Comparison of some mask protections of DES against power analysis

Kai Cao[1,a], Dawu Gu[1,b], Zheng Guo[1,2,c] and Junrong Liu[1,2,d]

[1]Shanghai Jiao Tong University, China

[2]Shanghai Viewsource information science & technology Co.,Ltd, China

[a]ck89119@sjtu.edu.cn, [b]dwgu@sjtu.edu.cn, [c]guozheng@sjtu.edu.cn, [d]junrong.liu@viewsource.com

Keywords: Side Channel Attack (SCA), Power analysis, Mask protection.

Abstract. Power analysis, proposed by Paul Kocher in 1998, is now a common kind of side channel attack on cryptographic devices. Mask technology protects devices by randomizing the intermediate values that are processed by cryptographic devices which makes the power consumption independent of the intermediate values. In this paper, we introduce three mask protections, and make a comparison of them by attacking them on simulation platform.

Introduction

While running, any cryptographic device cannot avoid leaking various types of information, such as power consumption, sound wave, electromagnetic wave and so on, which are called side channel information. After being collected, processed and analyzed, side-channel information can be used to recover secret information of cryptographic devices. Such attacks which are carried out by making use of the side-channel information are called Side Channel Attack (SCA). Power analysis, which was put forward in 1998 by Paul Kocher, is currently one of the most widely used side-channel attack method. It recovers secret information by analyzing power consumption of cryptographic devices. Nowadays, researches on power analysis develop quickly. There have been many deep studies on attacks and countermeasures for common cryptographic algorithms such as DES, AES, RSA and so on.

As cryptographic devices such as smart cards are widely used nowadays, faced with great threat form side-channel attack, researchers put forward a serious of useful countermeasures, one of which is mask protection. By randomizing the intermediate values operated by cryptographic devices, mask protection eliminates dependencies between power consumption and operated data, so as to achieve the purpose of security protection. This paper takes DES algorithm for example, makes a comparison of three implementations of mask protection, and contrasts the effects of these methods by experiment. The experiment device we use in this paper is Riscure's Inspector power analysis software.

Differential Power Analysis

Differential Power Analysis (DPA) is the most popular power analysis method nowadays. It exploit the data dependency of power consumption of cryptographic devices. DPA use a large number of power traces to analyze the power consumption at a fixed moment of time as a function of the processed data.

This attack strategy consists of five steps.

Choosing an intermediate result of the executed algorithm.

This intermediate result needs to be a function f(d, k), where d is a known value and k is a small part of the key.

Measuring the power consumption.
We write the known data values as vector $\mathbf{d} = (d_1, \dots, d_D)'$, where d_i denotes the data value in i-th encryption or decryption run. We refer to the power trace that corresponds data d_i as $\mathbf{t_i}' = (t_{i,1}, \dots, t_{i,T})$, where T denotes the length of the trace, and the traces can be written as matrix **T** of size D * T.

Calculating hypothetical intermediate values.
We write the possible keys as vector $\mathbf{k} = (k_1, \dots, k_K)$, where K denotes the total number of possible choices for k. Given the data vector **d** and the key hypotheses **k**, an attacker can easily calculate hypothetical intermediate values f(d, k) for all D encryption runs and for all K key hypotheses. We denote $f(d_i, k_j)$ as $v_{i,j}$, then we can get a matrix **V** of size D * K.

Mapping intermediate values to power consumption values.
By using simulation techniques, we can get the power consumption $h_{i,j}$ for each hypothetical intermediate value $v_{i,j}$, which means we get an hypothetical power consumption matrix **H**.

Comparing the hypothetical power consumption values with the power traces.
In this step, each column $\mathbf{h_i}$ of **H** is compared with each column $\mathbf{t_j}$ of **T**. The result of this comparison is matrix **R** of size K * T, where each element $r_{i,j}$ contains the result of the comparison between column $\mathbf{h_i}$ and $\mathbf{t_j}$. If the highest value of **R** is $r_{ck,ct}$, it means the column $\mathbf{h_{ck}}$ and $\mathbf{t_{ct}}$ are strong related. The indices of this value are the result of DPA attack.

