Analysis of the Split Mask Countermeasure for Embedded Systems

Jean-Sébastien Coron Ilya Kizhvatov

Université du Luxembourg
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg
{jean-sebastien.coron, ilya.kizhvatov}@uni.lu

ABSTRACT

We analyze a countermeasure against differential power and electromagnetic attacks that was recently introduced under the name of split mask. We show a general weakness of the split mask countermeasure that makes standard DPA attacks with a full key recovery applicable to masked AES and DES implementations. Complexity of the attacks is the same as for unmasked implementations. We implement the most efficient attack on an 8-bit AVR microcontroller. We also show that the strengthened variant of the countermeasure is susceptible to a second order DPA attack independently of the number of used mask tables.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Code Breaking; Standards (AES, DES); C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; Smartcards

General Terms

Security, Algorithms, Measurement, Experimentation

Keywords

cryptanalysis, countermeasures, DPA, masking, side channel analysis

1. INTRODUCTION

Masking intermediate values of a cryptographic algorithm with random data is a widely adopted method for reinforcing its implementation in an embedded device against side-channel attacks. Originally introduced into the scientific community by Goubin and Patarin [14] and Chari *et al.* [6], masking makes the side-channel leakage of a device running the implementation independent of the processed sensitive data at individual moments. It is an effective way of countering the differential power analysis introduced by Kocher *et al.* [17]. Using multiple masks for a single intermediate variable can also counter higher order DPA attacks, first described in detail by Messerges [19].

Masking however notably increases implementation footprint and decreases performance, as additional variables are introduced into computations. To maintain proper data processing one must keep track of these variables (masks) during the algorithm execution. This involves precomputations of intermediate masking values from a set of randomly generated masks. Specifically, masked lookup tables must be recomputed when a mask changes. Results of profiling a masked AES implementation reported by Mangard *et al.* [18] indicate that most part of the introduced overhead originates from the precomputations.

Hence, various resource-saving masking schemes for constrained devices are of great practical interest. Ways of reducing the amount of precomputations and memory while keeping resistance to higher order DPA are a target of recent research. For example, Itoh et al. [16] suggested using a limited set of masks, so that all possible masked lookup tables can be precomputed once and stored in memory, thus avoiding recomputations at all.

The split mask countermeasure analyzed in this work has been proposed by Gebotys *et al.* in [10, 11, 12, 13]. The general idea of this variant of resource-saving masking is to mask each entry of a lookup table with a different output mask. These output masks are split into several shares that are stored in additional mask tables. The input of the tables is masked with a value that is fixed for a single key. Recomputation of the tables under a single key is either not performed at all or involves simple operations for modifying the output masks. The countermeasure is applied to AES [8] and DES [9] and is claimed to thwart DPA attacks[1] of order N when N mask tables are used.

We show that the approach followed in the split mask countermeasure exhibits a general weakness. By exploiting this weakness, AES and DES implementations with the split mask countermeasure can be defeated by a standard DPA leading to a full key recovery in the known plain- or ciphertext model. Several attack paths are possible. The complexity of the attacks is the same as for unprotected implementations. We implement the most efficient attack on an AES implementation with the split mask countermeasure for an 8-bit Atmel ATmega16 microcontroller form the widely-used AVR family. The attack requires about 400 traces for the

WESS '09, October 15, 2009, Grenoble, France

[1]Throughout this paper, we use the term DPA also with the electromagnetic side-channel in mind.

full key recovery.

In [11] it was mentioned that the split mask countermeasure can be strengthened by changing table input masks after a number of encryptions. We sketch a DPA attack of order 2 against this strengthened variant that works for any number N of mask tables.

2. THE SPLIT MASK COUNTERMEASURE

We outline the general idea of the split mask by showing how a single lookup table is masked based on the description in the recent work of Gebotys [11].

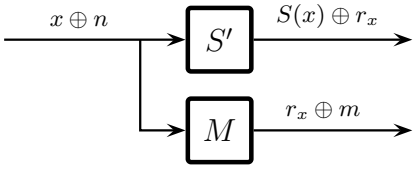

Figure 1: S-box with the split mask countermeasure

Let S be an S-box with input x and output $S(x)$ implemented as a lookup table. The split mask implementation of S, shown in Figure 1, consists of a masked table S' and a mask table M. These tables are defined as follows:

$$\begin{array}{rcl} S'(x \oplus n) & = & S(x) \oplus r_x, \\ M(x \oplus n) & = & r_x \oplus m. \end{array} \quad (1)$$

This means that the input of the S-box is masked with n, and each output value is masked with an individual random value r_x. This gives the masked table S'. The set of output masks r_x is also stored in the mask table M so that

$$S'(x \oplus n) \oplus M(x \oplus n) = S(x) \oplus m \quad (2)$$

holds for every input x. In other words, m can be viewed as the output mask of S that is split into two shares r_x and $M(x \oplus n)$, the splitting being individual for each table entry.

