A DPA Resistant FPGA Implementation of AES Cryptosystem with Very Low Hardware Overhead

M. Masoumi

Abstract:
Differential Power Analysis (DPA) implies measuring the supply current of a cipher-circuit in an attempt to uncover part of a cipher key. Cryptographic security gets compromised if the current waveforms obtained correlate with those from a hypothetical power model of the circuit. During last years, there has been a large amount of work done dealing with the algorithmic and architectural aspects of cryptographic schemes implemented on FPGAs. However, there are only a few articles that assess their vulnerability to such attacks which, in practice, pose far a greater danger than algorithmic attacks. This paper first demonstrates the vulnerability of the Advanced Encryption Standard Algorithm (AES) implemented on a FPGA and then presents a novel approach for implementation of the AES algorithm which provides a significantly improved strength against differential power analysis with a minimal additional hardware overhead. The efficiency of the proposed technique was verified by practical results obtained from real implementation on a Xilinx Spartan-II FPGA.

Keywords: Advanced Encryption Standard Algorithm, Power Analysis Attacks, Field Programmable Gate Arrays,Power Attack Countermeasure.

1 Introduction

The fact that secret keys are now embedded into a number of devices means that the hardware becomes an attractive target for hackers. Although cryptosystem designers frequently assume that secret parameters will be manipulated in closed reliable computing environments, Kocher et al. reported in 1998 that microcomputers and microchips leak information correlated with the data handled and introduced a new kind of attacks which were radically different from software and algorithmic attacks [1]. These attacks use leaking or side-channel information, like power consumption data, electromagnetic emanations or computing time to recover the secret key. Because of the simplicity of these attacks, and the growing spread of applications which use cryptographic implementations, the importance of researching on this topic is still growing. Power analysis is an attack where the attacker obtains the information or secret key by measurements of the power consumption of the device during the execution of one encryption. There are two different degrees of sophistication involved in such power analysis, simple and differential [2, 3].

A Simple Power Analysis (SPA) attack is described as an attack where the attacker can directly use a power consumption of a cipher system to break a cryptosystem. However, a developer can easily protect a cryptosystem from SPA using random dummy codes or avoiding memory access by processing data in registers. In contrast, a Differential Power Analysis (DPA) attack is much harder to protect against, as it uses a statistical and error-correcting method to extract secret information from a power consumption signal. The secret key is guessed by using thousands or several hundreds of sample inputs and their corresponding power consumption traces. Random noises in power measurements can be filtered through the averaging process using a large number of samples. Protecting implementations against this efficient and sophisticated attack is of prime importance as it is the only attack which is not simply countered.

Another form of these attacks, the so called Correlation Power Analysis (CPA) technique based on the correlation between the real power consumption of the device and a power consumption model, has been widely studied in the literature [3, 4]. In recent years, the security of the Advanced Encryption Standard (AES) against DPA has received considerable attention and there is a growing interest in efficient and secure realization of the AES [5]. As a result of these attacks, numerous hardware and algorithmic countermeasures have been proposed. Unfortunately, most of these techniques are inefficient or costly or vulnerable to higher-order attacks [6]. They include randomized clocks, memory encryption/decryption schemes, power consumption randomization, and decorrelating the external power supply from the internal power

Iranian Journal of Electrical & Electronic Engineering, 2012.
Paper first received 5 Apr. 2011 and in revised form 4 Sep. 2011.
* The Author is with Islamshahr Azad University, P.O. Box: 33135-369, Sayad Shirazi Ave., Namaz Sq., Islamshahr, Tehran, Iran.
E-mail: m_masoumi@eetd.kntu.ac.ir.

consumed by the chip. Moreover, the use of different hardware logic, such as complementary logic, sense amplifier based logic (SABL), and asynchronous logic have been also proposed [7, 8]. Some of these techniques require about twice as much area and will consume twice as much power as an implementation that is not protected against power attacks. For example, the technique proposed in [9] adds area three times and reduces throughput by a factor of four. Another well-known method is masking which involves ensuring the attacker cannot predict any full registers in the system without making run-specific assumptions that are independent of the inputs to the system. This is achieved by applying a reversible random mask to the plaintext data before encryption with a modified algorithm. This makes exploiting data from several encryptions impossible as it would require guessing the correct mask for each run. Unfortunately, this method is costly or inefficient even if it has been demonstrated that it works. The main problem with masking methods is that they usually require an extra data path that works in parallel to compute the modification of the mask by the algorithm which considerably increases hardware overhead and decreases the throughput as it is seen in Fig. 1.