Three Mask Protections

To against the threat of DPA, researchers worked out some effective countermeasures, one of which is mask protection. Now, we introduce three mask protections in detail.

Akkar's Protection

In this protection, a boolean mask X is applied before the Initial Permutation IP (we XOR the 64-bit message M with a 64-bit value X), and it will not be removed until the Final Permutation FP. The only non-linear part of DES algorithm is the S-Box. As a result of using the protective countermeasure, we need a modified S-Box to make the algorithm executing correctly.

The main process (one round) of the protection is as Fig. 1, where:

- X1 represents the 64-bit value IP(X);
- IP represents the initial permutation;
- $X1_{0-31}$ (respectively $X1_{32-63}$) represents the 32-bit low-weight (respectively high-weight) part of the 64-bit mask X;
- X2 represents the 48-bit value $EP(X1_{32-63})$;
- EP represents the expansive permutation of a DES round.

At the beginning of the algorithm, the message M (64-bit) will be masked by a 64-bit mask X. Then we start with the value $M \oplus X$. As X is known, by the introduction before, it is easy to get the values of X1 and X2. To reestablish the mask X1 at each round, we will use a modified S-box, denoted SM-Box. The result of SM-Box, which will be permutated by Permutation P and be XOR-ed with the left part message, must have a mask corresponding to X132-63. So, the SM-Box must satisfy that:

$$\text{SM-Box}(A) = \text{S-Box}(A \oplus X2) \oplus \text{P\_inv}(X1_{0\text{-}31} \oplus X1_{32\text{-}63})$$

where P_inv is the inverse of the permutation P.

It is also necessary to modify the left part of the message (XOR it with $X1_{0\text{-}31} \oplus X1_{32\text{-}63}$), so that the mask X1 will be preserved at the end of the round.