Masks m and n are fixed: they are randomlsy generated when tables (1) are precomputed for a given encryption key and then remain constant for different plaintexts. The set of output masks r_x is also randomly generated during table precomputation and can be refreshed for different plaintexts, as the table recomputations in this case are simple and fast. Thus, the performance overhead introduced by the countermeasure is small. Memory overhead can also be small if a single masked implementation is shared by several identical S-boxes of the algorithm, which is allowed by the original description of the split mask.

The split mask countermeasure with a single mask table is claimed to thwart the 1-st order DPA attack. For this, the original description requires that (2) should never be computed directly (*i.e.* appear as an intermediate value) during an algorithm execution. A generalization of the method uses N mask tables for splitting m into $N+1$ shares. In this case a DPA of order N is claimed to be thwarted under a similar condition.

In concrete AES and DES implementations the fixed masks at the input of the S-boxes originate from the masked round keys. Other details of concrete implementations can be found in [10, 11, 12, 13]. In these papers the countermeasure is proposed for an optimized AES implementation with 8×32-bit lookup tables that are used to compute the S-box and the diffusion simultaneously. We stress that this fact does not influence our analysis and the considerations presented in the following sections are independent from these implementation details.

3. WEAKNESS OF THE SPLIT MASK

First, we recall some details of the original DPA attack of Kocher *et al.* [17]. To mount it against an implementation of an encryption algorithm, one selects an intermediate bit b such that:

1. this target bit b can be expressed as a function $D(C, b, K_s)$ of a known plaintext or ciphertext C and a part K_s of the encryption key that one is willing to recover;
2. the size of the subkey K_s on which b depends is sufficiently small, commonly less than 32 bits (otherwise the attack is rendered impractical mainly by a large number of traces required to distinguish a correct subkey guess);
3. calculating D for an incorrect value of K_s yields the correct value for the target bit b with probability about $\frac{1}{2}$;
4. power consumption of the device depends somehow at some time on the value of b.

The selection function D is then used as a criterion to classify the collected power traces under all possible hypotheses for K_s. For each hypothesis, a differential trace is calculated as a difference of means of the two sets of power traces, one corresponding to $D = 1$, another to $D = 0$. For an incorrect guess of K_s, D is uncorrelated to the actual value of the target bit b computed by the device, and the differential trace is close to the zero line. For the correctly guessed key, the classification of power traces is performed based on the actual value of the target bit. Therefore, the differential power trace should exhibit distinguished peaks at the time when the power consumption of the device depends somehow on the value of the target bit.

Now, it is easy to see that if a target bit is masked with some fixed value then DPA still works. Mask 1 inverts the classification, which just changes the sign of the peaks; mask 0 does not affect anything at all.

For the split mask countermeasure this means that if we are able to find a target bit at the input of some layer of S-boxes that is expressed as a function of a sufficiently small number of key bits and a known plaintext or ciphertext, the original DPA will immediately apply without any respect to the number of shares into which the output mask is split.

In the next section, we describe concrete attacks on AES and DES implementations with the split mask countermeasure that exploit this weakness.

4. ATTACKS DEFEATING THE SPLIT MASK

Here we propose attacks on AES and DES implementations with the split mask countermeasure described in Section 2. We will describe 2 general attacks, 2 attacks for AES, 1 attack for DES. An attack on the strengthened variant of split mask is described in Section 6.

We consider a common security model in which an adversary is allowed to access the cryptographic device and aims to recover the secret key stored in it. She registers the power consumption or electromagnetic emanations of the device at the time it processes data. She also either passively registers the inputs or outputs (known plaintext or ciphertext model) or submits her own inputs (chosen plaintext model). An example of such scenario is when a smartcard of some legitimate owner falls into the hands of a technically equipped villain.

In the descriptions of the DPA attacks below, we will give a number of differential traces that have to be calculated in order to mount an attack. Complexity of calculating a single differential trace is proportional to the number of points in a power trace and to the number of collected power traces, which strongly depends on the properties of the device and on measurement conditions. For simple microcontrollers without hardware countermeasures a standard DPA typically requires hundreds of acquisitions [7, 5] to make the success probability of the attack close to 1. Our experimental results presented in Section 5 show the same complexity.

4.1 General Attacks

4.1.1 Brute force against short mask

First, there is a general attack that applies to both AES and DES implementations in case all S-boxes share the same l-bit fixed input mask n. We recall that in our case "fixed" means that it is the same for all plaintexts encrypted under a single key.

This fixed mask for the input of the first round S-boxes in DES and AES originates from the first round key. This means that the first round key k_0 is masked with the concatenation of identical l-bit masks:

$$k'_0 = k_0 \oplus \{n|n|\dots|n\}$$

Therefore, there are only 2^l possible values for the mask of the first round key (2^8 for AES and 2^6 for DES).

We can recover k'_0 by performing a DPA on `XOR` of k'_0 with the known plaintext as described by Chari *et al.* [5]. In this attack, a single key bit is determined per target bit by computing a single differential trace. Thus, the total number of differential traces needed to recover k_0 is equal to the number of bits in k_0, so post-processing complexity is 128 calculations of differential traces for AES and 48 for DES.

