key length is n or 2n. The PP-1 is an involutional Substitution-Permutation Network. It uses one S-box, which is an involution, and a bit permutation, which also is an involution. As a result, the same algorithm can be used both for encryption and decryption. The PP-1 structure is shown in Figs. 1 and 2.

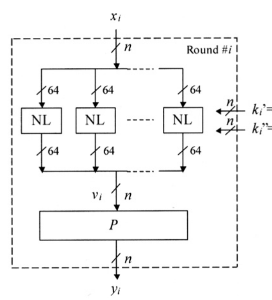

Fig. 1. The PP-1 cipher
Rys.1. Szyfr PP-1

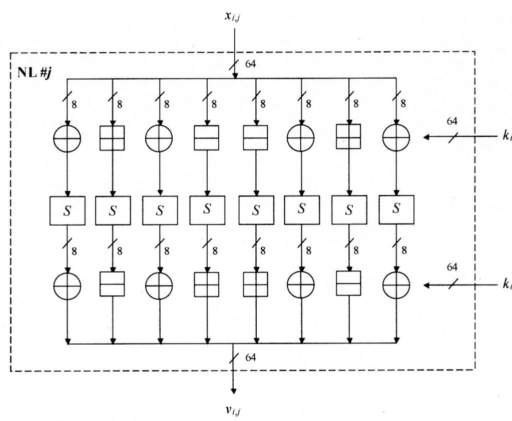

Fig. 2. The structure of NL element
Rys. 2. Struktura elementu NL

Symbols ⊕, +,– stand for xor, addition, subtraction, on 8-bit arguments. The S element is an 8×8 S-box. The *P* block is an *n*-bit permutation. The number of rounds depends on *n*. There are 11, 22, 32 and 43 rounds for *n*=64, 128, 192, 256, respectively. In each round two *n*-bit round keys are used.

The results presented in this paper are obtained for the 64-bit data block 128-bit key version of PP-1.

3. The Attack Concept

It is well known that one round block cipher can be easily broken (e.g., only 2 pairs of plaintext-ciphertext are required to break one round AES [9]). Let us assume that by inserting fault (faults) the circuit functionality have been changed in such a way that it produces a result which corresponds to one round version of the cipher with the same key.

To evaluate to what extent this can be useful for breaking the cipher we need to find answers to two questions: “what is the probability of obtaining such a result?”, and “how many results are required to break the cipher?”

Now we will show the solution to the second question. The first problem is addressed in the next section.

Let us consider the PP-1 cipher shown in Fig. 1. As we can see, there are 2 round keys applied in each round. Therefore, we can expect that breaking the one round version of the cipher, using pairs plaintext-ciphertext, will be more complicated than in a cipher with one round key in each round (e.g., the AES).

The structure of the PP-1 processing element is shown in detail in Fig. 2. There are 8 8-bit sections belonging to 2 classes. Half of the sections use operations xor-xor, whereas the other part operations plus-minus.

Let us assume that we have two different pairs plaintext ciphertext (m1,c1) (m2,c2) for

round 1 (a case with 1 pair was described in [4]). For xor xor section (Fig. 2) we have a system of equations:

$$c1 = S(m1 \oplus k1) \oplus k2 \qquad (1)$$
$$c2 = S(m2 \oplus k1) \oplus k2$$

Similarly, for *plus minus* section we have:

$$c1 = S(m1 + k1) - k2 \qquad (2)$$
$$c2 = S(m2 + k1) - k2$$

We know m1, c1, m2, c2 and we want to find k1, k2. The problem is: how many solutions exist for systems of equations (1) and (2)?

The answer is not obvious because of the nonlinear element S. Solutions were found by checking all possible m1, m2, c1, c2 such that and . The results are given in Tables 1 and 2.

Tab. 1. Solution distribution for *xor xor*
Tab. 1. Rozkład rozwiązań dla *xor xor*

Solutions count	Found times	%
0	1076920320	50,54
1	0	0
2	1038090240	48,72
3	0	0
4	15728640	0,74
Total	2130739200	100

Tab. 2. Solution distribution for *plus minus*
Tab. 2. Rozkład rozwiązań dla *plus minus*

Solutions count	Found times	%
0	779812864	36,5982
1	784203776	36,8043
2	392986624	18,4437
3	135462912	6,3576
4	30998528	1,4548
5	5505024	0,2584
6	1376256	0,0646
7	393216	0,0185
Total	2130739200	100

As we can see in Table 1, if a solution for xor xor exists, there are 2 unequally distributed possibilities: in about 98.5% of the cases we get two pairs of k1, k2, whereas in about 1.5% of the cases we have 4 different pairs of k1, k2. For the plus minus section from 1 to 7 solutions exist (Table 2).