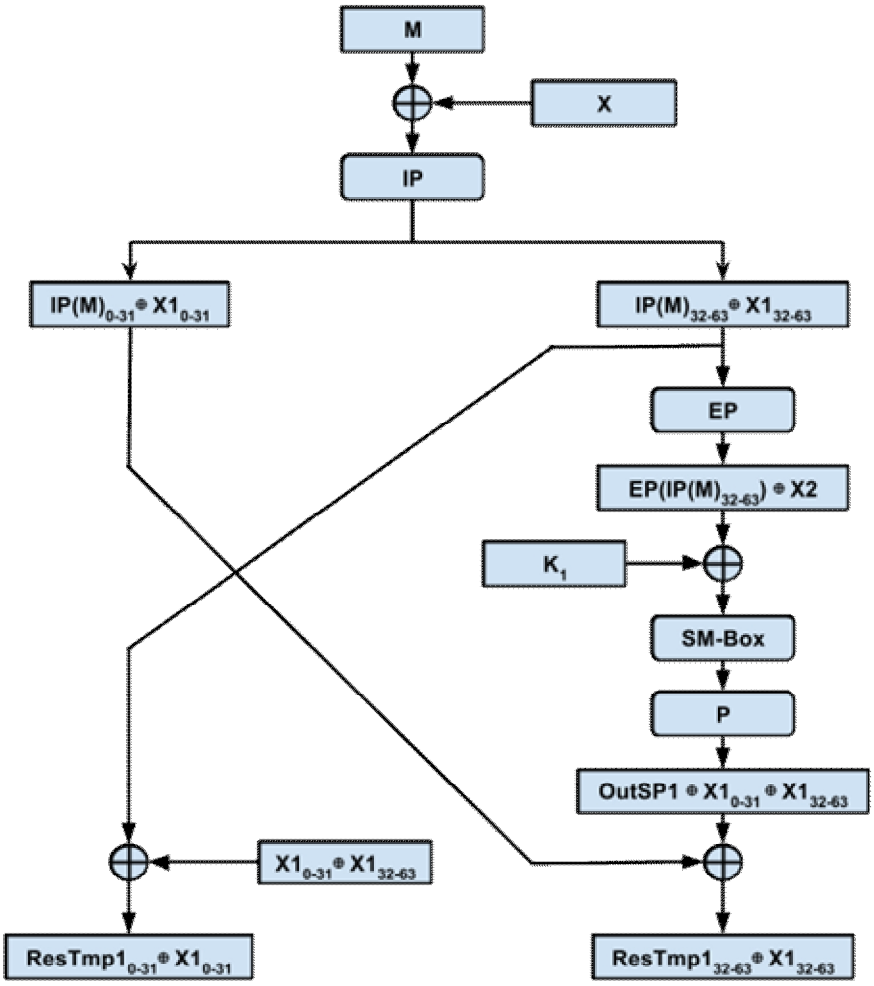

Fig. 1

Rivain's Protection

The main idea of this countermeasure is splitting real value into three parts, so that adversary can not get the real value algorithm processes. More specifically, the cipher state p and secret key k are represented by three shares -- (p0, p1, p2) and (k0, k1, k2), which satisfy following relations:

$$p = p0 \oplus p1 \oplus p2$$

$$k = k0 \oplus k1 \oplus k2$$

In order to ensure the security, (p1, p2) and (k1, k2), which are called masks, are generated randomly. As p and k are known, p0 and k0 can be calculated by the preceding equations.

The round transformation of this protection is as Fig. 2：

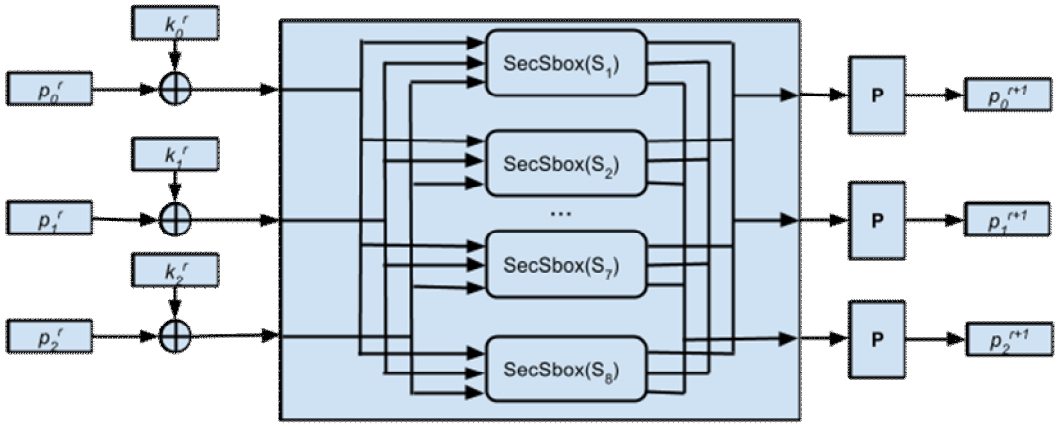

Fig. 2

The main problem of this transformation is SecSbox. If this problem can be solved, then the protection algorithm can be implemented as follows:

1. (p1, p2) = rand()
2. p0 = P ⊕ p1 ⊕ p2
3. for r = 0 to 15 do
4. (k0, k1, k2) = α_{sec} ((k0, k1, k2), r)
5. (p0, p1, p2) = (p0 ⊕ k0, p1 ⊕ k1, p2 ⊕ k2)
6. (p′1, p′2) = rand()
7. for j = 1 to 8 do
8. $(p'0)_j$ = SecSbox(S_j, $(p0)_j$, $(p1)_j$, $(p2)_j$, $(p'1)_j$, $(p'2)_j$)
9. (p0, p1, p2) = (P(p′0), P(p′1), P(p′2))
10. return p0 ⊕p1 ⊕p2

The inputs of this algorithm are: a plaintext P, a masked key k0 = K ⊕ k1 ⊕ k2 with random masks (k1, k2). The output is ciphertext C.