In case of AES k'_0 is the full key masked with one of the 2^8 possible masks, so we can determine the full key by a brute force of 2^8 encryptions. In case of DES, the first round key yields only 48 bits of the full 56-bit key and the brute force complexity is $2^6 \times 2^8 = 2^{14}$. This attack is practical until the effective size l of the key mask is made sufficient enough (by increasing the number of different S-box input masks) to make brute force complexity too large.

4.1.2 Side-channel collision attacks

This is is another class of attacks that apply both to AES and DES implementations with the split mask countermeasure. They can be considered as a kind of simple power analysis rather than as DPA and are based on the possibility of detecting the equality of intermediate variables using a small number of power traces. The fact that the S-box input mask is fixed for different plaintexts allows one to detect collisions at the inputs of S-boxes just as in an unmasked implementation (and with the same complexity). Original collision attacks first addressed in detail in the work of Schramm *et al.* [22] and employing byte collisions that occur at the single S-box input in a pair of algorithm executions, are possible if the masks at the inputs of different S-boxes are not necessarily the same. This scenario also enables impossible and multiset collision attacks proposed by Biryukov and Khovratovich [2]. In case single fixed mask is shared by different S-boxes, more efficient attacks with the generalized collisions, introduced by Bogdanov [3] and improved by Bogdanov *et al.* [4], are even possible.

Now we describe other attacks specific to AES and DES. The attacks do not require all S-box input masks to be the same. They require only that these possibly different input masks remain fixed for all plaintexts under a single key, which is the case of the split mask countermeasure.

4.2 Attacks on AES

4.2.1 Attack 1

This attack aims at the first two rounds. It is a DPA attack at the inputs of second round S-boxes that are masked with a fixed mask. The scheme of this attack is shown in Figure 2.

We view an input byte s_ϵ to a second round S-box as a `XOR` of a second round key byte $k_{1,\epsilon}$ masked by a fixed mask n_ϵ with a known function of the 4 plaintext bytes $t_\alpha,\dots,t_\delta$ and corresponding 4 bytes $k_{0,\alpha},\dots,k_{0,\delta}$ of the full encryption key k_0:

$$s_\epsilon = k_{1,\epsilon} \oplus n_\epsilon \oplus F(k_{0,\alpha} \oplus t_\alpha, k_{0,\beta} \oplus t_\beta, k_{0,\gamma} \oplus t_\gamma, k_{0,\delta} \oplus t_\delta). \quad (3)$$

In Figure 2 we denote $t \oplus k_0$ by x, and to be precise, use $\bar{m}$ to denote fixed mask m transformed by `SubBytes` and `MixColumns`.

Values of $k_{1,\epsilon}$ and n_ϵ are unknown but fixed for different plaintexts. So F is in fact masked with the fixed value $k_{1,\epsilon} \oplus n_\epsilon$. We select target bit b from s_ϵ (*i.e.* at the input of the second round S-box layer) and perform a standard DPA as described in Section 3 to recover the bytes $k_{0,\alpha},\dots,k_{0,\delta}$ of the full encryption key. By further selecting target bits from another 3 S-boxes (chosen from different columns according to diffusion properties of the AES round) we recover the remaining parts of the key. But in this way we have to guess 4 bytes of the key at a time, which demands a large number of traces and long post-processing.

To make this attack practical, we employ the linearity of AES `MixColumns` transformation that allows representing F as as a `XOR` of the 4 known functions of individual first round S-box input bytes, so that (3) becomes

$$s_\epsilon = k_{1,\epsilon} \oplus n_\epsilon \oplus F_1(k_{0,\alpha} \oplus t_\alpha) \oplus F_2(k_{0,\beta} \oplus t_\beta) \oplus F_3(k_{0,\gamma} \oplus t_\gamma) \oplus F_4(k_{0,\delta} \oplus t_\delta). \quad (4)$$

Now we choose the plaintexts that are fixed, say, in bytes t_β, t_γ and t_δ (in terms of differential cryptanalysis, we leave one active S-box). Then (4) can be viewed as

$$s_\epsilon = u \oplus F_4(k_{0,\alpha} \oplus t_\alpha)$$

where u is some unknown value that is fixed for different chosen plaintexts. We apply DPA as above, selecting a target bit from s_ϵ, but have to guess now only a single byte $k_{0,\alpha}$ of the initial key. By repeating this step another 15 times

