Defeating Embedded Cryptographic Protocols by Combining Second-Order with Brute Force

Benoit Feix(✉), Andjy Ricart, Benjamin Timon, and Lucille Tordella

UL Transaction Security Lab, Basingstoke, England
{benoit.feix,andjy.ricart,benjamin.timon,lucille.tordella}@ul.com

Abstract. Side-channel analysis is a well-known and efficient hardware technique to recover embedded secrets in microprocessors. Countermeasures relying on random masking have been proven to be sound protections against such threats and are usually added to protect sensitive intermediate data during the algorithm process. However, Second-Order Side-Channel Analysis have proven to allow secret key recovery in the presence of random masking. In [4] an attack was introduced which exploits the information exchange at the cryptographic protocol level in order to disclose the secret key of the ISO/IEC 9797-1 MAC algorithm 3 using DES operations. A countermeasure suggestion was for a mask to be applied at the protocol level in order to protect all secret data. This paper extends the attack idea previously published to second order attacks on masked implementations of the ISO/IEC 9797-1 MAC algorithm 3 and shows that securing against such attacks must be done with care.

Keywords: Side-channel analysis · DES · MAC ISO/IEC 9797-1 · HODPA · Masking · Exhaustive search

1 Introduction

Since its first introduction in 1996 by Kocher [10] side-channel analysis has proved to be a powerful threat for embedded cryptosystems. The fast paced growth of published attacks using side-channel techniques challenge cryptographic algorithms designers to constantly counteract these attacks.

While many countermeasures already exists to protect cryptographic algorithms against those attacks, [4] showed that these protections shall not be confined to the cryptographic layer only but shall be extended to the cryptographic protocol level. Indeed the authors presented an attack targeting the ISO9797-1 MAC Algorithm 3 [8] protocol mostly used on secure elements. The attack uses the fact that for this protocol, a data usually considered as a public value (the resulting ciphertext of a DES operation) could lead to the recovery of the 112-bits secret key of the algorithm combining statistical side-channel and brute-force attacks.

In [4], the authors suggested to prevent the intermediate information between the DES operation to appear in plain in order to withstand the presented attack.

K. Lemke-Rust and M. Tunstall (Eds.): CARDIS 2016, LNCS 10146, pp. 23–38, 2017.
DOI: 10.1007/978-3-319-54669-8_2

This paper presents how, in some attack scenarios, an implementation of the ISO9797-1 MAC Algorithm 3 [8] protocol secured with boolean masking could still be vulnerable to higher order side-channel analysis.

The paper is organized as follows. Section 2 reminds the structure of the ISO9797-1 MAC algorithm 3. The necessary knowledge and background on side-channel analysis to understand our second order attacks are also presented. In Sect. 3 describes our work to recover the secret keys of masked MAC computations. The practical results obtained are then presented in Sect. 4. In Sect. 5 we provide recent status on DES crackers capabilities and Sect. 6 is a discussion of the classical countermeasures and their efficiency to prevent our attack. Finally we conclude in Sect. 7.

2 Preliminaries

2.1 Side-Channel Analysis Background

Side-channel analysis has been studied for years since it has been introduced by Kocher [10]. Many attack paths have been published on the different cryptosystems like DES [5] and RSA [15] which are widely used in the majority of the hardware embedded devices like Banking or Identity products. In the same time many statistical attack techniques have improved the original Differential Side-Channel Analysis (DSCA) (i.e. Difference of Mean - DoM) from Kocher et al. [11]. We can for instance mention the Correlation Side-Channel Analysis (CSCA) introduced by Brier et al. [2], the Mutual Information Side-Channel Analysis (MISCA) from Gierlichs et al. [7] or the Linear Regression Side-Channel Analysis [3,16].

Second Order Side-Channel Attacks. Second order Side-channel analysis is an attack category allowing to recover a secret data in the presence of a first order countermeasure (i.e. where classical first-order side-channel attacks are failing). It has been originally introduced by Messerges [13]. For years now second-order attacks are mainly focused on attacks relying on the Hamming Weight.

When a boolean masking is implemented as a countermeasure, the intermediate data is replaced by $v_K^* = v_K \oplus M$ with $M \in \mathbb{F}_2^n$ and n being the bit length of v_K. By doing so, the sensitive data is no longer linked to the side-channel leakage. But it is still possible to succeed the attack on such an implementation with the knowledge of the mask side-channel emanations as explained in [9,14,17].

The core idea of SODPA is that if the attacker knows (or guesses) the time frames T_M and T_v where the mask and the masked sensitive data are respectively manipulated, he can combined them in order to decrease the mask efficiency. Both the guess on the intermediate values and the traces during T_M and T_v have to be combined.