Most importantly, some masking techniques that were proposed were shown to be susceptible to higher order DPA attacks. Even techniques that were shown to be theoretically provably secure were susceptible to DPA using predictions based on simulations and a back-annotated netlist [10].

In this work, we concentrate on algorithmic countermeasures to protect AES against power attack and present a novel core implementation which is very simple and effective with very low hardware cost. This countermeasure is based on mathematical properties of Rijndael algorithm, and retains perfect compatibility with the published standard. We have studied the use of composite field techniques and isomorphism for Galois Field arithmetic in the context of protection of Rijndael against differential power attack. In order to experimentally verify the effectiveness of our proposed countermeasure we have implemented two versions of AES, a protected and an unprotected, on a Xilinx Spartan-II FPGA and compared the results of the implementation in the terms of resistance against attack, speed, area and throughput. While FPGAs are becoming increasingly popular for cryptographic applications, there are only a few articles that assess their vulnerability to such attacks [3]. In particular, very little work has been done on the resistance of FPGAs to hardware or system attacks, which, in practice, pose far a greater danger than algorithmic attacks. The results we obtained in this work are very encouraging compared with the results reported in the literature. The area overhead is only 7% with no decrease in speed or clock frequency or alteration in the algorithm. In addition, our technique directed at both hardware and software realizations and could be easily used in variety of platforms such as FPGAs, smart cards, DSPs or other security tokens. Most importantly, this work shows that it is possible to design algorithms to be inherently impervious to DPA. This article is organized as follows: The AES algorithm is briefly described Section 2. The principle of DPA attack will be described in section 3. Section 4 explains principles of the implementation of the AES using composite field arithmetic. In section 5 previous works is reviewed. Section 6 explains the new proposed approach.In section 7 measurement setup used for the implementation of the attack and the obtained results are described. Section 8 explains countermeasures, challenges and some open questions about differential power analysis. Finally, we summarize the results of our work in the conclusions.

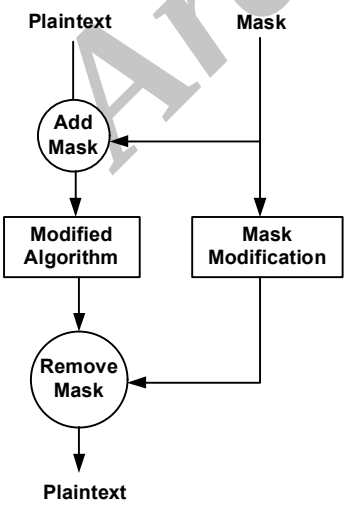

Fig. 1 Basic masking approach.

2. The AES Algorithm

AES has been developed and published by Daemen and Rijmen [11]. This algorithm is a byte-oriented symmetric block cipher, composed of a sequence of four primitive functions, Sub Bytes, ShiftRows, MixColumns, and AddRoundKey, executed round by round. Prior to each round AddRoundKey which combines the input with the cipher key is executed. In a 128-bit operation mode, at the start of the encryption, the message is divided to the blocks of length 128-bit and is copied to a 16 byte rectangular array called State. AddRoundKey is only a simple bit-wise XOR operation in which the elements of the State are XORed with RoundKey bit-by-bit. Sub Bytes is a non-linear bit-wise substitution of all bytes in the State. In Sub Bytes, each byte in the State is replaced by its corresponding byte in another table called S-Box. S-Box contains multiplicative inverse of all possible bytes over $GF(2^8)$ followed by an affine transformation.Eachbyte is an element of Galois field $GF(2^8)$ with irreducible polynomial $m(x)=x^8+x^4+x^3+x+1$. In the ShiftRows

