

Ciphertext and Plaintext Leakage Reveals the Entire TDES Key

Yongbo Hu, Chen Zhang, Yeyang Zheng, Mathias Wagner

firstname.lastname@nxp.com

NXP Semiconductors

Abstract

SCA(Side-channel analysis) is a well-known method to recover the sensitive data stored in security products. Meanwhile numerous countermeasures for hardware implementation of cryptographic algorithms are proposed to protect the internal data against this attack fortunately. However, some designs are not aware that the protection of the plaintext and ciphertext is also crucial. In this work, we attack an implementation TDES(triple DES) by taking advantage of such leakages detected in a widely used commercial product which is based on the hardware platform that passed the EAL5+ certification. In particular, we guess entire DES keys to construct hypotheses for the intermediate outputs in a TDES calculation. The time cost for this approach is nearly $\frac{1}{2^{32}}$ of that by a brute force. Furthermore, if in addition leakage about the key becomes available, the attack costs become practical. That is, reducing the key entropy of every DES key to 2^{28} allows an enumeration of the entire TDES in 21.6 hours.

1 Introduction

Nowadays cryptography algorithms and protocols are the critical components for billions of electronic products. Hence the sensitive data stored in an embedded device is potentially attacked by adversaries.

Among these attacking methods, Differential Power Analysis(DPA) has become a widely exploited cryptographic device vulnerability analysis technique since it was first proposed by Kocher [1]. It is powerful because at-

tackers only need to calculate the correlations between obtained side-channel information and internal sensitive data, then the secret information can be retrieved by analyzing the power consumption of various cryptographic operations.

Besides the internal data, the plaintext and ciphertext also leaks. Basically, these leakages at least leak the location of real calculations by analyzing the side-channel information. Additionally, Underwrites Laboratories have already attacked the MAC-ISO9797-1 standard successfully using the leakage of plaintext instead of exploiting sensitive data inside the DES block [9].

TDES is a standard secure algorithm and widely used in secure processors. It concludes 3 independent single DESes [5] with three keys. The encryption and decryption of TDES are as below, DES_{enc} indicates the DES encryption process, and DES_{dec} indicates the decryption.

$$\begin{aligned} \text{cipher} &= DES_{enc}(K3, DES_{dec}(K2, DES_{enc}(K1, \text{plaintext}))) \\ \text{plaintext} &= DES_{dec}(K3, DES_{enc}(K2, DES_{dec}(K3, \text{cipher}))) \end{aligned} \quad (1)$$

A 3-key TDES indicates that these three key are independent to each other and 2-key TDES is defined as $K1 = K3$. Basically a 3-key TDES need 2^{168} times of computations to exhaust the whole key search space. Even with meet-in-the-middle attack [10], both 2-key and 3-key TDES need 2^{112} times of brute forces to retrieve the private key. However, if the attacker can use plaintext and side-channel information to retrieve $K1$ using DPA, then TDES can be cracked DES by DES.

Below are the highlights of this proposal.

- Present an attack on TDES implementations where the DES itself is secure, but only plaintext, ciphertext and key leaks.
- Instead of targeting and S-box, the whole DES block is targeted at once.
- To make the attack practical, include information about key to reduce the search space.

The rest of this paper is organized as follows. In Section 2, the background and basic knowledge of side-channel analysis will be presented; Section 3 tells all the leakages we found in a particular commercial IC. In Section 4, the new DPA attack method to retrieve all the keys of TDES will be introduced in details and the attack results are presented. Additionally, the costs for a real attack is also evaluated and compared with that of a brute force attack and

a regular DPA for DES as well. Finally, we highlight some contributions we made in this paper.

2 Background

DPA has been widely studied and developed by cryptographic communities in last 20 years since Kocher's first set-up in 1999. The original DPA calculates the differential power for single bit of an internal data and several other statistical algorithms are proposed which can reduce the total number of traces needed to recover the key, such as PPA(Partitioning Power Analysis) [2], multi-bit DPA [3] and CPA(Correlation Power Analysis) [4]. The target of internal data are also from the single bit model to multi-bit with Hamming weight model. The model assumes that the power consumption is linearly related to Hamming weight. Let A be the internal data and $HW(A)$ be the Hamming weight of A . Let $T = \{T_i | 1 \leq i \leq N\}$ be the N trace acquired. This linear model is defined as below

$$T(t) = \xi H(A) + l \quad (2)$$

In Eq. (2), l indicates the noise in the traces and let ξ denote the linear parameter. Since the internal data A can be calculated by plaintexts, cipher texts and limited guessing keys. (E.g. $A = S(p \oplus k)$ where p and k are only a few bits, p is partial of known plaintexts for many traces and k needs to be guessed.) So for each key, the correlation between $HW(A_{key})$ and the traces which is defined as Pearson correlation can be computed.

