Multiplicative Masking and Power Analysis of AES

Jovan Dj. Golić*

Abstract

The recently proposed multiplicative masking countermeasure against power analysis attacks on AES is interesting as it does not require the costly recomputation and RAM storage of S-boxes for every run of AES. This is important for applications where the available space is very limited such as the smart card applications. Unfortunately, it is here shown that this method is in fact inherently vulnerable to differential power analysis. Other possible random masking methods are also discussed.

Key words AES, differential power analysis, countermeasures, multiplicative masking.

1 Introduction

Side-channel attacks on software or hardware implementations of various cryptosystems aim at recovering the secret key information from certain physical measurements performed on the electronic device during the computation such as the power consumption, the time, and the electromagnetic radiation. Power analysis attacks [8] are very powerful as they do not require expensive resources and as most implementations without specific countermeasures incorporated are vulnerable to such attacks. Among them, (the first-order) differential power analysis (DPA) attacks are particularly impressive, because they use relatively simple mathematical tools and as techniques are independent of the implementation of the cryptographic algorithm. Moreover, the countermeasures are typically costly in terms of speed performance and memory requirements.

The goal of simple power analysis (SPA) attacks is to deduce some information about the secret key, such as the Hamming weight of some parts of the key, from a single power consumption curve. This may be possible if, for example, there are branches in the computation that depend on the secret key. More generally, one can also collect a large training set

*The author is with Rome CryptoDesign Center, Gemplus, Via Pio Emanuelli 1, 00143 Rome, Italy. Email: jovan.golic@gemplus.com. Preliminary version of this work was presented at the Gemplus Quarterly meeting, La Ciotat, France, October 30-31, 2001.

of power consumption curves from different secret keys (and possibly different input data) and then use appropriate statistical hypothesis testing methods in order to identify traces or signatures of the parts of the secret key hidden in the curves. For example, key scheduling algorithms for block ciphers may especially be vulnerable in this regard, due to the absence of the randomization effect of input data. However, the statistical techniques to be used can be complicated and dependent on particular implementations.

The DPA attack [8] requires a set of power consumption curves obtained by running the cryptographic algorithm a number of times for the same secret key and different inputs. A necessary algorithmic condition, the so-called *fundamental hypothesis*, for the DPA attack to be effective is the existence of one or more intermediate variables in the algorithm that can be expressed as or are correlated to functions depending on a small number of key bits and on known input or output data. The key bits involved may then be reconstructed by partitioning the set of curves according to the value of the chosen intermediate variable corresponding to the key bits guessed and to the input or output data known and by computing and comparing some simple statistic, such as the average value, on the partitioned curves at individual points in time. The attack is successful if the correct guess about the key bits results in a significant difference between the computed average curves at one or more points in time. For other possibilities, see [4]. What makes the attack practically very interesting is that many cryptographic algorithms satisfy the fundamental hypothesis. For example, the intermediate variables in the first or the last few rounds of practical block ciphers are especially vulnerable.

A high-order DPA attack is a generalization of the (first-order) DPA attack in which the power consumption curves are analyzed by using a joint statistic applied to collections of points in time. The general attack is more powerful, but may be more complex and considerably more complicated as the choice of these points and possibly also of the joint statistic is likely to depend on the particular implementation.

In principle, the complexity of the side-channel attacks can be increased by introducing physical or algorithmic countermeasures. A general strategy to render the SPA and DPA attacks more difficult to mount is to balance and randomize elementary computations involving the secret key, e.g., by randomly introducing dummy operations and timing shifts, as well as by randomizing the order of elementary computations and the computations themselves. A general technique to prevent the first or high-order DPA attacks is random data splitting [7], [3], especially for the computation of intermediate variables satisfying the fundamental hypothesis. It is pointed out in [9] that for the (first-order) DPA, instead of splitting the data into two parts one may as well apply random masks to data which are easier to implement. Of course, one has to be careful to mask the data completely and thus avoid weaknesses such as the one shown in [5] for a masking technique from [9]. Also, the computations involved in masking have to be performed in a secure way, which itself is not vulnerable to DPA. Random masks have to be generated for each new run of the cryptographic algorithm, but may be repeated within the algorithm. The repetitions generally increase the vulnerability to high-order DPA. The random masks can be combined with data by using (quasi)group operations such as the bitwise addition or modular integer addition.