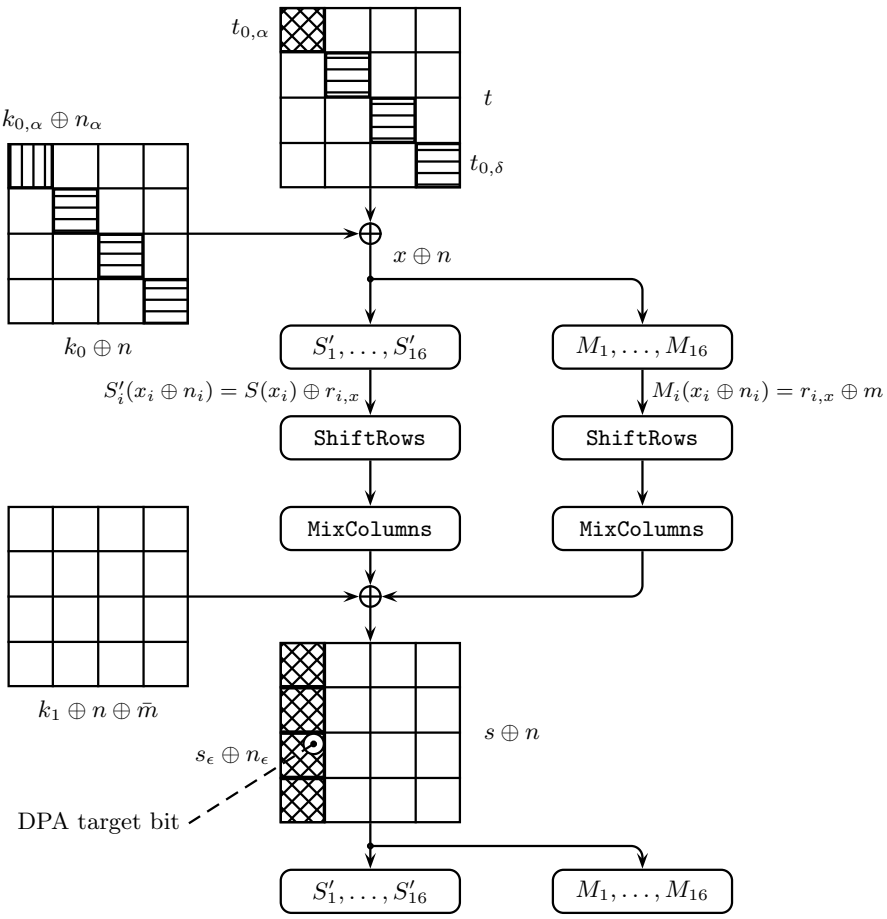

Figure 2: First AES rounds with the split mask

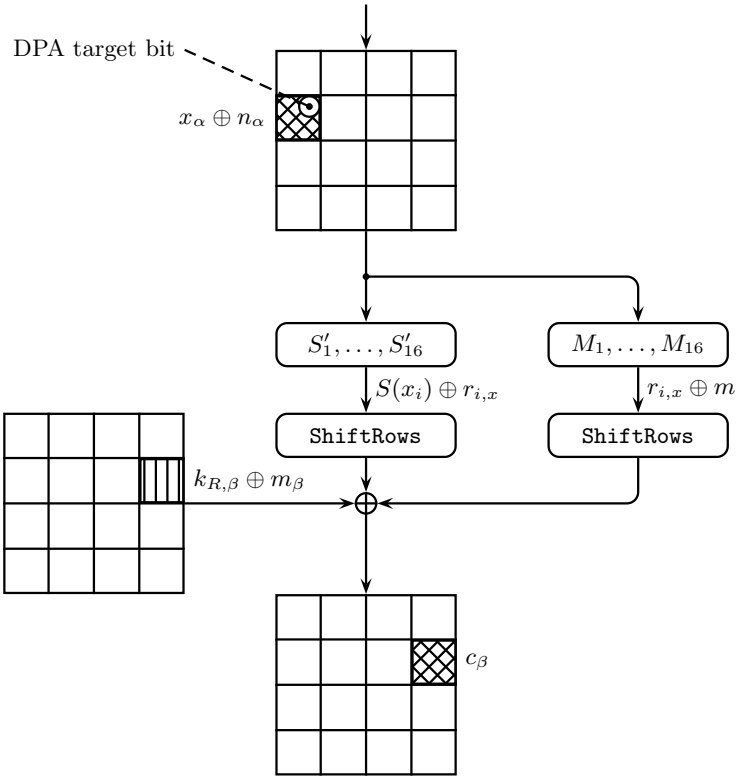

Figure 3: Last AES round with the split mask

with the target bits sequentially chosen at different second round S-box inputs and appropriately chosen plaintexts, we recover the remaining 15 bytes of the full key byte by byte. If plaintexts are chosen in a more sophisticated way to have all the bytes in a single column active and all the remaining bytes fixed, we can reuse them and thus reduce the total number of required chosen plaintexts by a factor of 4. A total of $16{\times}2^8 = 2^{12}$ differential traces are calculated in this attack.

We note that Attack 1 works in the chosen plaintext model. In the following, we describe another more powerful (and actually simpler) attack that works in the known ciphertext model only.

4.2.2 Attack 2

This attack aims at the last round S-box input and recovers the last round key. Figure 3 illustrates the attack.

We recall that the last AES round does not include `MixColumns` transformation. Therefore, a byte x_α at the input of the last S-box layer can be viewed as an `XOR` of the fixed mask n_α with the known function of a last round key byte $k_{R,\beta}$ and a corresponding ciphertext byte c_β,

$$x_\alpha = n_\alpha \oplus S^{-1}(k_{R,\beta} \oplus c_\beta).$$

Again, n_α is constant for different plaintexts. So choosing the target bit from x_α enables one to perform DPA in a straightforward manner and obtain the last round key byte by byte as if the implementation was not masked at all. As in the previous attack, $16{\times}2^8 = 2^{12}$ differential power traces have to be computed in total, but choosing plaintexts at the stage of collecting acquisitions is not required.