Let's consider about the implementation of SecSbox. As we described above, j-th SecSbox's inputs are: a masked values $p0_j$ (denoted x', if the real value is x, then x' = x ⊕ r1 ⊕ r2), a pair of input masks ($p1_j$, $p2_j$) (denoted r1 and r2), a pair of output masks ($p'1_j$, $p'2_j$) (denoted s1 and s2) and a LUT S_j(as known as Sbox, denoted S), and output is the masked Sbox shall output S(x) ⊕ s1 ⊕ s2.

Following algorithm is one of the proposals which satisfy above requirements:

1. r3 = rand(n)
2. r′ = (r1 ⊕ r3) ⊕ r2
3. for a = 0 to $2^n - 1$ do
4. a′ = a ⊕ r′
5. T[a′] = (S(x′ ⊕ a) ⊕ s1) ⊕ s2

6. return T[r3]

Variable n in above algorithm means the number of bits of SecSbox input, and it is 6 in case of DES.

Our Proposal

Inspired by Rivain's work, we proposed a new protection countermeasure. The main process of this protection in one round is show as Fig. 3.

As Rivain's algorithm, we split the right part R into three shares to hide the real value. Additionally, we XOR a random number r with R before it entering SecXor module, and r will be removed in SecXor module. During SecXor operation, we add the real value with a global variable gcnt which will be eliminated in SecSbox.

SecSbox has three inputs: data (48-bit), random mask ra (48-bit) and random mask rb (48-bit), and three outputs. We divide inputs into eight parts (6 bits each part), each into a corresponding Sbox, and finally we combine the outputs of all Sboxes as three outputs. SecSbox consists of eight same Sboxes, each Sbox has three parts and each part has the same structure which is as Fig. 4.

do_i in the Fig. 4 denotes S(i), the output of a standard Sbox in DES algorithm with input i. The busy signal is used to trigger an rotation operation, and do_i can be rotated to do_(i+gcnt) by gcnt rotation operations. By performing this operation, the global variable gcnt will be eliminated, which ensures that the output of protection algorithm is as the same as normal DES.