We will use these results to estimate the number of pairs of plaintext ciphertext required to break the cipher. As we can see in Fig. 2, there are four xor xor sections and four plus minus sections (the order of operations + – is of no importance). The smallest possible numbers of solutions are 2 for xor xor and 1 for plus minus. The total number of pairs of plaintext ciphertext required to break the cipher is [4]:

$$Lmin = 2^4 \times 1^4 = 16 \qquad (3)$$

The biggest possible numbers of solutions are 4 for *xor xor* and 7 for *plus minus*. In this case the total number of pairs plaintext ciphertext required to break the cipher is:

$$Lmax = 4^4 \times 7^4 = 614656 \qquad (4)$$

The weighted average is:

$$Laverage = 466.00215 \qquad (5)$$

Generally, for the PP-1 with n-bit data block and 2n-bit key we have

$$Lmax\_n = 4^{n/16} \times 7^{n/16} \qquad (6)$$

4. Experiments with faults insertion

We considered 3 types of faults: stack-at-0, stack-at-1 and bit-flop. Here we present the results for bit-flop fault (this also covers stack-at-0 and stack-at-1).

Tab. 3. Bit-flop faults, 10000 experiments
Tab. 3. Wyniki dla modelu odwrócenie wartości bitu; 10000 eksperymentów

Number of faults	Proper result	Round 1	Round 1 without permutation	Number of different results	Infinite loop
1	4576	54	169	1767	329
2	2105	66	314	3188	608
3	995	53	400	4243	818
4	451	54	448	4736	994
5	221	36	491	4927	1130
7	52	20	508	4959	1339
9	15	8	480	4840	1480
10	6	7	436	4798	1531
15	1	5	284	4850	1570
30	0	0	48	4684	1435

The program under test was executed on a specially prepared virtual machine. It was augmented with a fault insertion module and a results collection module. Several possible results were recorded: a result (same as for fault-free cipher implementation), a result the same as for a one-round fault-free cipher (the most interesting for cipher breaking), a result the same as for a one-round fault-free cipher, but without permutation (also very interesting for cipher breaking – in a one round version of a block cipher the permutation is of no cryptographic value), and an infinite loop. The results are presented in Table 3.

As we can see in Table 3, for 1 to 15 faults we can expect to obtain a result which can be used for attack on the cipher with the average probability 0.042. In this case, using 2 series of 100 experiments we can break the cipher with 0.97 probability. The probability of success in breaking the cipher as a function of the number of experiments is shown in Fig. 3.

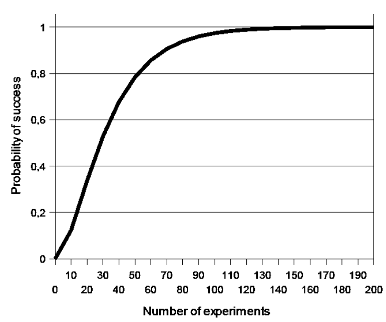

Fig. 3. Probability of breaking the cipher
Fig. 3. Prawdopodobieństwo złamania szyfru

For more in-depth analysis we considered faults limited only to program instruction (excluding data). We considered separately the faults in instruction code and in instruction argument. The average probabilities of obtaining a round 1 result (either with permutation or without permutation) are presented in Table 4.

Tab. 4. Probability of obtaining the result useful for cipher breaking
Tab. 4. Prawdopodobieństwo uzyskania rozwiązania przydatnego do złamania szyfru

No of Faults	Probability of round 1 result		
	Whole program	Instruction code	Instruction argument
1-5	0,0417	0,0687	0,0707
1-10	0,0457	0,0439	0,0432
1-15	0,0423	0,0303	0,0294

As we can see in Table 4, when a fault affects the program instruction, we can break the cipher more easily, but only if the number of faults is small (it is difficult to precisely insert one fault or an exact number of faults in a real circuit; inserting several faults is more probable). On the other hand, there is no significant difference between a fault affecting instruction code and a fault in instruction argument.

5. Improvements of resistance against fault attack

The danger of cipher breaking can be reduced by using more tamper resistant implementation. As was presented in Sections 4 and 5, the main problem is that in presence of faults the circuit can produce a result which can be used for breaking the cipher. Therefore the required countermeasures should reduce the probability of such an event.

One obvious solution is to add concurrent error detection in the executed program. If error is detected, no results (or some randomly chosen results) are presented at the output. This can be achieved in many ways. Some extra hardware is usually required to implement concurrent error detection effectively. To check that proper instructions are executed in proper order we can use signature analysis based on using a linear feedback shift register (LSFR). The main advantage of this approach is that it requires small overhead in hardware. All processed data (in the considered case executed processor instructions) are delivered to the LSFR input. As a result, we obtain some fixed-length vector of bits (e.g. 8-bit, 16-bit, or more) called signature. The computed signature is compared with those for known good implementation.