Supposedly if the side-channel information perfectly match the Hamming weight model, the correlation is one for correct key. In other words, the maximum correlation indicates the correct key hypothesis.

Fortunately, we have countermeasures against DPA such as masking technology [7, 11] which protects the internal data by involving a random number to reduce or even remove the correlation between the sensitive data with side-channel information.

And after several decades of development in the hardware implementation of cryptographic algorithm, most coprocessors themselves have been hardened to resist DPA. However, the protection of plaintext and ciphertext of the algorithm have relationship with the whole system architecture and the CPU structure, which makes the protection ever harder. This paper will gen-

eralize Underwrites Laboratories' approach in order to recover a full TDES key.

3 Leakage description

The leakages are originally found in single DES algorithm with EM traces.

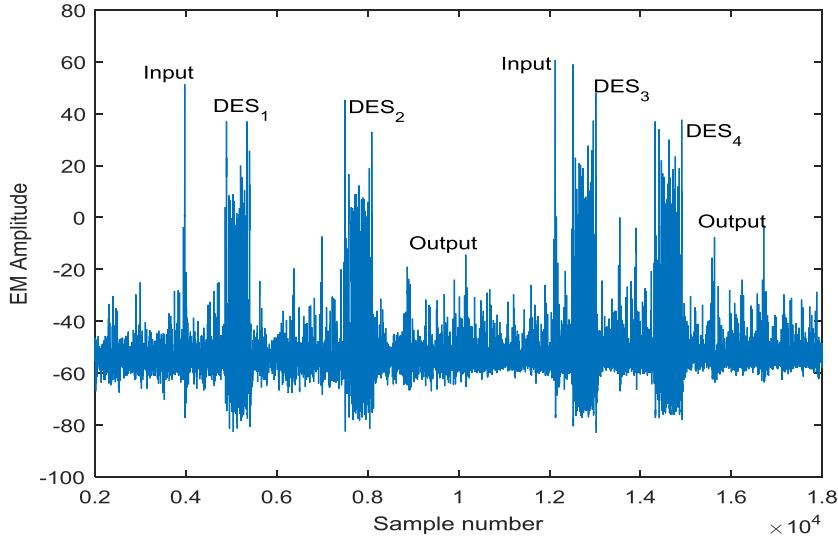


Figure 1: A typical EM rawtrace

Figure 1 illustrates a raw trace for DES encryption acquired with EM probe. Besides the input and output operations, it also tells 4 calls to the DES engine.

Let $P_{Byte}(i), C_{Byte}(i), K_{Byte}(i)\{0 \leq i \leq 7\}$ denote the 8 bytes plaintexts, cipher texts and private keys. Totally 4 kinds of leakages are found for this chip.

$$\begin{cases} HW(P_{Byte}(i) \oplus P_{Byte}(i+4))\{0 \leq i \leq 3\} \\ HW(C_{Byte}(i) \oplus C_{Byte}(i+4))\{0 \leq i \leq 3\} \\ HW(K_{Byte}(i) \oplus K_{Byte}(i+4))\{0 \leq i \leq 3\} \\ HW(P_{Byte}(i) \oplus C_{Byte}(i+4))\{0 \leq i \leq 3\} \end{cases} \quad (3)$$

When targeting on the Hamming weight of the leakage model in Eq. (3), the correlation between each model and the traces which have been properly aligned can be calculated and figured as Figure 2.