transformation, each row of the state is considered separately and the bytes in that row are cyclically shifted to the left based upon the key-size of the algorithm. For the 128-bit key, the first row is unchanged. However, the second, third and fourth rows are shifted one, two, and three bytes respectively. The MixColumns transformation is a bricklayer permutation operating on each column of the State. In MixColumns, columns of the State are considered as a four-term polynomial over GF(2^8),then are multiplied with a fixed polynomial $c(x)=\{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$. Multiplications are performed modulo (x^4+1). The algorithm for the decryption has the same structure but uses mathematical inverses of the encryption steps, i.e. InvSubBytes, InvShiftRows, and InvMixColumns. The round keys are the same as those in encryption but are used in reverse order. Fig. 2 shows the standard implementation of the AES.

3 DPA against AES

In DPA, an attacker uses a so-called hypothetical model of the attacked device. The model is used to predict several values for the side-channel output of a device. This hypothetical model for the AES is one AddRoundKey and the SBox lookup of the first round which is fed with the plaintexts and one byte of the first subkey. The output of SubBytes is usually attacked in practice since that is the only function in AES in which data and cipher key enter a direct operation. These predictions are compared to the real, measured side-channel output of the device. Comparisons are performed by applying statistical methods on the data. Among others, the most popular are the distance-of-mean test and the correlation analysis. For the correlation analysis, the model predicts the amount of side-channel leakage for a certain moment of time in the execution. These predictions are correlated to the real side-channel output. This correlation can be measured using the Pearson correlation coefficient. Let t_i denotes the i^{th} measurement data (i.e. the i^{th} trace) and T the set of traces. Let p_i denote the prediction of the model for the i^{th} trace and P the set of such predictions [3]. Then we calculate:

$$C(T,P)=\frac{E(T.P)-E(T)E(P)}{\sqrt{Var(T).Var(P)}} \quad (1)$$

Here $E(T)$ denotes the expectation (average) trace of the set of traces T and $Var(T)$ denotes the variance of a set of traces T. In practice it is not possible to know the true values for the covariance or standard deviation of variables, only calculate approximations of them based on the values discovered through experiments. If this correlation is high, it is usually assumed that the prediction of the model, and thus the key hypothesis, is correct. The scenario of DPA attack based-on distance-of-mean test is as follows. At first N plaintexts are randomly generated. Power consumption measures are taken for each plaintext. As before, a hypothetical model of the AES is fed with the plaintexts and one byte of the first subkey. Only the SBox of the first round is targeted by the attacker since that is the only function in AES in which data and cipher key enter a direct operation (See Fig. 3). To this output hypothesis, a selection function D is applied. This selection function divides the measures in two sets. One that the selection function returns one and the other for that returns zero. For each set the average is computed. Then, the difference between the two averages is calculated. This leads to 2^8 differential curves. Only for the correct subkey the selection function has worked properly and there will be well seen spikes in an otherwise flat curve [1]. High-order DPA uses more general DPA selection functions to perform differential power analysis. High-order attacks require using multiple samples in a single power trace to compute a DPA power trace value. Using multiple samples is analogous to second order or higher digital signal processing with memory. An attacker can

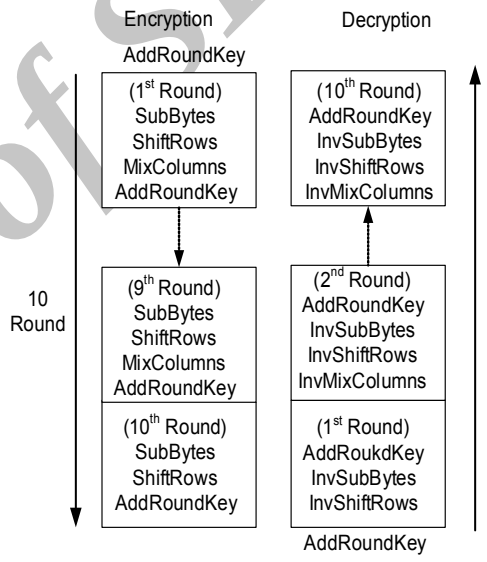

Fig.2 Standard implementation of the AES algorithm.