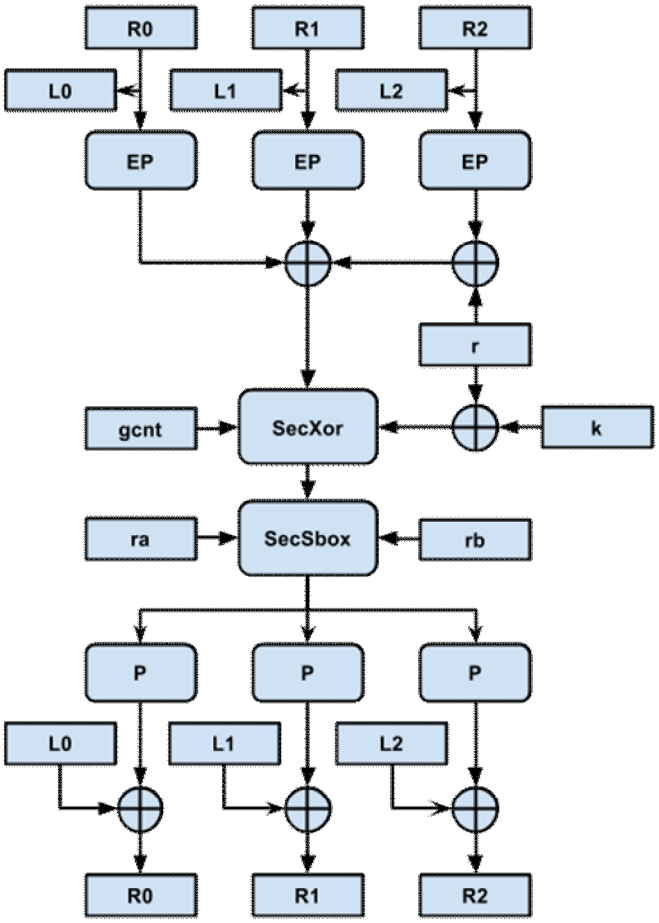

Fig. 3

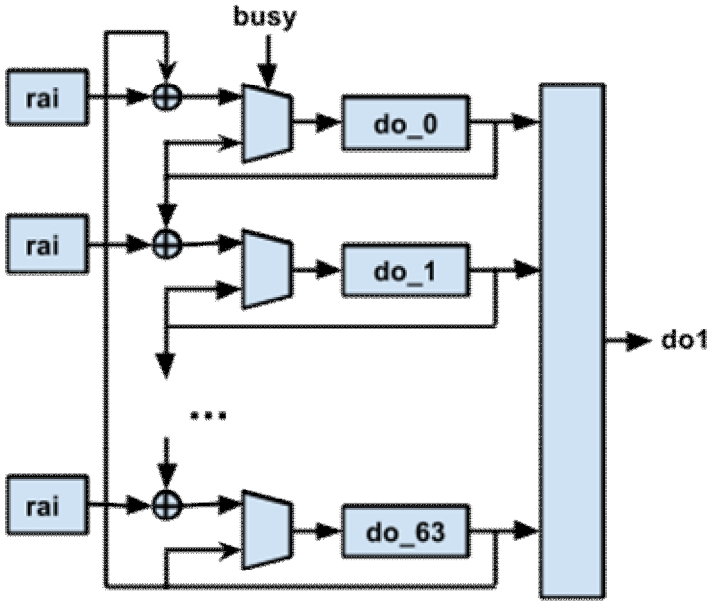

Fig. 4

Attack in practice

For the three protections above, we get 100000 power consumption curves each by simulation software and analysis the curves by Inspector. The result is as follows:

Akkar's protection:

We can reveal the correct key by only 6000 curves, and the result shows as Table 1.

Sbox	1	2	3	4	5	6	7	8
Subkey	0x23	0x1D	0x1C	0x27	0x38	0x33	0x31	0x32
Correction	0.1095	0.0800	0.1106	0.0848	0.0593	0.0439	0.0689	0.0627

Table 1

Rivain's protection:

We can get the key by about all 80000 curves, and the result shows as Table 2.

Sbox	1	2	3	4	5	6	7	8
Subkey	0x14	0x2B	0x4F	0x0C	0x2D	0x11	0x1E	0x12
Correction	0.0305	0.0561	0.1179	0.0681	0.0737	0.0778	0.0621	0.0565

Table 2

Our protection:

After processing all 100000 curves, we can't find the correct key of algorithm, and the result shows as Table 3.

Sbox	1	2	3	4	5	6	7	8
Subkey	0x30	0x16	0x10	0x30	0x12	0x09	0x3A	0x3E
Correction	0.0446	0.0966	0.0402	0.1110	0.0363	0.0692	0.0773	0.0334

Table 3

Conclusion

In this paper, we introduce three protection countermeasures in detail and analyze the simulation curves of these protections. By comparing the results of power analysis, we find that Akkar's protection is too weak under the attack nowadays. Rivain's protection have some effect, but is not good enough. Compared to Rivain's work, the protection we proposed has some improvements which have been proved effectively in practice.

Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 61402286, 61472250, 61472249), Special Fund Task for Enterprise Innovation Cooperation from Shanghai Municipal Commission of Economy and Informatization (No. CXY-2013-35), Plan of action for the innovation of science and technology of Shanghai (14511100300), Minhang District innovation project (No. 2015MH069).

References

[1]Paul Kocher, Joshua Jaffe, and Benjamin Jun, Differential Power Analysis, Proceedings of Advances in Cryptology-CRYPTO'99, pp.388-397, 1999.

[2] Mehdi-Laurent Akkar and Christophe Giraud, An Implementation of DES and AES, Secure against Some Attacks, CHES 2001, LNCS, Vol.2162/2001, pp.309-318, Springer, Heidelberg, 2001.

[3]Matthieu Rivain, Emmanuelle Dottax and Emmanuel Prouff, Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis, FSE 2008, LNCS 5086, pp. 127–143, 2008

[4]Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power Analysis Attacks: Revealing the Secrets of Smart Cards, Springer, LLC, 2007.