The intermediate values v_K^* and M can be combined using $S = HW(v_K^* \oplus M)$.

A study of combination methods for the traces has been conducted in [14] by comparing mainly the *Absolute Difference* processing traces such as $T = |T_M - T_v|$ and the *Multiplicative Combining* processing traces such as $T = (T_M - \mu_M) * (T_v - \mu_v)$ with μ_M and μ_v the mean values of the sets of traces T_M and T_v. It concluded the latest is the most efficient.

A correlation between S and the combined signal would then allow to recover K.

2.2 Background on ISO 9797-1 MAC Algo 3

The ISO/IEC-9797-1 MAC Algorithm 3 with DES (also known as retail MAC) computes the MAC of data using a 112-bit secret key. For a k 8-byte blocks data, the $k-1$ first blocks are processed through Single DES operations in CBC mode using 56 bits of the key and a triple DES is performed on the final block using a 112 bits secret key and the result of the processing of the $k-1$ first blocks as IV. The security of this protocol relies on the Triple DES but the use of a CBC-DES chain allow a reasonable computation time. Figure 1 describes the MAC algorithm 3 when the encryption algorithm selected is the standard DES.

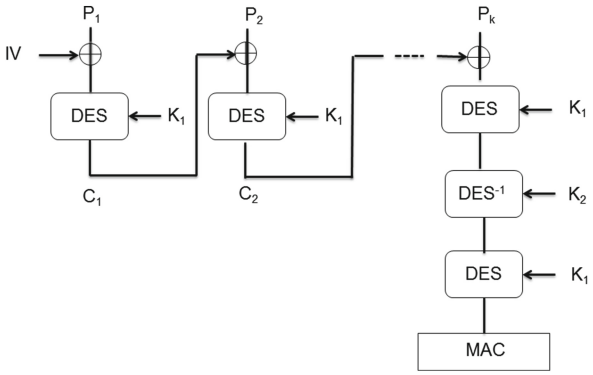

Fig. 1. ISO 9797-1 MAC Algo 3 Description with DES cipher

In the following we define K as the 112 bits key used for the calculation, divided into two subkeys K_1 and K_2 of equal length such as $K = (K_1, K_2)$. The i^{th} occurrence of the algorithm $\text{MAC}^{(i)}$ is the result value CBC-MAC-Algo3$(P_1^{(i)} P_2^{(i)} \dots P_k^{(i)})$ with $IV = 0$ and the n^{th} ciphertext is defined as $C_n^{(i)} = \text{DES}_{K_1}(P_{n-1}^{(i)} \oplus C_{n-2}^{(i)})$[1].

Side-Channel Attack on MAC Algo 3. The attack introduced in [4] targets the 112 bit-length master key used by the ISO 9797-1 MAC algorithm. The result of such an attack could be the disclosure of the whole master key. In the following, the different steps for implementing this attack are reminded.

[1] $\oplus$ represents the bitwise exclusive OR operation.

The first step of the attack is to recover one couple of value (plaintext P, ciphertext C) from a single DES computation from a set of ℓ executions of the ISO9797-1 MAC algorithm 3 using side channel traces collected from the targeted device during the MAC Algorithm 3 computation.

In order to recover the n^{th} block intermediate result, C_n has to be a fixed value. Hence, the n first 8-byte long blocks value $P_1^{(i)}, \ldots, P_n^{(i)}$ must be set to a constant value $P = P_1, \ldots, P_n$, identical for all of the ℓ MAC computation $MAC^{(1)}, \ldots, MAC^{(\ell)}$. The $n + 1^{th}$ block $P_{n+1}^{(i)}$ must then be chosen randomly or at least with a high entropy across the transaction set.

As the input $P_{n+1}^{(i)}$ is a known random value, it is possible to guess the value $D^{(i)} = C_n \oplus P_{n+1}^{(i)}$ from side channel traces using statistical attacks and to recover C_n.

Once recovered, a brute force attack can then be conducted to deduce the value of the 56-bit key K_1 from the couple (P_n, C_n), which in turn allows the recovery of K_2 with some more effort.

In [4] masking the operations between each DES operations of the MAC Algorithm 3 is proposed as a countermeasure against such an attack. This paper will however show that high order attacks could still be possible in some cases and that therefore care should be taken when implementing such a masking scheme.

In the following section we use the second order attack as well as the results from [4] to target a masked implementation of the ISO9797-1 MAC Algorithm 3.

3 Second Order Side-Channel Attack on the MAC Algorithm 3

As previously mentioned, masking the operations between each DES operation of a MAC Algorithm 3 should be done with care. A second order attack on two different masked implementations of a MAC Algorithm 3 will be presented in this section.