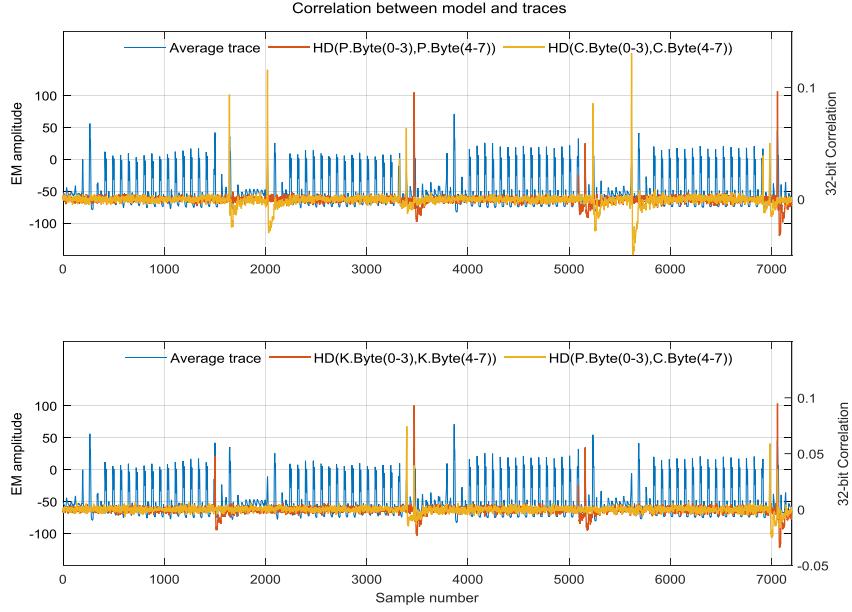


Figure 2: Correlation between 4 models and the traces

Figure 2 indicates that the leakages are coming from the hardware because the spikes are appearing at one or two clocks away from the first or last round of DES calculation, while software will cost more clock cycles for related instructions.

There are totally 4 spikes for key, so the keys of 4 DES calls are not independent with each other and these 4 DES calls are most likely the forward or backward calculations of single DES. What's more, the spikes of $P_{Byte}(i) \oplus C_{Byte}(i + 4)$ and $C_{Byte}(i) \oplus C_{Byte}(i + 4)$ are located at the same time in the second and 4th DES calls, which tell these two DES calls are most likely the same operation. Hence, the multiple calls to the DES engine are due to countermeasures against fault attacks with the order of forward-backward-forward-backward.

Additionally, the plaintext, ciphertext and key are most likely transferred

in the portion of word(32 bits) using shifting operation in the internal architecture of this DES engine. So without proper masking the data register switches from one word to another and leads to a Hamming distance leakage between the 32 bits MSB and LSB.

To see how many traces are needed to exploit the leakage, we perform a UL attack using the leakage of $P_{Byte}(i) \oplus P_{Byte}(i + 4)$. DPA can be applied to attack MAC such as the proposal from Underwrites Laboratories [9] and the minimal trace number needed is as Table (1).

Leakage Model	$P_{Byte}(0) \oplus P_{Byte}(4)$	$P_{Byte}(1) \oplus P_{Byte}(5)$
Trace Number	3800	2800
Leakage Model	$P_{Byte}(2) \oplus P_{Byte}(6)$	$P_{Byte}(3) \oplus P_{Byte}(7)$
Trace Number	3800	2600

Table 1: Trace number needed to perform UL attack

We can also build templates for key. More specifically, target on the $K_{Byte}(i) \oplus K_{Byte}(i + 4)\{0 \leq i \leq 3\}$ and build the template here. This allows us to reduce the entropy of K which later on allows to make our attack practical.

Having these ingredients we can focus on the main contribution of the DPA attacking using the leakage of $HW(C_{Byte}(i) \oplus C_{Byte}(i + 4))$.

4 New DPA attack method

In this section we will show that mounting a side-channel attack directly on the DES output (instead of on e.g. the S-box output) is not only more efficient than attacking TDES, but is even practical with the help of some additional information on key.

Let $Kset$ denote the search space we needed for the first DES and we have $Kset \leq 2^{56}$. Supposedly, N traces are acquired when N randomized plaintexts are applied to the chip, let $P_i\{1 \leq i \leq N\}$ and $C_i\{1 \leq i \leq N\}$ denote the plaintexts and cipher texts. For each $K \in Kset$, the corresponding cipher texts $C(P_i, K)$ can be computed accordingly. Let $Cset(K)$ be the cipher text sets specified as below

$$Cset(K) = \{C(P, K) | P = P_i, 1 \leq i \leq N\}, \{K \in Kset\} \quad (4)$$

For each specific K , the related correlation between the cipher texts and traces can be calculated. So the correct key will be distinguishable from other keys. Hence we have

$$Key = \arg \max_{K \in Kset} \rho(Cset(K), trace) \quad (5)$$

4.1 Practical attack results

Totally 1500 traces and 128 hypothesis of keys are involved in the coming results and figures of DPA. In next section, the total costs for this attack will be evaluated with details. the internal data is defined as the 32 bits MSB exclusive-or with 32 bits LSB of cipher text. The attack result is illustrated as Figure 3. In this figure, the correct key shows up a distinguishable correlation spike which indicate a successful attack.