If an affine transformation is applied to masked data and if the masking operation is the same as the corresponding linear operation, then only the additive constant has to be recomputed for each new mask in order to maintain the equivalence of the data transformations. However, this is generally not the case with nonlinear transformations. They typically have to be recomputed depending on the mask and this can be very costly for many cryptographic algorithms including AES [6]. In AES, the only nonlinear transformations are nonlinear parts of 8×8-bit S-boxes which perform the multiplicative inversion in GF(256). In [2], a masking technique is proposed which combines the usual binary additive masking with the multiplicative masking of data, using the multiplication in GF(256), thus avoiding the costly recomputation of the S-boxes.

In this paper, it is shown that this masking technique is vulnerable to the first-order DPA attack and in some sense even more than AES without masking. This is essentially because the all-zero input to the S-boxes is not effectively masked by the multiplicative mask and the binary additive mask is first removed in order to apply the multiplicative mask. Moreover, it is argued that the weakness is inherent to the multiplicative masking and hence cannot be remedied. It is also pointed out that the masking technique [9] in which only one S-box is recomputed and stored in RAM and used repeatedly during one execution of AES may be vulnerable to a relatively simple second-order DPA attack. The main lines of the DPA attack [8] applied to AES are described in more detail in Section 2 and the multiplicative masking technique is presented in Section 3. The inherent weakness is explained in Section 4 and conclusions are given in Section 5.

2 Differential Power Analysis of AES

AES is a product block cipher composed of a number of rounds each consisting of a reversible nonlinear transformation providing *confusion* and a reversible linear transformation providing *diffusion*, where the linearity is with respect to the binary field, GF(2). The expanded secret key is bitwise added to the plaintext and to the output of each round. The nonlinear transformation consists of identical 8×8-bit S-boxes each performing the byte substitution ByteSub defined as the multiplicative inversion in GF(256), leaving the all-zero input intact, followed by an affine 8×8-bit transformation. The linear transformation consists of a permutation of byte outputs of S-boxes denoted as ShiftRow followed by a linear transformation denoted as MixColumn, which is removed from the last round. More details can be found in [6], but are irrelevant for our analysis.

According to [8], the DPA attack on AES consists of the following stages. The intermediate variables satisfying the fundamental hypothesis are the output bytes of the S-boxes or of just the nonlinear parts of the S-boxes in the first round. Each of them is a function of the input byte which itself is a bitwise sum of the corresponding plaintext and expanded key bytes. Accordingly, if the plaintext byte is known and the key byte is guessed correctly, then the S-box output byte can be computed correctly. The objective of the attack is to recover the expanded key in a byte-by-byte divide-and-conquer manner.

In the first stage, a sufficient number, N, of curves are obtained by measuring the power consumption during the execution of (the first round of) AES for the same secret key and N different plaintexts. The average value C of these N curves is then computed. The second stage is performed for each of the S-boxes in the first round. For each of 256 possible values of the targeted expanded key byte K, a subset of M (on average, $M = N/2^m$) plaintexts resulting in a chosen fixed m-bit value of the partial output byte of the considered S-box are identified, the corresponding M curves are extracted, and their average value $C(K)$ computed. For example, the chosen fixed value may have maximal or minimal Hamming weight (all-one or all-zero m-bit words). More generally, if one knows good power consumption models of the involved components, an optimal subset of M curves can be chosen according to a set of 2^{8-m} or of any other number of output byte values best distinguished from the others with respect to power consumption, as proposed in [1] for $m = 1$.

