False Secret Keys to Disturb Power Analysis

Seungkwang Lee, Taesung Kim
Information Security Research Division, ETRI
218 Gajeong-ro, Yuseong-gu, Daejeon, 34129 Korea.
{skwang, taesung}@etri.re.kr

***Abstract*—** Protecting against power analysis on block cipher implementations is no longer optional but mandatory. One of the most well-known techniques to prevent power analysis is higher-order masking which randomizes sensitive intermediate values by using masks picked randomly for each execution of cryptographic algorithms. Because of its slow performance however, we need to find a more simple way. In this paper, we present a new method for achieving this goal. In the presence of deceiving runs of encryption using several false keys, we aim to disturb power analysis. Our experiment finds out how many false keys are required to disturb power analysis.

Keywords—block cipher; power analysis; countermeasure

I. Introduction

Now that any unprotected implementation of block ciphers in embedded devices can be easily broken by power analysis, it is responsible for developers to protect secret keys from being exposed. Among a variety of protecting methods, a masking scheme is popularly used in most cases. Because a first-order masking scheme, which uses only one random mask to hide an intermediate value, is vulnerable to the higher-order DPA (HODPA), a higher-order masking has attracted a lot of attention so far. In general, a higher-order masking protecting against d-th order DPA [1] is called the d-th order masking. In this case, every intermediate value x occurring during the cryptographic operations is randomly split into d+1 shares (x_0,x_1, ..., x_d) in order to satisfy the following relation for a group operation $\perp$:

$$\perp(x_0, x_1, ..., x_d) = x.$$

This randomly split (d+1)-tuple guarantees that there is no combination of d or less elements depending on x. Thus, security against d-th order DPA can be assured. In most cases, $\perp$ is the exclusive-or (XOR) operation denoted by $\oplus$. However, the computational costs due to a higher-order masking scheme are typically quite high — tens to hundreds times slower than a straightforward implementation in case of AES. This high computational effort is largely due to the precomputation of the masked lookup tables because masks have to be different from execution to execution.

In this paper, we propose another countermeasure, which is a non-masking-like method, to make power analysis unsuccessful. To be specific, we randomly pick several false keys and execute dummy encryptions using them. It is somewhat similar to the inserting dummy operations and shuffling, but overcomes their drawbacks.

II. Related Work

A. Inserting Dummy Operations and Shuffling

The basic idea of inserting dummy operations is literally to randomly insert dummy operations before, during, and after the execution of cryptographic operations [2]. Because the total number of inserted operations is the same for all executions, an attacker is unable to obtain any useful information about the number of inserted operations by measuring the execution time. However, too many dummy operations decrease the execution speed, and a programmer is unable to reuse well-implemented standard libraries. Shuffling, an alternative to the insertion of dummy operations, is a random change in the operation sequence of a cryptographic algorithm. This does not reduce throughput as much as inserting dummy operations. However, it is not always applicable because the number of operations that can be shuffled in a particular cryptographic algorithm is limited. In practice, the random insertion of dummy operations and shuffling are often combined to provide randomness in the power consumption. Unfortunately, even this combination does not guarantee a high level of protection against power analysis.

B. Masking

Masking is a technique to randomize the intermediate values processed by a cryptographic device. So far, various masking techniques have been proposed in many publications [3][4]. Unfortunately, several higher-order DPA attacks [5][6] have been shown capable of breaking first-order countermeasures. Moreover, multiplicative masks have one major disadvantage that they cannot hide the intermediate value 0. Though many studies of higher-order masking have been proposed, the costs of masking are typically high in terms of computation time — tens to hundreds times slower than a straightforward AES implementation.

C. Software Dual-rail Implementation

Countermeasures using software DRP logic have been suggested [7][8]. In [7], a PRESENT algorithm is implemented using the DRP logic of 2-bit representation. However, the authors in [8] show that this is still vulnerable to power analysis, and they also demonstrate that it is still vulnerable to even if the DRP logic is extended to more than 2-bit representation. It is because the power consumption in the device is affected by both the HW and bit positions set.

III. Proposed Scheme

Recall that the inserting dummy operations and shuffling methods are to reduce a correlation to the key at a particular point of the power trace. As pointed out previously, they have their own drawbacks; the random insertion of dummy operations needs a re-implementation of existing libraries and shuffling cannot be applied in all cases. Our proposed scheme combines their main characteristics and gets over their shortcomings at the same time. In general, we call an API to encrypt a message *m* using a secret key *k* for a particular algorithm *E* by *E*(*m*, *k*). Our proposed scheme calls the function *E* as it is without having to modify it for the power analysis protection. The following explains the details.

First, randomly generate *n* false keys *k*1, *k*2, … , *k*n. Then we have now a correct key *k* and *n* false keys.