3.1 All DES Inputs Masked with the Same Mask

The first masking scheme is presented on Fig. 2. While this implementation is secure against the first order attack presented in [4], should the same mask M be used for all the DES of the same transaction it would then be possible to mount a second order attack against this implementation by combining two DES inputs as explained in Sect. 2.1.

Indeed, considering $P_n \oplus M$ the n^{th} DES input and $C_n \oplus P_{n+1} \oplus M$ the $(n + 1)^{th}$ DES input, it is possible to perform a second order CPA in order to retrieve C_n.

As for the first order attack, a set of side channel traces focusing on the execution of both the first and second DES operations is collected with $P_0 \ldots P_n$ fixed and P_{n+1} randomized.

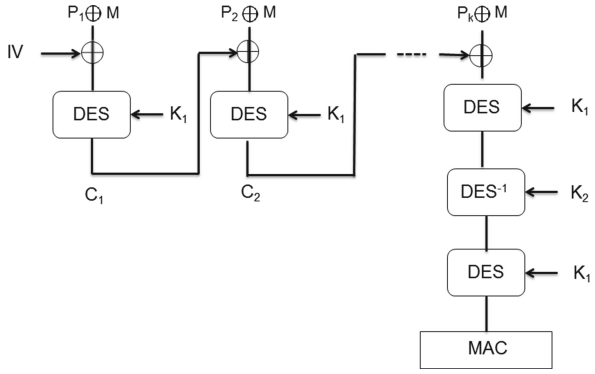

Fig. 2. ISO 9797-1 MAC Algo 3 Description with DES cipher

To cancel the unknown mask value, the first and second DES inputs can be combined as follows: $(P_n \oplus M) \oplus (C_n \oplus P_{n+1} \oplus M) = C_n \oplus P_n \oplus P_{n+1}$.

Therefore the two areas of the side channel traces corresponding to the two DES inputs processing, from the same transaction i, have to be combined. For $T_1^{(i)}$ and $T_2^{(i)}$ corresponding to these two parts of the signal $D_1 = (P_n \oplus M)$ and $D_2 = C_n \oplus P_{n+1} \oplus M$, any combination function $F(T_1, T_2)$ can be used as explained in [14].

A correlation between $D_1 \oplus D_2$ and the combined signal $F(T_1, T_2)$ would then enable the recovery of C_n which would then lead to the retrieval of keys K_1 and K_2 using brute force technique. It can be highlighted that if the value of P_n is fixed to 0, then the targeted intermediate state is the same as for the first order attack: $C_n \oplus P_{n+1}$. In this case, the mask is directly manipulated by the device during the data loading. A legitimate question is to know if fixing P_{n+1} to 0 improves the attack. This question will be studied in Sect. 4.

3.2 All DES Inputs Masked with Different Masks, Mask Is only 1 Byte

In the case of each block being masked with a different value, as presented in Fig. 3, the previous attack cannot be applied. In the specific case of these masks being only 8-bit masks (i.e. for the n^{th} block, $M_n = (R_n|R_n|\ldots|R_n)$ with R_n an 8-bit random) then a more complex attack can still reveal the secret key.

Indeed in this case the input of one DES operation can be combined with itself to cancel the mask as such: $(C_{n,j} \oplus P_{n,j} \oplus R_n) \oplus (C_{n,j+1} \oplus P_{n,j+1} \oplus R_n) = C_{n,j} \oplus P_{n,j} \oplus C_{n,j+1} \oplus P_{n,j+1}$ (for each byte j of the ciphertext). As for the previous attack, the two parts of the signal T_1 and T_2 corresponding to the processing of respectively $D_1 = (C_{n,j} \oplus P_{n,j} \oplus R_n)$ and $D_2 = (C_{n,j+1} \oplus P_{n,j+1} \oplus R_n)$ are combined using the function $F(T_1, T_2)$.

In order to compute the value $D_1 \oplus D_2$ the attacker has to guess the value of $C_{n,1}$ and $C_{n,2}$, therefore the complexity of the correlation attack will be higher

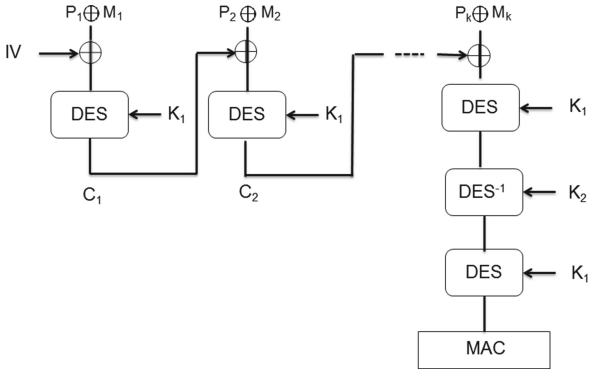

Fig. 3. ISO 9797-1 MAC Algo 3 Description with DES cipher

than the attack presented earlier. If successful, this attack would then enable the recovery of $C_{n,j} \oplus C_{n,j+1}$. In order to recover the ciphertext C_n, the attacker would still have a 1-byte uncertainty due to the XOR operation between the bytes. Although the complexity of the brute force key recovery on the MAC Algorithm 3 would be higher, the effort would remain affordable as explained in Sect. 5.