A value K is then decided to be correct if the difference between the two average curves, $C(K)$ and C, contains one or more noticeable peaks. The peaks are due to the same S-box output being computed at the same time for each of the extracted M curves, if the value K is correct, and to unbalanced power consumption associated with different S-box output values. *This is the main point of the DPA attack.* On the other hand, if the value K is incorrect, then the outputs of the S-box will vary and the peaks will not be observed. More precisely, this is the case for $m < 8$. For $m = 8$, as the S-boxes are reversible, a fixed output byte value implies that the input byte value is also fixed. Therefore, even if the guessed value K is incorrect, then for the extracted curves both the input and output bytes will have fixed values, different for different K, also giving rise to observable peaks, possibly of different magnitudes for different K. As a consequence, the reliable statistical distinction of the correct K may be infeasible.

The smaller m is, the bigger M becomes, but the impact of the repeated computation on each of the M extracted power consumption curves is statistically less significant. Also, it is not clear how one can simultaneously use more than just one fixed output m-bit value in order to increase the statistical distinction between the correct and incorrect key values. Nonetheless, this may be possible.

According to the key schedule in AES, if the key size is not bigger than the plaintext block size, then the recovered expanded key bytes directly specify the secret key. Otherwise, the DPA attack should also be applied to the second round of AES in order to recover the whole secret key.

3 Multiplicative Masking of AES

The starting idea of the method proposed in [2] in order to prevent the DPA attack on AES is to use the binary additive mask which is compatible with the binary linear or affine transformations in AES. Accordingly, as far as the affine transformations are concerned, only the additive constants are affected by this mask. However, if the binary additive mask is applied to the input of the nonlinear part of an S-box in AES, then this nonlinear part has to be recomputed for each new mask used. Recall that the nonlinear part, F, of the S-box transformation ByteSub is the multiplicative inversion in GF(256) extended by mapping the all-zero input into the all-zero output. For simplicity, F is called the inversion in GF(256). The main idea from [2] is to use the nonzero multiplicative mask, with respect to the multiplication in GF(256), for the data passing through F, without having to recompute F. To this end, one has to convert the binary additive mask into the multiplicative mask at the input of each F and to reproduce the binary additive mask from the multiplicative mask at the output of each F. A way, secure with respect to DPA, of converting the masks is suggested in [2]. More details are given below.

Fig. 1 shows the data flow in the i-th round of AES without and with the masking countermeasure. A general rule in all the figures presented is that the expressions for input, output, and all intermediate data are displayed within the rectangular blocks. It is assumed that the ByteSub and Modified ByteSub transformations act on all the bytes in a block. Note that the expanded key is bitwise added to the plaintext to form the input to the first round, that the MixColumn transformation is removed from the last round, and that the binary additive mask is not produced at the output of the last round. According to [2], the binary additive mask X is the same in every round. In fact, keeping the same binary additive mask in every round would be important if the S-boxes had to be recomputed for each new mask, because in that case the same recomputed S-boxes could be used in each round. So, restoring the same mask X in the last step of each round is not really needed. Instead, one can just add the expanded key K_i and thus effectively obtain the output mask X^3. Only in the last round, the outuput mask has to be removed. Here, the masks are transformed by the linear transformations as $X^1 = L(X)$, $X^2 = \mathrm{ShiftRow}(X^1)$, and $X^3 = \mathrm{MixColumn}(X^2)$, where L denotes the linear part of the affine transformation of ByteSub combined for all the bytes in a block. So, essentially only the ByteSub transformation has to be modified, because of the nonlinear parts contained.

The data flow through the original and modified ByteSub transformations, acting on bytes, is shown in Fig. 2 (the index j stands for a particular byte in a block and the index i stands for a particular round). The affine transformation is unchanged, and only the nonlinear transformation, F, has to be modified. This is achieved by using a nonzero multiplicative mask $Y_{i,j}$ in a way displayed in Fig. 3, which is self-explanatory.