The probability of error detection depends on the signature length

$$p=1-2^{-m} \tag{7}$$

where p denotes the probability of error detection, m is the signature length.

For example, for m = 8 we get p ≈ 0.996; for m = 16 p ≈ 0.9998.

Another approach is to rearrange the program code to reduce probability of obtaining potentially hazardous for the cipher security result. The main advantage of this approach is that no support in hardware is required. Therefore we applied this approach to our implementation of the PP-1 cipher.

As it has been presented in the previous section, if faults lead to a round one result at the output, it causes a hazard for the cipher security. To avoid this, the program code has been changed in such a way, that the results of consecutive rounds are modified before storing in memory. Only the final result is disclosed. This modification is achieved by xoring the data with some mask.

The mask is secret or may be chosen at random. We used fixed mask in our experiments. The results are given in Table 5. As we can see, the results for round 1 are completely eliminated. In fact, it is still possible that multiple faults can cause a round 1 result, but the probability is so small that none of the simulation experiments show this.

Extra instructions are needed to perform masking, which leads to 7% overhead in program size and 4% overhead in processing time.

Tab. 5. Bit-flop faults; 10000 experiments - tamper-resistant version
Tab. 5. Defekty typu bit-flop; 10000 eksperymentów –wersja odporna na atak

Number of faults	Proper result	Round 1	Round 1 without permutation	Number of different results	Infinite loop
1	4074	0	0	2051	346
2	1628	0	0	3774	584
3	627	0	0	4808	804
4	250	0	0	5124	971
5	114	0	0	5099	1063
7	7	0	0	4882	1245
9	1	0	0	4642	1340
10	0	0	0	4516	1342
15	0	0	0	4334	1362
30	0	0	0	3570	1215

6. Conclusions

Fault attack may be an effective method for breaking the cipher. The research shows that the considered implementation of the PP-1 may be broken relatively easily. Only two series of 100 experiments with fault insertion are required to break the cipher with probability greater than 0.97.

We also show that the PP-1 implementation may be substantially improved using relatively simple program modifications. The time overhead for such tamper-resistant implementation does not exceed 4%.

This proposed method of fault attack is applicable to all ciphers in which the main key, or all the round keys, can be deduced from the round key for the first round, and the round key is applied twice in each round.

This work was partially supported by the Polish Ministry of Science as a 2010–2013 research project.

7. References

[1] Bar-El H., Choukri H, Naccache D., Tunstall M and Whelan C.: The Sorcerer's Apprentice Guide to Fault Attacks, Cryptology ePrint Archive: Report 2004/100, http://eprint.iacr.org/2004/100.

[2] Bertoni G., Bregeveglieri L., Koren I., Maistri P.: An Operation-Centered Approach to Fault Detection in Symmetric Cryptography Ciphers, IEEE Trans. On Computers Vol. 56 No 5, May 2007, pp.635-649.

[3] Bertoni G., Bregeveglieri L., Koren I., Maistri P., Piuri V.: Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard, IEEE Trans. On Computers Vol. 52 No 4, April 2003, pp.492-505.

[4] Bucholc K.: Fault Attack Technique Against Software Implementation of a Block Cipher, Pomiary Automatyka Kontrola, vol. 55, 2009, pp. 831-834.

[5] Bucholc K., Idzikowska E.: Multiple Error Detection in Substitution Blocks for Block Ciphers. Advances in Information Processing and Protection, ed. J.Pejas and K.Saeed, Springer US, 2007, pp. 181-190.

[6] Chmiel K., Grocholewska-Czuryło A., P. Socha, Stokłosa J.: Scalable cipher for limited resources. Polish Journal of Environmental Studies, Vol. 17/4C/2008 pp. 371-377.

[7] Joye M., Mannet P., Rgaud J.B.: Strengthening hardware AES implementations against fault attacks, IET Inf. Secur., vol. 1, 2007, pp.106-110.

[8] National Inst. Of Standards and Technology, Federal Information Processing Standard 197, The advanced Encryption Standard (AES), November 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[9] Piret G. and Quisquater J.J.: A differential fault attack technique against SPN structures, with application to the AES and Khazad, Proc. of CHES 2003, pp. 77-88.

[10] Saha D., Mukhopadhyay D., Chowdhury D. R.: A Diagonal Fault Attack on the Advanced Encryption Standard, http://eprint.iacr.org/2009/581.pdf.

[11] Skorobogatov S and Anderson R.. Optical fault induction attacks. Cryptographic Hardware and Embedded Systems Workshop (CHES-2002), pages 2-12, 2002. Lecture Notes in Computer Science No. 2523.

otrzymano / received: 21.09.2010
przyjęto do druku / accepted: 01.11.2010 *artykuł recenzowany*