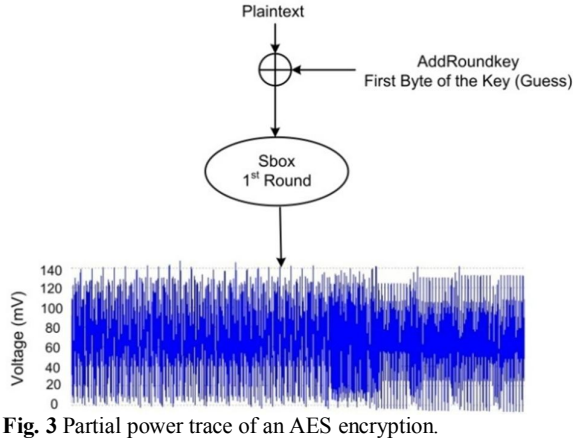

Fig. 3 Partial power trace of an AES encryption.

mount a second order attack by computing joint statistics on power signatures at different sections on the encryption code. One drawback to high-order DPA is increased memory and processor requirements because of the need to store multiple samples for a single DPA computation. The required trace equipment for high-order DPA is identical to the trace equipment required for SPA and DPA, but more sophisticated post-processing requires additional off-line resources. Knowledge of the encryption algorithm and specific implementation is more critical in high-order DPA than first-order. Attackers need to know specific points of execution where joint statistics can be meaningfully computed.

4 AES and Composite Field Arithmetic

The SubBytes and the InvSubBytes in the AES algorithm are traditionally implemented by look-up tables (LUT). Non-LUT-based approaches, which employ combinational logic only, such as the composite field (or tower field) inversion over $GF(2^8)$ are used to avoid the unbreakable delay of LUTs, and it can be used to create compact AES implementations [12]. Composite field arithmetic can be employed, such that the field elements of $GF(2^8)$ are mapped to elements in some isomorphic composite fields, in which the field operations can be implemented by lower cost subfield operations. The two pairs $\{GF(2^n), Q(y)\}$ and $\{GF(2^n)^m), P(x)\}$ constitute a composite field if $GF(2^n)$ is constructed from $GF(2)$ by $Q(y)$ and $GF((2^n)^m)$ is constructed from $GF(2^n)$ by $P(x)$, where $Q(y)$ and $p(x)$ are polynomials of degree n and m respectively. The fields $GF((2^n)^m)$ and $GF(2^k)$, $k = nm$, are isomorphic to each other.

The most costly operation in the SubBytes is the multiplicative inversion over a field A (the AES field), where A is extended from of $GF(2)$ with the irreducible polynomial $m(x) = x^8+x^4+x^3+x^2+1$. To reduce the cost of this operation, the following 3-stage method is adopted:

Stage 1 :Map all elements of the field A to a composite field B, using an isomorphism function δ.

Stage 2 :Compute the multiplicative inverses over the field B.

Stage 3 :Re-map the computation results to A, using the function δ^{-1}. Fig. 4 shows the outline of an S-Box implementation using the composite field technique.

To reduce the cost of Stage 2 as much as possible, it is known to be efficient to construct the composite field B using repeated degree-2 extensions under a polynomial basis using these irreducible polynomials [Ĥ]Ĥ

$$\begin{cases} GF(2^2) & \rightarrow P_0(x) = x^2 + x + 1 \\ GF\left((2^2)^2\right) & \rightarrow P_1(x) = x^2 + x + \phi \\ GF(((2^2)^2)^2) & \rightarrow P_2(x) = x^2 + x + \lambda \end{cases} \quad (2)$$

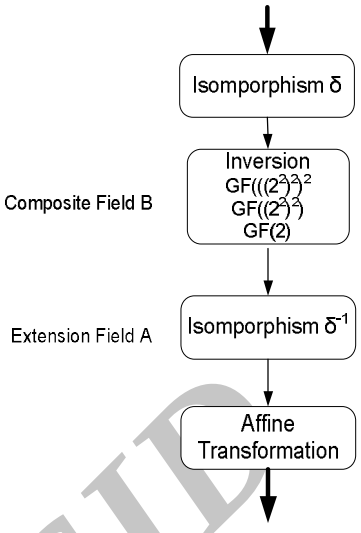

Fig. 4 Computation sequences of composite-field-based S-Box.