Recall that the addition in GF(256) is the same as the bitwise addition. It follows that F does not have to be recomputed and stored in a look-up table for each new mask $Y_{i,j}$. This is due to the multiplication in GF(256) being compatible with F or, more precisely, to the equality

$$F(A \otimes Y) \; = \; F(A) \otimes F(Y) \tag{1}$$

where $F(Y) \neq 0$ if $Y \neq 0$, so that $F(A)$ can be recovered from $F(A) \otimes F(Y)$. In other words, if a masked input is transformed by F itself, then the masked desired output is obtained. So, one just has to convert the multiplicative into the binary additive mask and vice versa, and that can be done by one more inversion, four multiplications, and two additions in GF(256).

Note that in general, if two, possibly different, group operations $*$ and $\bullet$ are used for masking the input and output data for a transformation F, then the masked data should be transformed by the modified transformation F' satisfying $F'(A * Y_1) = F(A) \bullet Y_2$. Equiva-

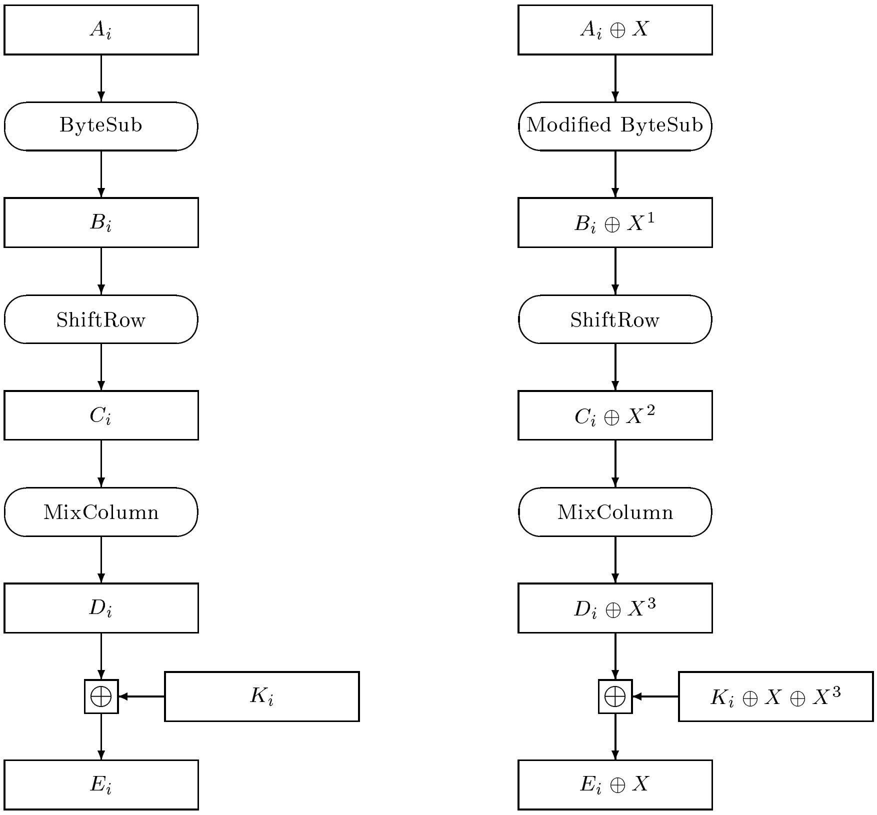

Figure 1: The round i of AES without and with masking countermeasure.