4 Practical Results

Several practical experimentations were conducted to assert the feasibility of the attacks we proposed in Sect. 3. Tests were first conducted on simulated power traces relying on a standard Hamming weight leakage model with a white Gaussian noise. An ISO9797-1 MAC algorithm 3 implemented on a 32-bit ATMega microprocessor was then targeted, the tests focusing on the electromagnetic emanations of the device collected during several MAC computations. All of the results are presented in this section.

4.1 Simulated Traces

Tests on Fixed Mask. Simulations were performed in order to validate the proposed attack path. Another goal of the simulations was to determine whether or not, the choice of the fixed value P_n (fixed to 0 or not) can influence the result of the attack. We chose to attack the 1^{st} DES output C_1. To do so, we use the classic power model $L(x) = HW(x) + N(\alpha, \sigma)$, where $HW(x)$ refers to the Hamming weight of the value x, and N is a Gaussian noise of mean α and standard deviation[2] σ. We simulated the attack on an 8-bit architecture.

[2] Regarding the standard deviation of the noise, a unit corresponds to the side-channel difference related to a one bit difference in the Hamming weight.

Simulation Protocol. Draw an 8-byte random P_1 to simulate the first fixed data block and an 8-byte random K_1 the DES key used in Fig. 1. For each simulated trace proceed as follows:

- Draw an 8-byte random M as mask value,
- Compute $D_1^{(i)} = P_1^{(i)} \oplus M^{(i)}$ and $C_1 = \text{DES}_{K_1}(P_1)$,
- Draw an 8-byte random $P_2^{(i)}$ to as random second data block,
- Compute the next masked value $P_2^{(i)} \oplus M^{(i)}$,
- Compute the next DES operation input $D_2^{(i)} = C_1 \oplus P_2^{(i)} \oplus M^{(i)}$,
- Compute the combination for the second order attack $D_1^{(i)} \oplus D_2^{(i)}$.

The power consumption of the described sequence was simulated using the power model previously defined. The DES operation was not included in the simulation to avoid any risk of leakage from this process. The second order attack as previously proposed was implemented and tested. Figures 4 and 5 show the attack success on the bytes 3 and 6 of C_1 to illustrate the results (all bytes showed similar results).

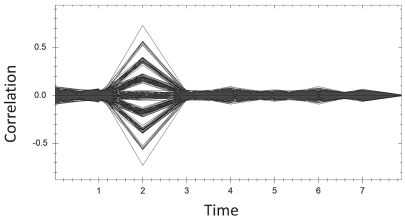

Fig. 4. 2^8 correlation traces attack result on byte 3 of C_1 for $\ell = 100$ traces and $\sigma = 1$

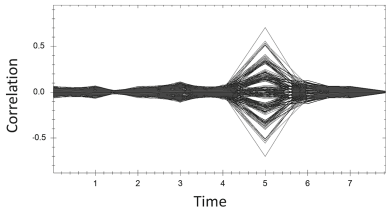

Fig. 5. 2^8 correlation traces attack result on byte 6 of C_1 for $\ell = 100$ traces and $\sigma = 1$

The success rate of our attack technique was assessed by performing 100 experiments as described earlier. The tests have been performed for different noise standard deviation values, from $\sigma = 0.1$ to $\sigma = 7$. The results are presented in Fig. 6. These tests confirmed in practice the relevance of the theoretical attack presented in Sects. 4.2 and 4.3.

Next, experiments were performed in order to conclude about the influence of P_1 on the results. First, the attack was performed with P_1 fixed to 0 and then with P_1 fixed to a random non-null value. The result of these attacks are presented in Fig. 7.

As it can be observed, the correlation levels are similar in both cases, which means that the chosen value of the fixed data P_1 does not influence the result of the attack.

Tests on 1-Byte Random Masks. The same power model was kept for the simulation $L(x) = HW(x) + N(\alpha, \sigma)$. Here, the attack was simulated on an 8-bit architecture.