where $\Phi = \{10\}_2$, $\lambda = \{1100\}_2$. The isomorphism functions δ and δ-1 in Stages 1 and 3 are constructed as follows. The δ (and δ^{-1}) can be found as follows.

$$\delta = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\delta^{-1} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Let q be the element in $GF(2^8)$, then the isomorphic mappings and its inverse can be written as δ^*q and $\delta^{-1}*q$, which is a case of matrix multiplication as shown below, where q_7 is the most significant bit and q_0 is the least significant bit. The matrix multiplication can be translated to logical XOR operation as is shown at top of the next page.

Thus, the multiplicative inversion in $GF(2^8)$ can be carried out in $GF((2^4)^2)$ by the architecture illustrated in Fig. 5.

$$\delta \times q = \begin{bmatrix} q_7 \oplus q_5 \\ q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_6 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_6 \oplus q_4 \oplus q_1 \\ q_6 \oplus q_1 \oplus q_0 \end{bmatrix}$$

$$\delta^{-1} \times q = \begin{bmatrix} q_7 \oplus q_6 \oplus q_5 \oplus q_1 \\ q_6 \oplus q_2 \\ q_6 \oplus q_5 \oplus q_1 \\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1 \\ q_5 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_5 \oplus q_4 \\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_0 \end{bmatrix}$$

The multipliers in *GF(2⁴)* can be further decomposed into multipliers in *GF(2²)* and then to *GF(2)*, in which a multiplication is simply an AND operation. Fig. 6 illustrates this decomposition, together with the other blocks used in Fig. 5 except the inversion in *GF(2⁴)* block.

4.1 Composite Field Arithmetic Operations

Any arbitrary polynomial can be represented by $bx+c$ where b is upper half term and c is the lower half term [12]. Therefore, from here, a binary number in Galois Field q can be spilt to $q_H x+q_L$. For instance, if $q=\{1011\}_2$, it can be represented as $\{10\}_2 x+\{11\}_2$, where q_H is $\{10\}_2$ and $q_L=\{11\}_2$. q_H and q_L can be further decomposed to $\{1\}_2 x+\{0\}_2$ and $\{1\}_2 x+\{1\}_2$ respectively. The decomposing is done by making use of the irreducible polynomials introduced at (2). Using this idea, the logical equations for the addition, squaring, multiplication and inversion which were shown in Fig. 5 can be derived. Detailed explanation of the implementation of all these blocks is out of the scope of this article. However, to clarify the subject to the readers to better understand how these modules can be implemented using combinational logic, the realization of constant multipliers ($\times\lambda$) is briefly illustrated. For the purpose of practicality, the depth of the mathematics involved has been reduced in order to allow the reader to better figure out the internal operations within the S-Box.

Let $k = q\lambda$, where $k = \{k_3\ k_2\ k_1\ k_0\}_2$, $q = \{q_3 q_2 q_1 q_0\}_2$ and $\lambda = \{1100\}_2$ are elements of *GF(2⁴)*.

$$k = \{k_3 k_2 k_1 k_0\} = k_H x + k_L = \{q_3 q_2 q_1 q_0\}(1100)$$

Where $\lambda_H = \{11\}_2$ and $\lambda_L = \{00\}_2$

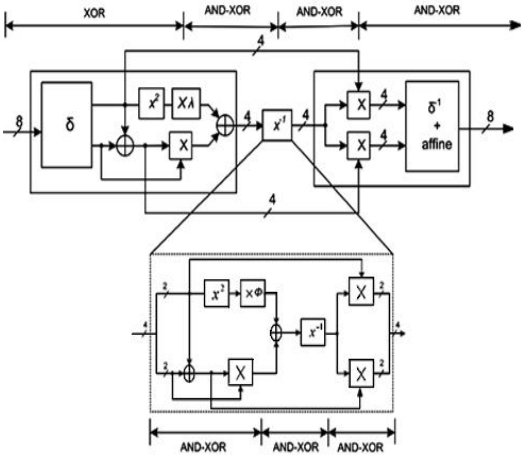

Fig. 5 Implementation of SubBytes transformation using composite field arithmetic [12].