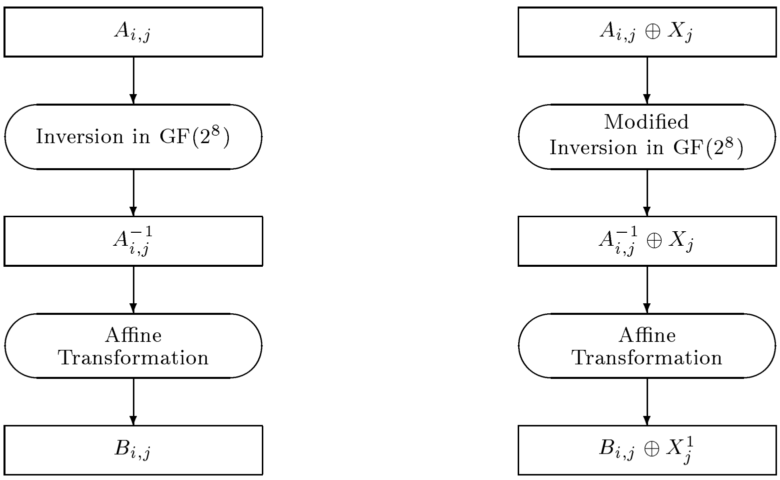

Figure 2: The ByteSub transformation without and with masking countermeasure.

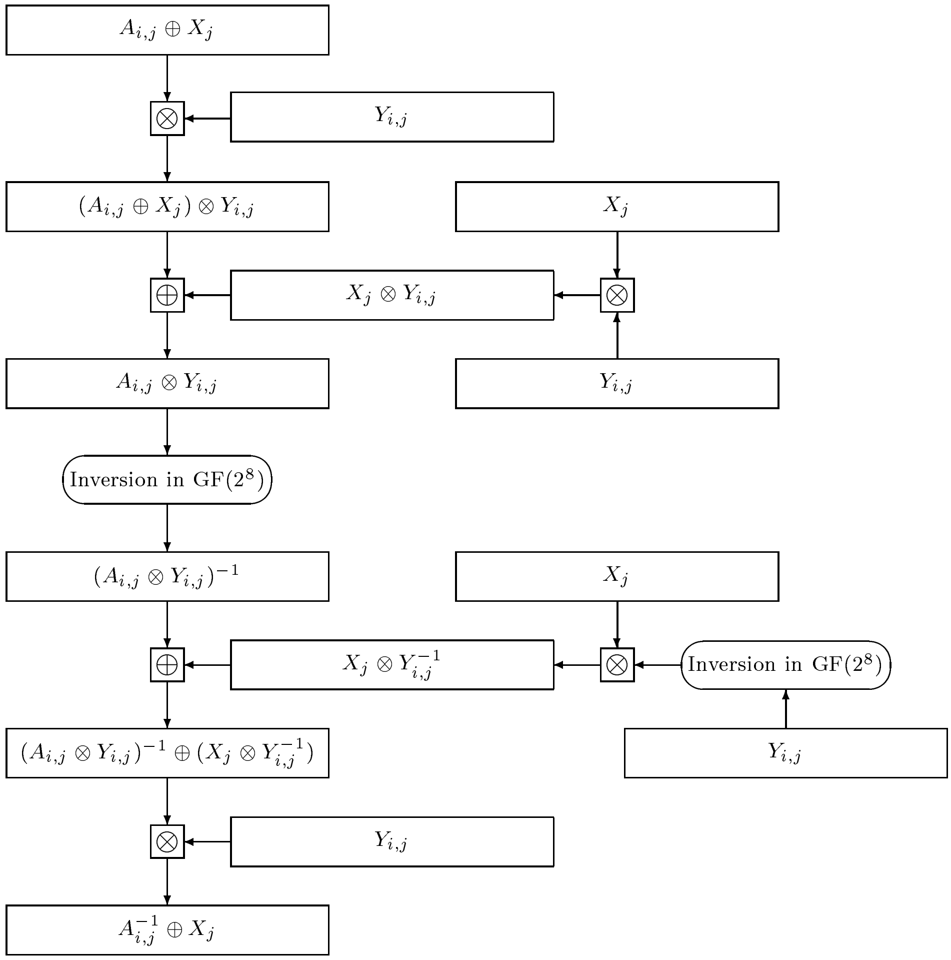

Figure 3: Modified inversion in GF(256) with multiplicative masking countermeasure.

lently, F' is defined by

$$F'(A) \;=\; F(A * Y_1^{-1}) \bullet Y_2, \tag{2}$$

where Y_1 and Y_2 are the input and output masks, respectively, which can be mutually related. However, in order to resist DPA, F' should not be directly implemented by using F and (2). For example, a secure way would be to use a look-up table for F', but it has to be recomputed and stored for every new pair of masks Y_1 and Y_2.