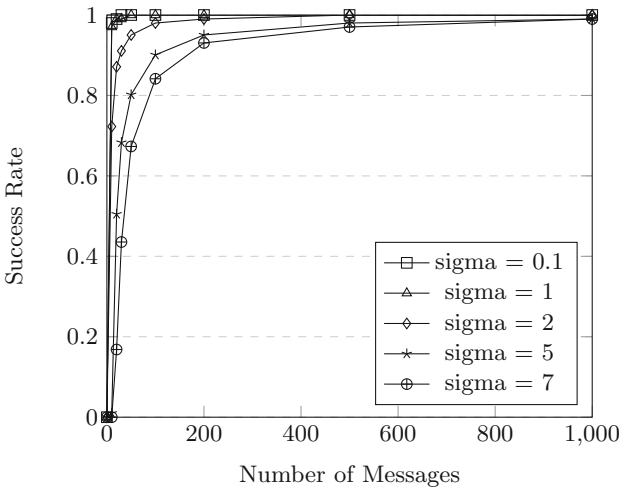

Fig. 6. Attack success rate for different noise value - Recover the full ciphertext

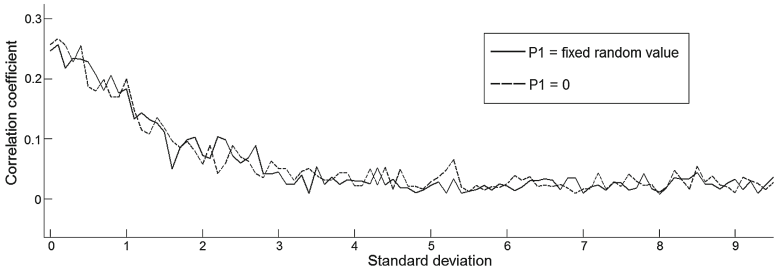

Fig. 7. Comparison of the attack efficiency depending on the value of P1.

Simulation Protocol. Draw an 8-byte random P_1 to simulate the first fixed data block and an 8-byte random K_1 the DES key used for the CBC DES chain of the protocol. For each trace simulation proceed as follows:

- Draw a 1-byte random $M_1^{(i)}$ as mask value,
- Compute $D_1^{(i)} = P_1^{(i)} \oplus M_1^{(i)}$ and $C_1 = \mathrm{DES}_{K_1}(P_1)$,
- Draw an 8-byte random $P_2^{(i)}$ as random second data block,
- Draw a new 1-byte random $M_2^{(i)}$ to simulate the new mask value,
- Compute the next masked value $P_2^{(i)} \oplus M_2^{(i)}$,
- Compute the next DES operation input $D_2^{(i)} = C_1 \oplus P_2^{(i)} \oplus M_2^{(i)}$,
- Compute the combination for the second order attack $D_{2,j}^{(i)} \oplus D_{2,j+1}^{(i)}$ (for each byte j of the ciphertext).

The power consumption of the described sequence was simulated using the power model previously defined. The DES operation was not included in the

simulation to avoid any risk of leakage from this process. The second order attack as previously proposed was implemented and tested. Figures 8 and 9 show the attack success on some of the bytes of C_1 to illustrate the results (all the bytes showed similar results).

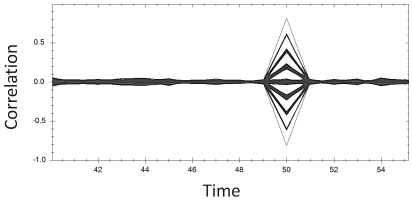

Fig. 8. 2^8 correlation traces attack result on $C_{1,3} \oplus C_{1,4}$ for $\ell = 100$ traces and $\sigma = 1$

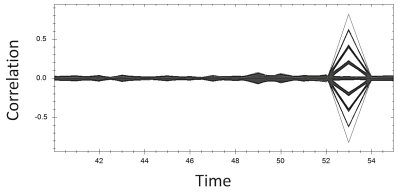

Fig. 9. 2^8 correlation traces attack result on $C_{1,6} \oplus C_{1,7}$ for $\ell = 100$ traces and $\sigma = 1$

The success rate of our attack technique was assessed by performing 100 experiments as described earlier. The tests have been performed for different noise standard deviation values, from $\sigma = 0.1$ to $\sigma = 7$. The results are presented in Fig. 10.