4.3 Attack on DES

For DES implemented with the split mask, we simply exploit the same approach as described for AES. See Figure 4 that shows the attack path (dashed line) on the scheme of the DES implementation with the split mask. We view a 6-bit input s of a second round S-box as a `XOR` of a fixed 6-bit part k_2 of the second round key masked with a fixed mask n, 6 bits of the left plaintext part L and a known function of 6 bits $k_{1,\alpha},\ldots,k_{1,\zeta}$ of the first round key entering one of the first round S-boxes and corresponding bits $R_\alpha,\ldots,R_\zeta$ of the right plaintext part:

$$s = k_2 \oplus n \oplus L \oplus F(k_{1,\alpha} \oplus R_\alpha, \ldots, k_{1,\zeta} \oplus R_\zeta).$$

In Figure 4, $k_1 \oplus R = x$. As $k_2 \oplus n$ is fixed for different plaintexts and L is known, we can mount a DPA attack with the target bit from s to recover 6 bits of the first round key.

By attacking the inputs of different second round S-boxes in a sequential way, all 48 bits of the first round key can be recovered by 6-bit chunks at a time. In total we have to compute $8{\times}2^6 = 2^9$ differential power traces. Remaining 8 bits of the full key can be determined by an exhaustive search. The attack works in the known plaintext model.

Note that all attacks described above do not depend on the number of mask tables used (*i.e* on the number of output mask shares). They are also independent of dynamic updating of the output masks r_x that suggested in [11] to strengthen the countermeasure. This updating is performed after access of data from both $S'(x)$ and $M(x)$ by remasking of these two entries with a newly generated random value. In Section 6 we will show that another strengthened version