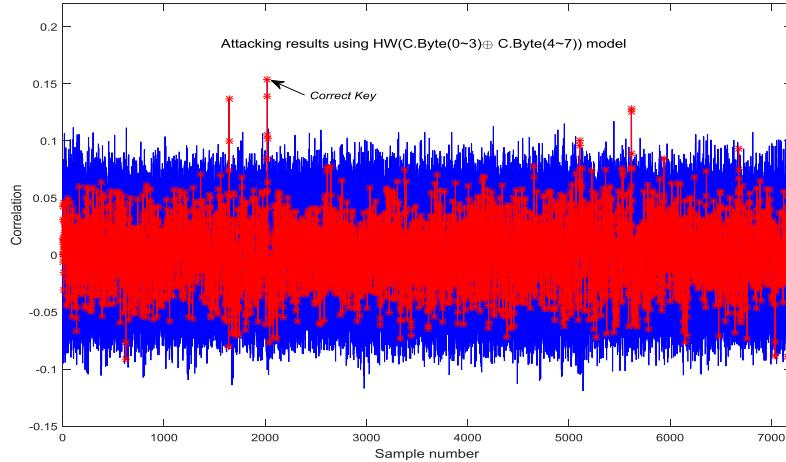


Figure 3: Attacking results using 1500 traces

When filter out the sample point 2021 out of the original traces, the correlation trend when increase the trace number can also be plotted as Figure 4,

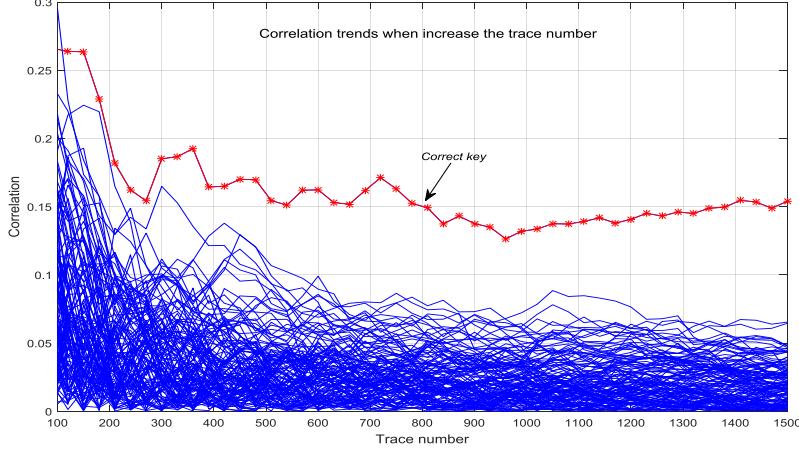


Figure 4: Correlation trends when increase trace number

Figure 4 indicates that the minimal trace number needed to perform a successful attack is less than 256. This is also reasonable because DES algorithm is a non-linear system and if key is not correct the cipher text can be regarded as random number and the correlation between the model and the trace will be approximately zero.

4.2 Costs evaluation

The cost for an attack can be defined as.

$$T = T_{acquisition} + T_{pre-computation} + T_{correlation-calculation} \quad (6)$$

The pre-computation indicates the internal data calculation such as the Hamming weight of output of SBOX. The third part means the costs for computing calculation between the internal data and the traces.

Typically, one million traces can be acquired within 12 hours. So 1k traces costs less than one minute.

For correlation calculation of two vectors in an Intel i7-5600U CPU(dual core with 4 threads) with Matlab, 10^6 times of correlations for two 10^3 elements vectors cost nearly 85 seconds. Let t_{des} denote the time cost of one DES encryption for brute force and $t_{des-cpu}$ be the time cost when using Intel i7-5600U CPU. Moreover, COPACABANA is capable of processing more

than 292 billion DES per second utilizing 128 FPGA cores[12], which indicates that $t_{des} \approx \frac{1}{292*2^{30}} \approx 2^{-38}$ second.