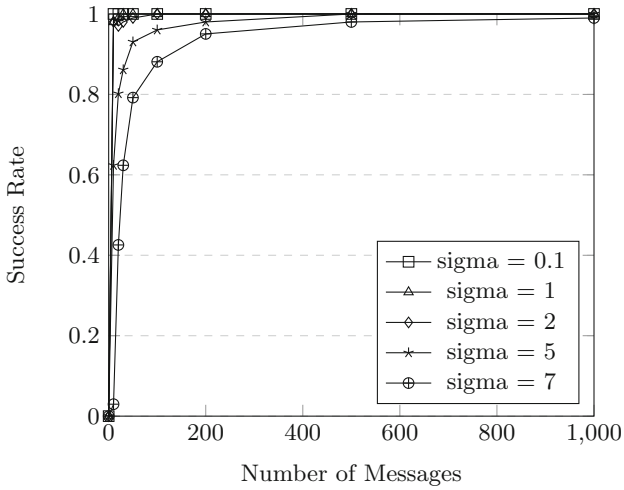

Fig. 10. Attack success rate for different noise value

4.2 Tests on an 32-Bit AT-Mega Microprocessor

The single DES operations of an ISO9797-1 MAC algorithm 3 were implemented on a 32-bit Arduino board. Different tests were conducted to validate the previous attacks presented on a real device as well as to corroborate the results

computed from the simulation traces. First, an unmasked implementation of the MAC Algorithm 3 was targeted to reproduce the results of [4]. Tests on a masked implementation of the MAC Algorithm 3 were then conducted. The results of the different test are presented in the following.

First Order on Unmasked Implementation. An unmasked implementation of the MAC Algorithm 3 was first targeted. Around 10,000 Electromagnetic traces related to the first two DES operations were collected, with P1 fixed and P2 randomised to follow the attack presented in [4]. The area of interest is shown in Fig. 11.

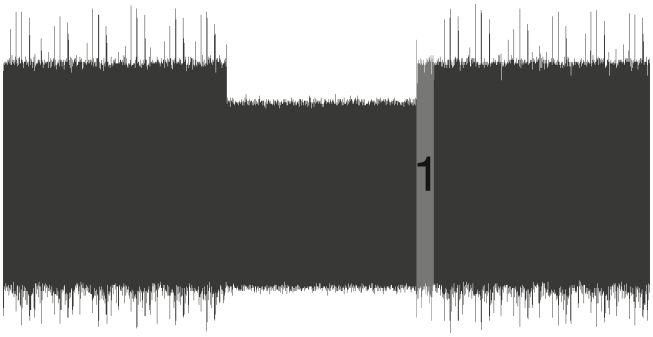

Fig. 11. Area or interest - $P_2 \oplus C_1$ (area 1)

Less than 130 traces were needed to recover all 8 bytes of the targeted ciphertext. As an example of results, Fig. 12 shows the results for the fourth byte. The results for the other bytes were similar.

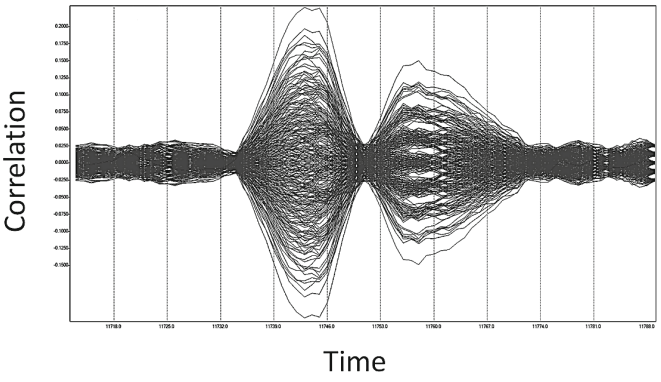

Fig. 12. Correlation peak on byte 4 of C_1 for the First Order attack

The detailed summary of the number of traces needed to recover each of the bytes of the targeted ciphertext is presented in Table 1.

Table 1. Number of traces needed to recover the targeted bytes, first order attack on an unmasked implementation

Targeted byte	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Traces needed	30	70	50	90	70	90	130	90

Tests on a Masked Implementation. Tests were then conducted on a masked implementation of the MAC Algorithm 3. For these tests, a 8-byte mask $M^{(i)}$ was randomly generated for each $MAC^{(i)}$ computation and applied to each DES operation in order to conceal the XOR operations $D_{k+1}^{(i)} = C_k \oplus P_{k+1}^{(i)}$ and prevent the recovery of C_k with first order statistical attacks. It is to be noted that the DES operations themselves were modified following the scheme proposed in [1] to ensure that the input of each DES would remain masked during the entire computation and prevent any leakage from the DES internal computations.

100,000 traces were collected, with the input block P_1 fixed to zero. A first order attack was conducted focusing on the area of leakage previously identified in order to ensure that the masking scheme is efficient enough to prevent any first order leakage. The correlation traces corresponding to the bytes of the targeted ciphertext showed that no peak could be obtained of the previously leaking area thus proving that the implemented masking scheme is sufficient to prevent such an attack.

The leakage areas T_1 and T_2 corresponding respectively to $D_1 = (P_1 \oplus M)$ and $D_2 = (P_2 \oplus M \oplus C_1)$ were identified on the traces as shown on Fig. 13.