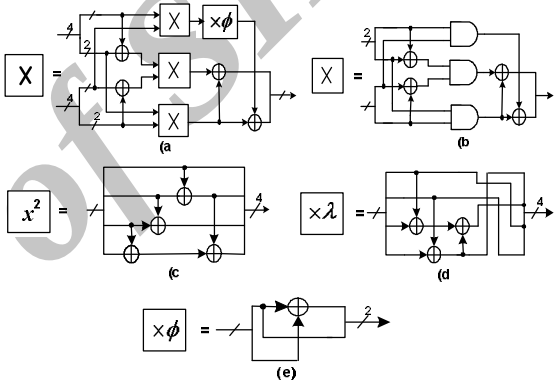

Fig. 6 Implementation of individual blocks, (a) multiplier in *GF(2⁴)*, (b) multiplier in *GF(2²)*, (c) squarer in *GF(2⁴)*, (d) constant multiplier ($\times\lambda$); and (e) multiplier($\times\Phi$) [12].

$$k = (q_H x + q_l)(\lambda_H x + \lambda_l)$$

λ_l can be canceled out since $\lambda_l = \{00\}_2$

$$k = q_H \lambda_H x^2 + q_l \lambda_H x$$

Modulo reduction can be performed by substituting $x^2=x+\varphi$ using the irreducible polynomial in (2) to yield the expression below.

$$k = q_H \lambda_H (x+\varphi) + q_L \lambda_H x$$
$$k = (q_H \lambda_H + q_l \lambda_H)x + (q_H \lambda_H)\varphi \in GF(2^2)$$

k_H and k_L terms can be further broken down to *GF(2)*.

$$k_H = q_H \lambda_H + q_L \lambda_H$$
$$k_H = (q_3 q_2)(11)_2 + (q_1 q_0)(11)_2$$
$$k_H = (q_3 x+q_2)(x+1)+(q_1 x+q_0)(x+1)$$
$$k_H = q_3 x^2+(q_3+q_2)x+q_2+q_1 x^2+(q_1+q_0)x+q_0 \qquad (3)$$

Substituting $x^2 = x+1$, would then yield the following.

$$k_H = q_3(x+1) + (q_3+q_2)x + q_2+q_1(x+1)+(q_1+q_0)x+q_0$$
$$k_H = (q_3+q_3+q_2+q_1+q_1+q_0)x+(q_3+q_2+q_1+q_0)$$
$$k_3 x+k_2 = (q_2+q_0)x + (q_3+q_2+q_1+q_0) \in GF(2)$$

The same procedure is taken to decompose k_L to $GF(2)$.

$k_L = q_H \lambda_H \varphi$
$k_L = (q_3 q_2)(11)_2(10)_2$
$k_L = (q_3 x+q_2)(x+1)x$
$k_L = q_3 x^3 + q_2 x^2 + q_1 x + q_0$

Again, the x^2 term can be substituted since $x^2 = x + 1$. Likewise, x^3 is also substituted with $x^3 = 1$. So, we have:

$k_L = q_3(1) + q_2(x+1) + q_3(x+1) + q_2 x$
$k_L = (q_3+q_2+q_1)x + (q_3+q_2+q_1)$
$k_1 x + k_0 = (q_3)x + (q_2) \in GF(2)$ (4)

From equations (3), (4) combined, the formula for computing multiplication with constant λ is shown below.

$$k_3 = q_2 \oplus q_0$$
$$k_2 = q_3 \oplus q_2 \oplus q_1 \oplus q_0$$
$$k_1 = q_3$$
$$k_0 = q_2 (5)$$

From the equation (5), the architecture for multiplication with constant λ can be depicted as is shown in Fig. 7 ($\Phi = \{10\}_2, \lambda = \{1100\}_2$).

The same method could be used for multiplication with constant λ when $\lambda = \{1111\}_2$. Fig. 8 shows this architecture. Other architectures shown in Fig. 5 can be implemented simply using the same technique. As it is seen any change in $\{\Phi, \lambda, \delta, \delta^{-1}\}$ could be easily translated to Boolean logic and changing the SBox architecture by using this method has a very low hardware cost.