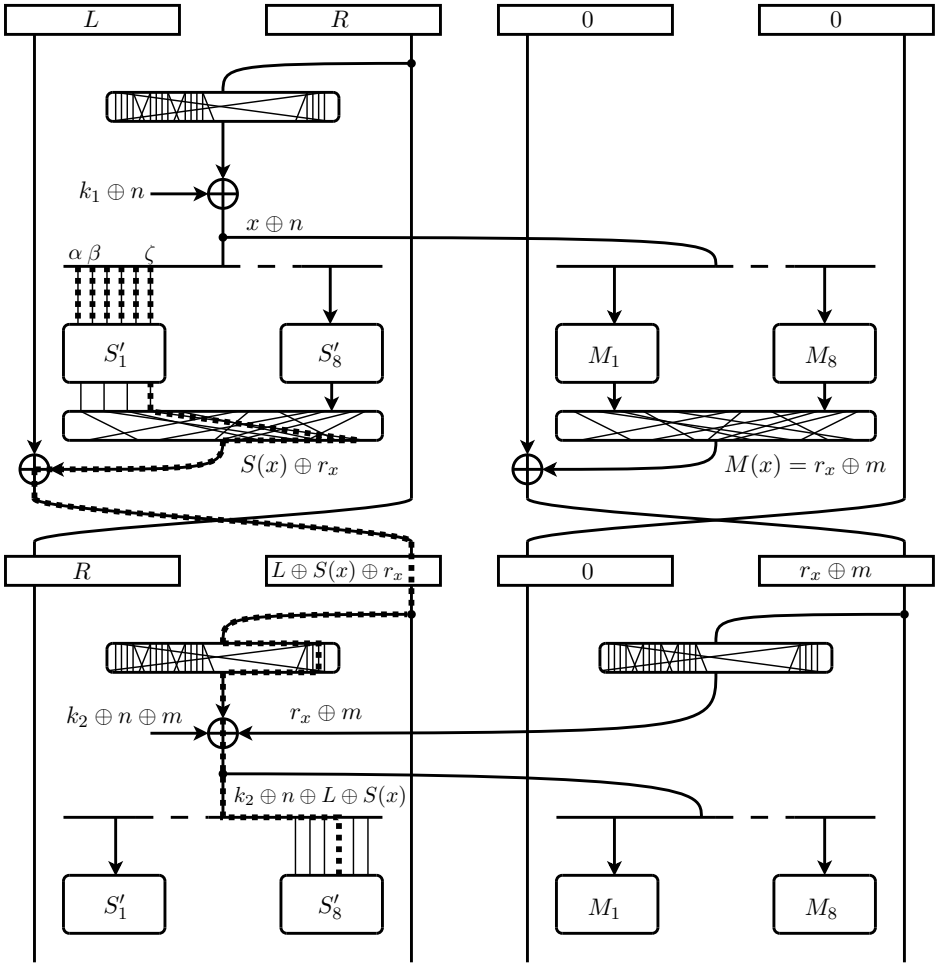

Figure 4: First 2 rounds of DES with the split mask

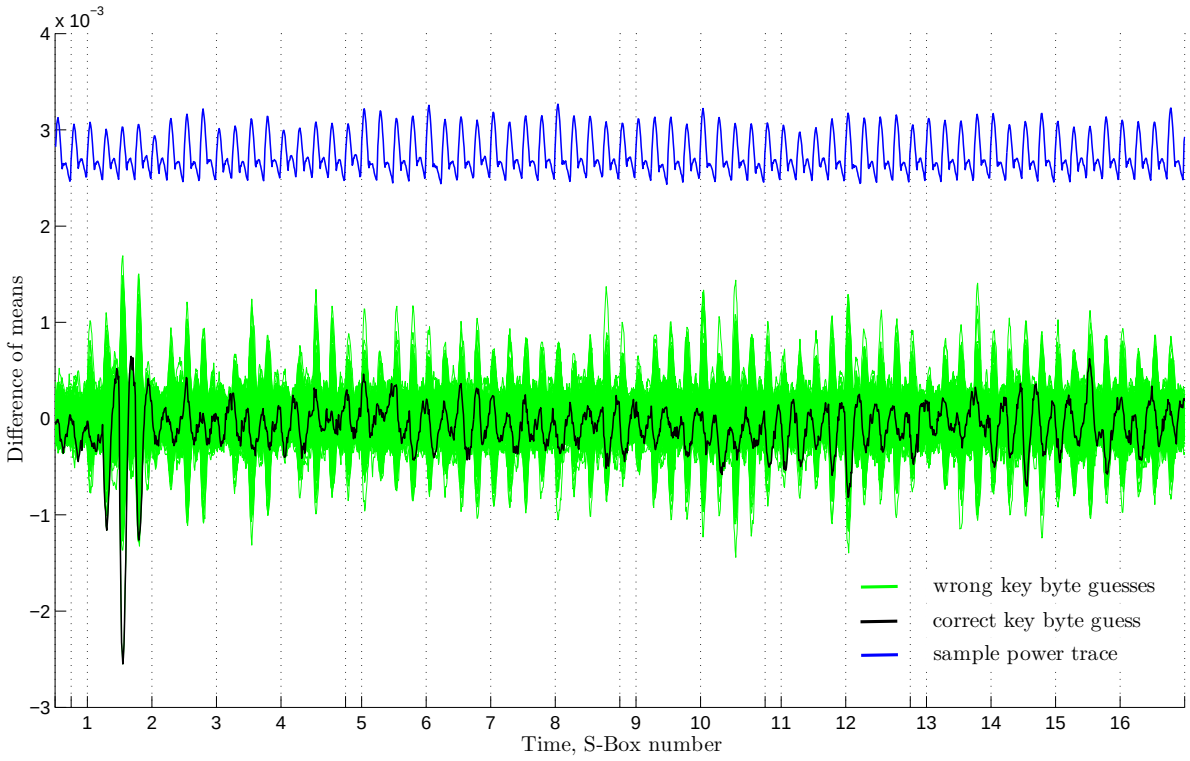

Figure 5: DPA on the last AES round with the split mask countermeasure

of the countermeasure is not as secure as it is supposed in [11] to be.

5. PRACTICAL ATTACK IMPLEMENTATION

We performed the experimental verification of our attack for the case of AES-128 with the split mask countermeasure. The known-ciphertext Attack 2 on the last AES round was implemented.

The target device is Atmel ATmega16 [1]. This is a microcontroller from the 8-bit AVR family. AVR is a RISC core with most instructions taking 1 clock cycle. It follows a Harvard architecture, having separate data and program memories. ATmega16 has 16 KBytes of flash program memory, 1 KByte of SRAM and 512 Bytes of EEPROM. 8-bit AVR microcontrollers are widely used in embedded devices and smart-cards.

We added the split mask countermeasure to *RijndaelFurious* AES-128 implementation of Poettering [20]. Both masked and mask tables were stored in the program memory and accessed with the `LPM` instruction. SRAM was used to store intermediate mask values. A trigger signal was set by the implementation on a microcontroller's pin at the beginning of the last round.

The microcontroller was clocked at 3.68 MHz and supplied with an operating voltage of 5V from a standard laboratory power source. A shunt resistor of 5.6 Ohm was inserted into the ground line of the microcontroller to observe the variations of the power consumption. The target board with the microcontroller, the quartz oscillator and the shunt resistor was placed into a Faraday cage made of copper foil to reduce the electric field noise from the surrounding devices. Communication with the controlling PC was performed via the ATmega16 built-in USART controller and a MAX232-based voltage converter at the transfer rate of 9600 baud. We note that this measurement setup is just a reference one and is neither noise- nor speed-optimized.

The measurements were performed with a LeCroy WaveRunner 104MXi DSO equipped with ZS1000 active probe. The DSO has 8-bit resolution, 1 GHz input bandwidth (with the specified probe) and maximum sampling rate of 10 GS/s. We captured the data at 10 GS/s to reduce the trigger jitter, following suggestions from [23] and decimating traces afterwards. Vertical resolution was set to 160 mV peak-to-peak. With the given shunt resistor, variations of the consumed current as small as 110 μA could be registered (while the average power consumption of the microcontroller in the given configuration was about 12 mA). The DSO, the power source and the target device shared the common ground, also connected to the Faraday cage.