Fig. 13. Areas of interest - $D_1 = P_1 \oplus M$ (area 1) and $D_2 = P_2 \oplus M \oplus C_1$ (area 2)

The previously explained second order attack was firstly performed by combining T_1 and T_2 such as $T_{sec} = \|T_2 - T_1\|$ with T_{sec} the traces targeted by the attack. Correlation peaks allowed the recovery of each bytes of the targeted value, Fig. 14 shows the example of the fourth byte.

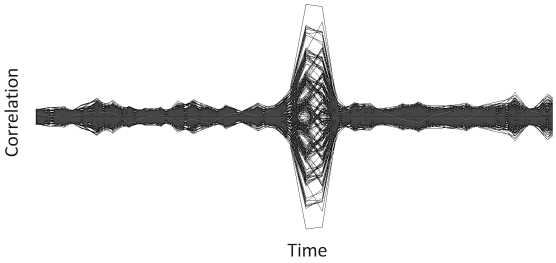

Fig. 14. Correlation peak on byte 4 of C_1 on traces using *Absolute Difference*

Table 2. Number of traces needed to recover the targeted bytes, second order attack with $P_1 = 0$ using *Absolute Difference*

Targeted byte	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Traces needed	3,500	34,000	7,100	4,000	6,800	8,000	4,200	5,300

This attack enabled the recovery of the ciphertext with less than 34000 traces. Table 2 shows the number of traces needed to recover each bytes of the ciphertext.

The exact same attack was performed but combining T_1 and T_2 such as $T_{sec} = (T_2 - \mu_{T_2}) * (T_1 - \mu_{T_1})$ with T_{sec} the traces targeted by the attack. Again, correlation peaks allowed the recovery of each bytes of the targeted value, Fig. 15 shows the example of the fourth byte.

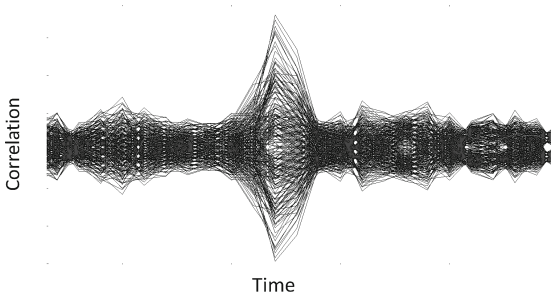

Fig. 15. Correlation peak on byte 4 of C_1 on traces using *Multiplicative Combining*

This attack enabled the recovery of the ciphertext with less than 8500 traces. Table 3 shows the number of traces needed to recover each bytes of the ciphertext.

These results prove that the attacks we presented in Sect. 3 can be a threat to some masked MAC Algorithm 3 computations. The following section will present possible countermeasures against those second order attacks.

Table 3. Number of traces needed to recover the targeted bytes, second order attack with $P_1 = 0$ using *Multiplicative Combining*

Targeted byte	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Traces needed	1,400	1,000	8,500	4,200	3,000	4,900	4,000	4,800

4.3 Tests on a Secure Integrated Circuit from the Industry

This attack was also performed this attack on a more secure integrated circuit than a commercial Arduino. Indeed it embeds several hardware security mechanisms. The objective was to measure the effectiveness of this attack path on a state-of-the-art security product. The same second order attack has been conducted with success on this device, with P_1 fixed to zero. In this case it was also possible to recover all bytes of C_1 with less than $65,535$ traces which is the maximum number of traces available for such a protocol for some banking products.

5 Follow-Up: How to Complete the Attack - The Efficiency of DES Crackers

Once we have recovered with side-channel analysis the couple of values (P_n, C_n), it is required to perform a brute force attack to recover the first single DES 56-bit key K_1 of the MAC algorithm 3. Today performing such a single DES brute force can be achieved for a low cost in a short time period: from few hours to 1 week depending on the solution chosen.

It can be performed using a dedicated hardware such as the Rivyera engine sold by the SciEngines company based on the COPACABANA DES cracker [12] presented initially at the CHES 2009 conference. Using such an engine enable the recovery of a single DES key in 1–2 days. Once this secret key K_1 is recovered a second brute force is performed to recover K_2.

The attack presented here requires the value P to be fixed to a unique constant value (e.g. plaintext $P = 0$) that can be chosen or at least is known by the attacker. Therefore it is also possible to use DES rainbow table - constructed on this given plaintext value - to perform the DES brute force operation.

In 2015 two students and their professor from Limoges University succeeded to build a DES rainbow table in less than 1 month using GPU from the university [6]. Once the DES rainbow table is available (128 Gigabyte in this case) only few days on a computer are necessary to recover with high probability the secret key. This example shows that no expensive or complicated hardware is necessary. Indeed a student can use a classroom and rainbow tables found on the internet and brute force a single DES key in only a few days.