Moreover, Biham's research [8] indicates that a DES takes about 300 instructions. So we have

$$t_{des-cpu} \approx \frac{300}{2 * 3.2G} \approx 2^{-24} \text{ second} \quad (7)$$

Next we will set up some cost references to compare with this attack. The first scenario is for a single DES certification with one million traces. The total costs for this attack is

$$T_{regularDPA} = 12 \text{ hours} + 0 + 0 \approx 12 \text{ hour} \quad (8)$$

The second one is the meet-in-the-middle brute force attack. And a TDES costs 2 single DES.

$$T_{bruteforce} = 0 + 2^{112} * 2t_{des} + 0 \approx 2^{113}t_{des} \quad (9)$$

For this approach, totally 256 traces are needed and the key search space for a single DES is $2^k = \mathcal{K}_{set}$ and the costs are

$$\begin{aligned} T_{this} &= 1 \text{ min} + 2^k * 256t_{des-cpu} + \frac{2^k * 85}{10^6} \\ &\approx \frac{2^{28}t_{des-cpu} + 85}{2^{20-k}} \\ &\approx 101 * 2^{k-20} \text{ second} \end{aligned} \quad (10)$$

Compared with Eq. (9), this approach is apparently reasonable for a real attack, actually

$$\frac{T_{bruteforce}}{T_{this}} = \frac{2^{113}t_{des}}{101 * 2^{k-20}} \approx \frac{2^{113} * 2^{-38}}{101 * 2^{56-20}} \approx 2^{32.3} \quad (11)$$

The equation above means that this attack is 2^{32} faster than brute force. Additionally we have $101 * 2^{29-20} \approx 14.3 \text{ hour}$, so if the key search space are reduced to 2^{29} , then the efforts for this attack is similar to the regular DES certification. Moreover, this reduction of key space will be also possible if one can build template attack for the key. In particular, it turns out that in our case, if use the leakage for key in Eq. (3) and build template for $K_{Byte}(i) \oplus K_{Byte}(i+4)$, then only 28 bit key entropy will be remained. Then this attack only costs 7.2 hours to retrieve $K1$. Finally, by applying the same technique on all DES keys, a TDES key can be recovered in only 21.6 hours.

5 Conclusions

In this paper, we presented 4 types of leakages for a widely used commercial product which is based on the hardware platform that passed the EAL5+ certification. Moreover, since this chip uses the hard-macro design technology, the leakage would affect the other family members as well.

Using the leakages for plaintexts, we also applied an attack to the protocol MAC-ISO9797-1 standard and the trace number required is less than 16K.

Moreover, using the leakage of ciphertext, we applied a DPA to DES and it turns out that only 256 traces is required to perform a successful attack. After that the private key of TDES can be retrieved DES by DES.

Another contribution of this paper is that the effort and cost of this attack have also been evaluated. If the key search space for each DES can be reduced to 28 bits, then this attack can break TDES within 21.6 hours by using an Intel i7-5600U CPU, which means this approach can be much more powerful if combined with other kinds of attacks.

To protect products against this attack, the design needs to apply hardware countermeasures when retrieving ciphertext.

Acknowledgements

The authors would like to thank Dexter Feng, Lirui Liu and Gavin Yao for valuable comments.

References

- [1] P. C. Kocher, J. Jaffe, and B. Jun. "Differential Power Analysis: Leaking Secrets." International Cryptology Conference 1999.
- [2] M. L. Akkar, R. Bevan, P. Dischamp, D. Moyart. Power Analysis, What Is Now Possible. Advances in Cryptology, ASIACRYPT 2000. Springer Berlin Heidelberg.
- [3] T. S. Messerges, E. A. Dabbish, R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. IEEE Transactions on Computers, 51(5), 541-552.

- [4] E. Brier, C. Clavier, F. Olivier. Correlation Power Analysis with a Leakage Model. Cryptographic Hardware and Embedded Systems, CHES 2004. Springer Berlin Heidelberg.
- [5] National Institute of Standards and Technology (U.S.). Data Encryption Standard(DES). FIPS Publication 46-3, NIST, 1999. Available at <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>.
- [6] National Institute of Standards and Technology (U.S.). Advanced Encryption Standard (AES). Available at <http://csrc.nist.gov/publications/drafts/dfips-AES.pdf>.
- [7] M. L. Akkar, C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. International Workshop on Cryptographic Hardware and Embedded Systems (Vol.2162, pp.309-318). Springer-Verlag.
- [8] E. Biham. A fast new DES implementation in software[J]. Lecture Notes in Computer Science, 1999, 1267(1267):260-272.
- [9] B. Feix and H. Thiebeauld. Defeating ISO9797-1 MAC algo 3 by combining side-channel and brute force techniques. Cryptology ePrint Archive, Report 2014/702, 2014.
- [10] W. Diffie. Exhaustive Cryptanalysis of The NBS Data Encryption Standard[J]. 1977.
- [11] E. Oswald, S. Mangard, N. Pramstaller, et al. A Side-Channel Analysis Resistant Description of the AES S-Box[C]. International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers. 2005:413-423.
- [12] Break DES in less than a single day [Press release of Firm, demonstrated on 2009 Workshop]