Second, given *n*+1 keys (*k*, *k*1, *k*2, …, *k*n), randomly permutate their order. Let this re-ordered key list by (*K*0, *K*1, *K*2, … , *K*n). Remember the index *i* for the correct key.

Third, for $j \in [0, n]$, call *Cj* = *E*(*m*, *Kj*), where *C* is a ciphertext.

Finally, return the correct ciphertext *Ci* obtained by the correct key *Ki*.

Then, our method requires *n*+1 executions of encryption in total. Also, additional memory space to store *n* more keys is needed.

It is previously studied that if the moment when the target operation occurs is uniformly distributed across *n* time instants, the number of power traces for a successful DPA grows in n^2 in case that DPA is performed straightforwardly [9]. In the following section, we perform several experiments to show how many false keys interfere with power analysis.

IV. Experimental results

In this section, we perform CPA attacks on SubBytes output in the first round for a non-protected AES-128 and our protected AES-128 implementations. To be specific, CPA attacks are performed both in mono-bit and HW-based models using SCARF, a side-channel analysis framework [10].

A. CPA on Non-protected AES-128

On an AES-128 implementation without any protection, we have performed mono-bit and HW-based CPA attacks. To generate noise-free power traces, we have serialized each 8-bit SubBytes output in the first round into zeros and ones to conduct mono-bit CPA and have written the HW value to cover HW-based CPA. Noise-free power traces obtained by this way considerably enhance the CPA performance compared to the classical power traces acquired by an oscilloscope. If CPA is performed with classical power traces, the number of traces for the successful attack will significantly increase.

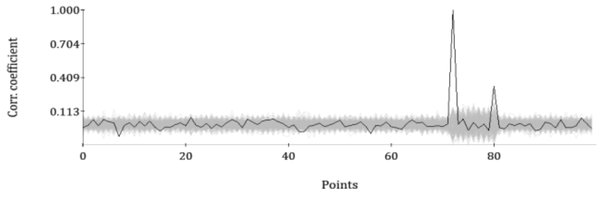


Fig. 1. Mono-bit CPA result on the non-protected AES-128. Black line: correct subkey, gray line: wrong subkey.

A mono-bit CPA revealed 9 out of 16 subkeys from the 20th trace, and all subkeys were revealed from the 25th trace. In case of the HW-based CPA, 10 traces revealed 9 subkeys and 15 traces were enough to analyze all subkeys. Fig.1 shows a mono-bit CPA result on the 8th subkey; the correct subkey has a correlation coefficient value 1.

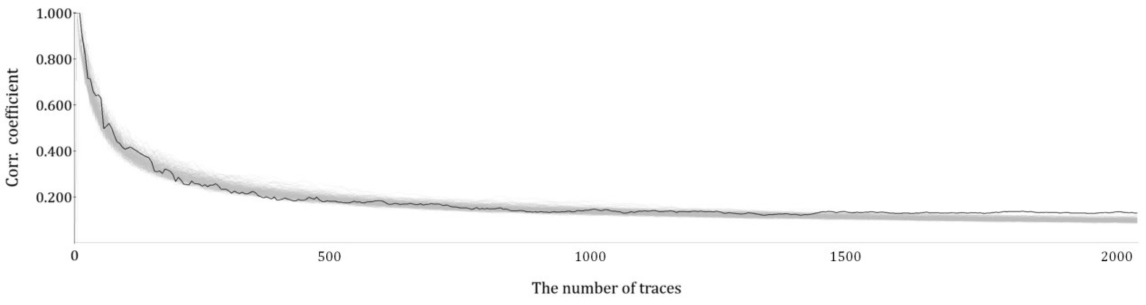
B. CPA on our Protected AES-128

Now we testify our protected method with 9 false keys in addition to a correct secret key. For each execution of AES-128 encryption, we have randomly generated 9 false keys and permutated their order. If this re-ordering at the second step in our proposed method is performed uniformly at random, the number of traces needed to succeed in CPA is supposed to increase about 100 times as pointed out previously. To demonstrate it, we have collected 2000 noise-free traces with random plaintexts and performed mono-bit and HW-based CPA attacks. It is expected that the subkeys are not revealed with less than 100 traces and the number of exposed subkeys will be drastically reduced when performing CPA with 1000 traces. Of course, 2000 traces may be enough to reveal overall subkeys, because they are analyzed with 20 traces in case of non-protected AES-128.