More recently it has been published on twitter it is possible to brute force a DES in less than 60 hours for ∼5000$ using hashcat.

6 Countermeasures

Session keys: a first protection to consider relies on the protocol level design. Using session keys instead of a master key when computing ISO9797-1 MAC Algorithm 3 cryptograms, the session keys being regenerated for each transaction, is still considered as the strongest countermeasure against the first and second (and higher) order attacks presented here.

Enhanced random masking: the classical technique consisting in using several random mask is still efficient if properly designed. Indeed if a masking scheme similar to the one presented in Sect. 3 is used to protect the MAC Algorithm 3 against first order attacks, the developer should make sure that each DES block are masked using a different mask, even if this requires more resources. If the RNG used to generate the different masks is not biased, it would then insure protection against first and second order attacks.

Execution counter: as a second order attack on the MAC Algorithm 3 might require more traces than a first order attack to be successful, limiting the number of MAC computations can still be considered as a temporary solution against this attack. However, our experiments showed that this cannot be always considered enough to protect an implementation against this attack. Tests should therefore still be done to ensure that the limit counter chosen would be sufficient. However improving the attack efficiency in the future could still defeat this light protection.

Signal desynchronization: adding noise to the signal in order to lower the signal to noise ratio should also reduce the efficiency of any attack, as for the first order but that should not be considered enough and needs to be combined with another security measure.

7 Conclusion

In this paper we have explained how much it can be difficult to prevent some cryptographic protocols (i.e. MAC algorithm 3) implementation from state-of-the-art side-channel attacks. Indeed we have presented successful practical second order attacks that can defeat masked implementations of the MAC algorithm 3 designed to be resistant to the original side-channel attack presented in 2014 by Feix et al. These new results highlight the serious need for strong high-order side-channel resistant countermeasures at the protocol level and not only during the DES cryptographic operations. Hence random masks applied on DES inputs and outputs - as well as during the XOR operations - must be updated very regularly. It requires developers to apply random mask values from the beginning of the CBC chain to the end of the MAC computation by refreshing the random masks for each plaintext and ciphertext block manipulated. Nowadays the efficiency of high-order side-channel attacks combined with the fact it has

become quite simple to access DES cracker computations make such attacks very realistic and practical on secure products. It is then very important to embed strong countermeasures when performing MAC algorithm 3 computations with master keys.

Acknowledgements. The authors would like to thank Loic Thierry for the fruitful discussions we had on this subject in the past.

References

1. Akkar, M.-L., Giraud, C.: An implementation of DES and AES, secure against some attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001). doi:10.1007/3-540-44709-1_26
2. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28632-5_2
3. Doget, J., Prouff, E., Rivain, M., Standaert, F.-X.: Univariate side channel attacks and leakage modeling. IACR Cryptology ePrint Archive, 2011:302 (2011)
4. Feix, B., Thiebeauld, H.: Defeating ISO9797-1 MAC Algo 3 by Combining Side-Channel and Brute Force Techniques. Cryptology ePrint Archive, Report 2014/702 (2014)
5. Federal Information Processing Standards Publication (FIPS). Data Encryption Standard - DES, FIPS PUB 46-3 (1999)
6. Fournier, R.: Implementation et Evaluation des Attaques par Brute-Force et par Analyse de Canaux Auxiliaires sur des Algorithmes Cryptographiques l'Aide de Processeurs Graphiques. Intership Limoges University (2015)
7. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85053-3_27
8. ISO/IEC. Information technology - Security techniques - Message Authentication Codes (MACs). ISO/IEC Standards (1999)
9. Joye, M., Paillier, P., Schoenmakers, B.: On second-order differential power analysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 293–308. Springer, Heidelberg (2005). doi:10.1007/11545262_22
10. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). doi:10.1007/3-540-68697-5_9
11. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_25
12. Kumar, S., Paar, C., Pelzl, J., Pfeiffer, G., Schimmler, M.: Breaking ciphers with COPACOBANA –a cost-optimized parallel code breaker. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 101–118. Springer, Heidelberg (2006). doi:10.1007/11894063_9
13. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000). doi:10.1007/3-540-44499-8_19
14. Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)

15. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**, 120–126 (1978)
16. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). doi:10.1007/11545262_3
17. Waddle, J., Wagner, D.: Towards efficient second-order power analysis. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28632-5_1

http://www.springer.com/978-3-319-54668-1

Smart Card Research and Advanced Applications
15th International Conference, CARDIS 2016, Cannes, France, November 7–9, 2016, Revised Selected Papers
Lemke-Rust, K.; Tunstall, M. (Eds.)
2017, XII, 265 p. 93 illus., Softcover
ISBN: 978-3-319-54668-1