Data acquisition was controlled by a host PC. LeCroy's ActiveDSO ActiveX component was used to control the DSO remotely via 100 Mbit Ethernet connection, and serial interface was used to communicate with the target microcontroller. The acquisition rate was about 12 traces per second. The number of acquired traces was 400, so the total acquisition time was about half a minute.

Following the acquisition, the traces were decimated by a factor of 100 to reduce the amount of processed data. The sampling rate was thus reduced to 100MS/s. The decimation included low-pass filtering to avoid aliasing in the frequency domain.

Figure 5 shows the result of Attack 2 on the input of the

1-st masked S-Box of the last AES round. Differential traces for all 256 guesses of the corresponding last round subkey byte are shown. The differential trace for the correct guess is highlighted; a distinct DPA peak can be seen in it. Marks on the horizontal axis locate individual S-Box executions within a trace. Each S-Box takes 4 clock cycles, 1 for setting the lookup address with the `MOV` instruction and 3 for the lookup itself with the `LPM` instruction. A reference power consumption trace is also shown above to help locating clock cycles.

Remaining last round S-boxes are attacked in the same way (reusing the power traces) to recover the full 128-bit last round key, from which the full 128-bit AES encryption key is easily derived. Thus, the attack can be performed in practice with the success probability very close to 1 with about 400 power traces. It requires about half a minute for data acquisition, less than 1 MByte of memory for trace storage and about a minute for post-processing.

6. ATTACKS ON THE STRENGTHENED METHOD

In the most recent work [11] on the split mask countermeasure the existence of a chosen-plaintext Attack 1 for AES is briefly mentioned and a solution to thwart it is suggested. This solution extends the original split mask countermeasure so that the masks at the inputs of S-boxes are no longer fixed. They are generated randomly or selected from a pregenerated set of masks as in [16] after a large number of algorithm executions, which is less than the number of acquisitions required for the successful DPA attack.

We note that changing the S-box input masks infrequently still allows for collision attacks that require only a small number of side-channel signal acquisitions [4]. Therefore it is reasonable to consider the modified split mask with the S-box input masks modified for each plaintext, *i.e.* new random masks for the S-box inputs are generated and the tables are recomputed in the beginning of each encryption run. First, this eliminates the performance advantage of the split mask countermeasure over other existing masking techniques. Second, in the following we show that this modified method can be attacked with a second-order DPA attack without respect to the number of mask tables used for splitting the S-box output mask.

Second order DPA attacks [19] exploit the fact that the unmasked internal variable can be correlated to some combination of two power consumption values: one corresponding to processing of the masked variable and the other corresponding to the processing of the mask. To cope with the noise, second order DPA attacks typically require more power traces compared to a common (first-order) DPA attack.

To attack the input of an S-box that is masked according to the modified split mask countermeasure with a mask n, we should look for points in a power trace that depend only on the value of the mask n. Such points exist at the times when the mask is generated by the device and when the tables are recomputed. So the second order DPA attack can be performed to recover the subkey byte corresponding to that S-box entry.

Even if there were no points depending on the mask n (imagine that the mask and the precomputed tables are just loaded into the device prior to the algorithm execution), we could create them by choosing proper plaintexts in the following tricky way.

We assume that S-box input masks within one round are all not necessarily the same, but all AES rounds share same masked S-box implementations (as the original description [11] allows). In other words, i-th S-box in any round has the same input mask n_i that is randomly chosen in each execution as we are considering now the modified method. We choose plaintexts that are fixed in i-th byte so that one byte at the input of i-th S-box in the first round is fixed. That means that the power consumption for this S-box input depends only on the value of the mask n_i. Thus, it can be chosen as one of the points for the second order DPA attack. The other point, corresponding to the masked variable, is at the input of the i-th S-box of the second round, as it uses the same input mask. To be able to guess only one byte of the key at a time, we choose plaintexts with the additional restrictions described in Attack 1 on AES.

Complexity of such second order DPA attack against the strengthened variant of the split mask countermeasure is higher than that of the standard DPA attacks described in Section 4 for the original split mask method. It would require thousands of traces to be acquired. But we stress that this attack does not depend on the number of mask tables used. It has the same complexity for any number of shares the output mask is split into. Thus, the claim in [11] that the split mask countermeasure using N mask tables thwarts an N-th order DPA attack does not hold.

This supports a general consideration that in order to properly protect an implementation from an N-th order DPA attack, every intermediate variable should be split into $N + 1$ shares.

7. CONCLUSION

We showed that the split mask countermeasure does not provide any protection against the standard DPA attack independently of the number of mask tables and of the possible modifications of the output masks. Several attack paths were presented for the AES and the DES implementations protected with this countermeasure. The most effective DPA attacks require nothing more than selecting an appropriate attack point, the technique and complexity of these attacks being the same as for the attacks on the unmasked implementations.

We verified this by implementing the attack on the last round of AES against the AES implementation with the split mask countermeasure running on an 8-bit AVR microcontroller. This practical full key recovery attack requires about 400 power traces acquired with a non-optimized measurement setup.