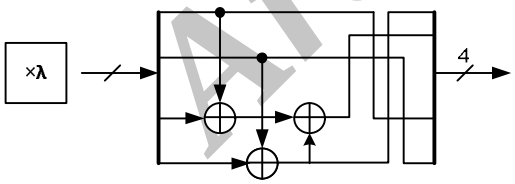

Fig. 7 Hardware implementation of the multiplication by constant λ (λ = 12).

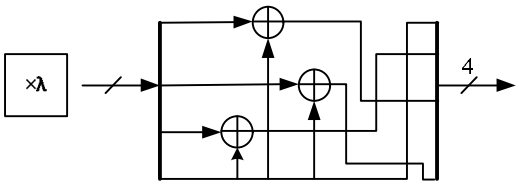

Fig. 8 Hardware diagram for multiplication with constant λ (λ=15).

As another example consider multiplication with constant ($\times\varphi$) in GF(2^2) where $\varphi = \{10\}_2$. Let $k = q\varphi$ where $k=\{k_1 k_0\}_2$, $q = \{q_1 q_0\}_2$ and $\varphi = \{10\}_2$ are elements of GF(2^2).

$k = k_1 x + k_0 = (q_1 q_0)(10)_2 = (q_1 x + q_0)(x)$
$k = q_1 x^2 + q_0 x$

Substitute the term x^2 with $x+1$, yield the expression below.

$k = q_1(x+1) + q_0 x$
$k = (q_1+q_0)x + (q_1) \in GF(2)$

hence, the formula for computing multiplication with φ can be derived and is shown below.

$k_1 = q_1 + q_0$
$k_0 = q_1$

Hardware implementation of the multiplication with $\varphi = \{10\}_2$ is shown in Fig. 9.

The hardware implementation of the multiplication with $\varphi = \{11\}_2$ can be obtained as follows.

$k = k_1 x + k_0 = (q_1 q_0)(11_2) = (q_1 x + q_0)(x+1)$
$k = q_1 x^2 + (q_0 + q_1)x + q_0$

Substitute the term x^2 with $x+1$, yield the expression below.

$k = q_0 x + (q_0 + q_1) \in GF(2)$

Hence, the formula for computing multiplication with $\varphi = \{11\}_2$ can be derived and is shown in Fig. 10.

$k_1 = q_0$
$k_0 = q_0 + q_1$

For the implementation of MixColumn/InvMixColumn, the architecture proposed in [12] was used, in which the "XTime" block implements the constant multiplication by {02} in GF(2^8), the "X4Time" block computes the constant multiplication of $\{04\}_{16}$ and can be implemented by two serially concatenated "XTime" block and $S_{i,c}$ denotes the i-th byte of the c-th column of the state matrix. Fig. 11 illustrates the architecture for efficient implementation of MixColumn/InvMixColumn.

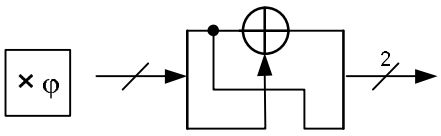

Fig. 9. Hardware implementation of the multiplication with $\varphi = \{10\}_2$.

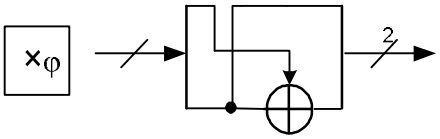

Fig. 10. Hardware implementation of the multiplication with φ = $\{11\}_2$.