The multiplicative masks $Y_{i,j}$ and the binary additive masks X_j can be randomly chosen so as to be uniformly distributed and mutually independent. In fact, the masks $Y_{i,j}$ can be the same for each round i and possibly related to X_j. Of course, this generally increases the vulnerability to high order DPA. Since all the intermediate variables in Fig. 3 are masked, it is claimed in [2] that the masked AES should be resistant to the (first-order) DPA attack. This masking method is important, because it avoids the recomputation and storage of S-boxes for each new run of AES, which would, for example, require 256×16 bytes of RAM for the 128-bit AES if all the S-boxes in a round are masked by independent masks.

4 Differential Power Analysis of Masked AES

In this section, a subtle security flaw of the masking method [2] described in Section 3 is pointed out. In addition, it is argued that the multiplicative masking for AES is inherently vulnerable to the DPA attack.

The basic problem with the multiplicative mask is that it does not mask the all-zero byte value of data, that is, the all-zero byte remains unchanged after masking by a multiplicative mask. On the other hand, the all-zero (intermediate) data bytes cannot be avoided in AES. As a consequence, there are intermediate variables in the modified inversion scheme from Fig. 3 that are not masked completely. Namely, they satisfy the fundamental hypothesis for DPA by being correlated to a function depending on only 8 key bits and 8 plaintext bits.

More precisely, in the first round of the masked AES, the vulnerable intermediate variables are the byte input $Z_{1,j} = A_{1,j} \otimes Y_{1,j}$ and the byte output $Z_{2,j} = F(A_{1,j} \otimes Y_{1,j})$ of the block implementing the inversion in GF(256). Note that the data byte $A_{1,j}$ is given as $A_{1,j} = P_j \oplus K_{0,j}$ where P_j and $K_{0,j}$ are the corresponding plaintext and expanded key bytes, respectively. It follows that

$$K_{0,j} \;=\; P_j \;\;\Rightarrow\;\; Z_{1,j} \;=\; 0 \;\;\Rightarrow\;\; Z_{2,j} \;=\; 0. \qquad (3)$$

So, interestingly, it turns out that the (first-order) DPA attack on the masked AES can be mounted in essentially the same way as on the original AES without masking, which is described in Section 2. The difference is that one has to target the all-zero input byte or, equivalently, the all-zero output byte of F. In other words, for each of 256 possible values of the corresponding expanded key byte $K_{0,j}$, the power consumption curves for which $P_j = K_{0,j}$ are extracted and used for identifying the correct key. In this case $m = 8$, so that the DPA attack on AES without masking would not be effective, as explained in Section 2.

However, for the masked AES, the DPA attack will be able to distinguish between correct and incorrect guesses of the expanded key byte, because of the randomization effect provided by the random multiplicative mask. Moreover, if $K_{0,j}$ is guessed correctly, the peaks will appear because of the repeated simultaneous computation not only of the all-zero output byte $Z_{2,j}$, but also of the all-zero input byte $Z_{1,j}$. Altogether, the DPA attack may be more effective than the one on AES without masking, especially if one cannot find an effective way to simultaneously use more than just one fixed target m-bit value in the DPA attack on AES, where $m < 8$, or, more generally, a way to use (possibly optimal) partitions of power consumption curves into more than just two sets, provided that the power consumption models are available.

Now, the question is if the described DPA attack can be somehow prevented by using some other implementation of the multiplicative masking. For example, one may try to replace the modified inversion performed on $A_{i,j}$ by the modified inversion performed on some nonzero input byte whenever $A_{i,j} = 0$, and then to replace the computed output value by the desired one. However, detecting whether $A_{i,j} = 0$ and replacing the computed output value require specific computations that are themselves vulnerable to the DPA attack. In conclusion, it appears that the weakness of the multiplicative masking for AES is unavoidable.