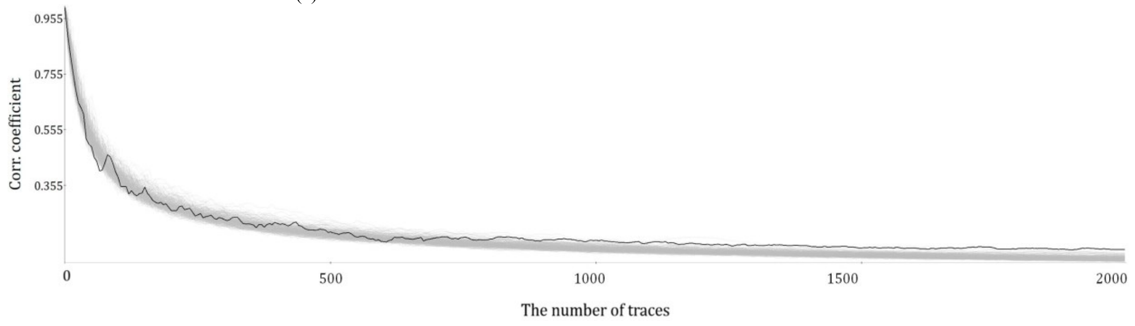
In case of mono-bit CPA, there was no subkey revealed until the 100th trace was analyzed, and 2 and 8 subkeys were revealed when analyzing the 1000th and the 1500th trace, respectively. As a result, 2000 traces found 14 subkeys in total. In case of HW-based CPA, there was also no subkey exposed at the time where the 100th trace was being analyzed, and 4 and 8 subkeys were analyzed when the 1000th and the 1500th trace were being analyzed. As a result, 2000 traces revealed 13 subkeys in total. Fig. 2 shows that the correlation coefficient of the correct subkey is not noticeably distinguishable from other candidates before analyzing the 2000th trace. Thus we can conclude our method can be resistant to power analysis.

V. Conclusion

In this paper, we presented a power analysis countermeasure that combines the characteristics of the random insertion of dummy operations and shuffling while solving their drawbacks. To do so, we randomly generated several false keys for each execution and performed the encryption operations using the correct key and the false keys in random order. This gives us two main advantages: 1) we do not have to modify well-programmed cryptographic implementations, and 2) this method can be applied in all cases regardless of the cryptographic details. Our experiment results demonstrated that *n*-1 false keys make CPA approximately n^2-times complex.



(a) Mono-bit CPA correlation coefficient for different numbers of traces.



(b) HW-based CPA correlation coefficient for different numbers of traces

Fig. 2. CPA results on our protected AES-128. Black line: correct subkey, gray line: wrong subkey.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00364, Development of Training-based Template Attack Software for Security Verification of Secret Data Leakage of Embedded Secure Element [the T&HO project]), and (No.2016-0-00399, Study on Secure Key Hiding Technology for IoT Devices [the KeyHAS] project).

REFERENCES

[1] T. S. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software," in *Proc. 2nd Int. Workshop CHES*, MA, USA, August 17-18, 2000.

[2] S. Mangard, E. Oswald, and T. Popp, "Power Analysis Attacks: Revealing the Secrets of Smart Cards," in *Advances in Information Security*, 2007.

[3] M.-L. Akkar and C. Giraud, "An Implementation of DES and AES, Secure against Some Attacks," in *Proc. 3rd Int. Workshop CHES*, London, UK, 2001, pp. 309‑318.

[4] J.-S. Coron and L. Goubin, "On Boolean and Arithmetic Masking Against Differential Power Analysis," in *Proc. 2nd Int. Workshop CHES*, London, UK, 2000, pp. 231–237.

[5] M. Joye, P. Paillier, and B. Schoenmakers, "On Second-Order Differential Power Analysis," in *Proc. 7th Int. Workshop CHES*, 2005, pp. 293–308.

[6] J. Waddle and D. Wagner, "Towards Efficient Second-Order Power Analysis," *Lecture Notes in Computer Science*, vol. 3156. Springer, 2004, pp. 1–15.

[7] P. HOOGVORST, G. DUC, and J.-L. DANGER, "Software Implementation of Dual-Rail Representation," in *Proc. 2nd Int. Workshop, COSADE,* 2011, pp. 73–81.

[8] A. MAEKAWA, N. YAMASHITA, T. OKAMURA, K. MINEMATSU, T. SUZAKI, and Y. TSUNOO, "Tamper-Resistance Techniques Based on Symbolic Implementation against Power Analysis," in *30th Symposium on Cryptography and Information Security*, 2013, pp. 73–81.

[9] J. GroBschadl and I. Kizhvatov, "Performance and Security Aspects of Client-Side SSL/TLS Processing", *Cryptology and Network Security: 9th Int. Conf.*, 2010.

[10] Seungkwang Lee, Dooho Choi, and Yongje Choi, "Improved Shamir's CRT-RSA Algorithm: Revisit with the Modulus Chaining Method," *ETRI J.*, vol. 36, no. 3, June. 2014, pp. 469-478.