The suggested attack paths should be true not only for AES and DES, but also for the other cryptographic algorithms protected with the split mask. Though for more complex hardware like 32-bit platforms the complexity of the attacks will increase, they will still work. This is due to the fact that the attacks are caused by the general weakness of the countermeasure. The weakness is in using fixed masks that do not change for different plaintexts.

Modifying the countermeasure by introducing frequent updates of the masks cancels its performance advantage over other masking proposals and provides protection only against 1-st order DPA attacks. We have outlined a way to mount a second-order DPA attack on this strengthened variant of

split mask with any number of mask tables.

To fix the weakness, any fixed masks should be avoided and for N-th order DPA resistance *every* sensitive intermediate variable within the implementation should be split into $N + 1$ shares. This will increase the performance overhead, so a trade-off between security and performance should be carefully chosen and combination with other countermeasures should usually be considered. Examples of the recent practical masking schemes combined with hiding can be found in [15] and [21].

8. REFERENCES

[1] ATmega16: 8-bit AVR microcontroller with 16K bytes In-System Programmable Flash, revision S, May 2009. Available from `http://www.atmel.com/dyn/resources/prod_documents/doc2466.pdf`, accessed 6 August 2009.

[2] A. Biryukov and D. Khovratovich. Two new techniques of side-channel cryptanalysis. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embeded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 166–180. Springer, Heidelberg, 2007.

[3] A. Bogdanov. Improved side-channel collision attacks on AES. In C. Adams, A. Miri, and M. Wiener, editors, *The 14th Annual Workshop on Selected Areas in Cryptography (SAC 2007)*, volume 4876 of *Lecture Notes in Computer Science*, pages 84–95. Springer, Heidelberg, 2007.

[4] A. Bogdanov, I. Kizhvatov, and A. Pyshkin. Algebraic methods in side-channel collision attacks and practical collision detection. In D. R. Chowdhury, V. Rijmen, and A. Das, editors, *INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 251–265. Springer, Heidelberg, 2008.

[5] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. A cautionary note regarding evaluation of AES candidates on smart-cards. In *Second Advanced Encryption Standard Candidate Conference: AES2*, Rome, March 1999. Available from `http://csrc.nist.gov/archive/aes/round1/conf2/papers/chari.pdf`, accessed 6 August 2009.

[6] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, Heidelberg, 1999.

[7] C. Clavier, J.-S. Coron, and N. Dabbous. Differential power analysis in the presence of hardware countermeasures. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, Heidelberg, 2000.

[8] FIPS PUB 197: Specification for the Advanced Encryption Standard, 2001. Available from `http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf`, accessed 6 August 2009.

[9] FIPS PUB 46-3: Data Encryption Standard, 1999. Available from `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`, accessed 6 August 2009.

[10] C. H. Gebotys. A split-mask countermeasure for low-energy secure embedded systems. *ACM Trans. on Embedded Computing Systems*, 5(3):577–612, August 2006.

[11] C. H. Gebotys. A table masking countermeasure for low-energy secure embedded systems. *IEEE Trans. on VLSI*, 14(7):740–753, July 2006.

[12] C. H. Gebotys, S. Ho, and C. C. Tiu. EM analysis of Rijndael and ECC on a wireless Java-based PDA. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 250–264. Springer, Heidelberg, 2005.

[13] C. H. Gebotys, C. C. Tiu, and X. X. Chen. A countermeasure for EM attack of a wireless PDA. In *International Conference on Information Technology: Coding and Computing (ITCC 2005)*, volume 1, pages 544–549, April 2005.

[14] L. Goubin and J. Patarin. DES and differential power analysis: The "duplication" method. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, Heidelberg, 1999.

[15] C. Herbst, E. Oswald, and S. Mangard. An AES smart card implementation resistant to power analysis attacks. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security – ACNS 2006*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252. Springer, Heidelberg, 2006.

[16] K. Itoh, M. Takenaka, and N. Torii. DPA countermeasure based on the "masking method". In K. Kim, editor, *Information Security and Cryptology – ICICS 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 440–456. Springer, Heidelberg, 2002.

[17] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Weiner, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 338–397. Springer, Heidelberg, 1999.

[18] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[19] T. S. Messerges. Using second-order power analysis to attack DPA resistant software. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, Heidelberg, 2000.

[20] B. Poettering. AVRAES: The AES block cipher on AVR controllers, 2006. Available from `http://point-at-infinity.org/avraes`, accessed 6 August 2009.

[21] M. Rivain, E. Prouff, and J. Doget. Higher-order masking and shuffling for software implementations of block ciphers. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 171–188. Springer, Heidelberg, 2009.

[22] K. Schramm, T. Wollinger, and C. Paar. A new class of collision attacks and its application to DES. In T. Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 206–222. Springer, Heidelberg, 2003.

[23] H. Seuschek. DPA-Analyse von Implementierungen symmetrischer kryptographischer Algorithmen. Dimplomaarbeit, TU München, April 2005. (In German.) Available from `http://www.torsten-schuetze.de/reports/diplom_seuschek.pdf`, accessed 6 August 2009.