Of course, it would be practically important, especially for applications where the space is very limited, to find another masking method that will not require the recomputation and storage of the S-boxes for every new run of AES. To this end, one has to use group or, more generally, quasigroup operations for masking the whole range of possible byte values which would at least simplify the secure computation or storage of modified F according to (2), where F is the inversion in GF(256). However, this does not appear to be very likely.

5 Conclusions

In the absence of any other more efficient and secure countermeasures against the DPA attack on AES, one can use random binary additive masks and accordingly recomputed S-boxes stored in RAM, for each new run of AES. In fact, it is proposed in [9] to recompute only one S-box and use it repeatedly during one execution of AES. A general rule is that increasing the number of mutually independent random masks increases the resistance against high-order DPA as well as against more sophisticated statistical analysis of power consumption curves.

With respect to the first-order and high-order DPA, it is critical to protect the first and the last few rounds of AES by random masks, whereas the protection of intermediate rounds may be useful with respect to more sophisticated statistical power analysis. In this regard, it is safer to repeat the masks in the intermediate rounds than in the first or the last few rounds.

Even if one uses the same recomputed S-box throughout the whole AES, it appears that the DPA attack is prevented as it targets the individual points of power consumption curves in time. However, such a solution may be vulnerable to a relatively simple second-order DPA attack similar to the one proposed in [10], especially for implementations in which the executions of S-box transformations are well separated in time (e.g., in software or low-space hardware). One has to identify the execution times of any two S-box transformations in the first round of AES, and then compare the power consumption curves at these two points by using some simple statistic. The corresponding two expanded key bytes have to be targeted simultaneously and the curves can be measured in a possibly chosen-plaintext scenario. To increase the security, it is then prudent to randomize the order of S-box transformations within a round, as already suggested in the literature.

References

[1] M.-L. Akkar, R. Bevan, P. Dischamp, and D. Moyart, "Power analysis, what is now possible...," Advances in Cryptology - Asiacrypt 2000, *Lecture Notes in Computer Science*, vol. 1976, pp. 489-502, 2000.

[2] M.-L. Akkar and C. Giraud, "An implementation of DES and AES, secure against some attacks," Cryptographic Hardware and Embedded Systems - CHES 2001, *Lecture Notes in Computer Science*, vol. 2162, pp. 309-318, 2001.

[3] S. Chari, C. Jutla, J. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," Advances in Cryptology - CRYPTO '99, *Lecture Notes in Computer Science*, vol. 1666, pp. 398-412, 1999.

[4] J.-S. Coron, P. Kocher, and D. Naccache, "Statistics and secret leakage," Financial Cryptography - FC 2000, *Lecture Notes in Computer Science*, vol. 1962, pp. 157-173, 2001.

[5] J.-S. Coron and L. Goubin, "On Boolean and arithmetic masking against differential power analysis," Cryptographic Hardware and Embedded Systems - CHES 2000, *Lecture Notes in Computer Science*, vol. 1965, pp. 231-237, 2000.

[6] J. Daemen and V. Rijmen, "AES proposal: Rijndael," `http://www.nist.gov/aes/`, 1999.

[7] L. Goubin and J. Patarin, "DES and differential power analysis: The duplication method," Cryptographic Hardware and Embedded Systems - CHES '99, *Lecture Notes in Computer Science*, vol. 1717, pp. 158-172, 1999.

[8] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," Advances in Cryptology - CRYPTO '99, *Lecture Notes in Computer Science*, vol. 1666, pp. 388-397, 1999.

[9] T. Messerges, "Securing the AES finalists against power analysis attacks," Fast Software Encryption - FSE 2000, *Lecture Notes in Computer Science*, vol. 1978, pp. 150-164, 2001.

[10] T. Messerges, "Using second-order power analysis to attack DPA resistant software," Cryptographic Hardware and Embedded Systems - CHES 2000, *Lecture Notes in Computer Science*, vol. 1965, pp. 238-251, 2000.