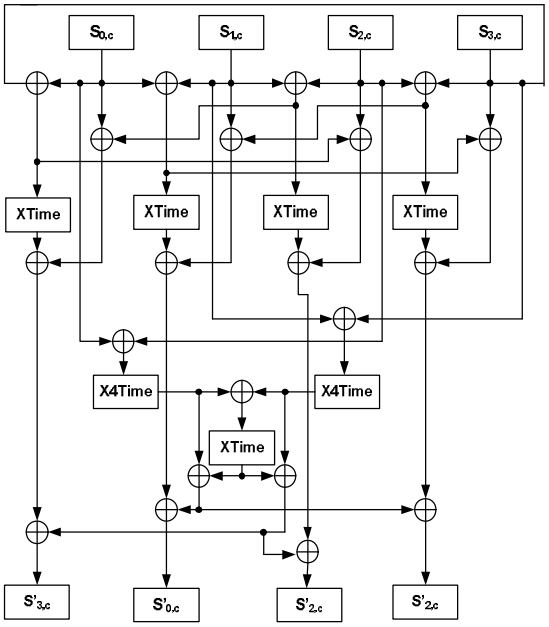

Fig. 11 Efficient implementation of MixColumns/ InvMixColumns [12].

5. Previous Works

In [13] Rostovtsev and Shemyakina describe using ofisomorphisms of the underlying finite field. But as the authors admit, their method has comparatively small efficiency. The technique proposed in [14] randomizes power consumption of SBox by randomly choosing irreducible generator polynomials of the field $GF(2^8)$. The approach is interesting but introduces significant hardware cost with almost 300% increase in area and 60% decrease in speed. Another method based on univariate polynomials of Blomeret. al. has been illustrated in [15]. This can be seen as a perfectly general method that can be applied to any S-box, as any function over a finite field can be seen as a univariate polynomial. This makes the method more efficient than in the general case and suitable for Rijndael. However, it suffers from considerable decrease in performance. Tower Fields Methods by Oswald, et al. [16, 17] are designed for hardware implementations. In these methods, the computing of Inv in $GF(2^{2k})$ is reduced to a secure computation with masked values of multiplications and inverses in $GF(2^k)$, by representing $GF(2^{2k})$ as a quadratic extension of $GF(2^k)$. Multiplications can be computed with additive masking and we are left with the problem of a secure computation of Inv at the lower level. In [18] Schramm proposes mask multipliers for $GF(2^2)$ and $GF(2^4)$ which are used in the masked composite field-based AES SBox for software applications. However, this approach needs 1536 bytes ROM to store the masked SBox and takes 13600 clock cycles to implement masked AES encryption while a similar unmasked AES realization takes 800 cycles.

6. New Proposed Approach

In the power analysis, the key detection is possible because of the dependency between the power consumption of devices and the intermediate values of the cryptographic algorithms. Therefore, if we want to prevent from these kinds of attacks, this dependency should be broken. Considering this fact, we have studied the structure of SBox and based on our knowledge of math, the three parameters $\{\Phi, \lambda, \delta\}$ are not constant since there could be much more than one eligible isomorphism (see Figs. 4 and 5). The polynomial $P_1(x) = x^2 + x + \phi$ is an irreducible polynomial over the field GF(2^2) if and only if $1<\Phi<4$, and $P_2(x) = x^2 + x + \lambda$ is irreducible over $GF(2^4)$ if and only if $7<\lambda<16$. So, there are 16 different combinations for $\{\phi, \lambda\}$. We can also search in the set of linear transformations from $GF(2^8)$ to itself or all autoisomorphism over $GF(2^8)$ and it can be proved that there are $\prod_{i=0}^{7}(2^8 - 2^i)$ of such transformations. So we can have the same number of different δ/δ^{-1} [19, 20] and considerable number of sets $\{\Phi, \lambda, \delta, \delta^{-1}\}$ for Rijndael. It must be mentioned that all of the possible combinations of $\{\Phi, \lambda, \delta, \delta^{-1}\}$ will not result in an appropriate field isomorphism. We have found 32 different suitable sets of the mentioned parameters which can be implemented with minimum combinational logic. For example, these three following sets have such characteristics. The elements of matrices δ and δ^{-1} and the values of Φ and λ are represented in decimal.

$\Phi = 2, \lambda = 15$
$\delta = \{160, 126, 114, 162, 182, 84, 16, 217\}^T$
$\delta^{-1} = \{46, 28, 174, 2, 122, 26, 144, 75\}^T$

$\Phi = 3, \lambda = 12$
$\delta = \{160, 222, 172, 174, 202, 238, 44, 227\}^T$
$\delta^{-1} = \{102, 212, 230, 162, 10, 234, 176, 233\}^T$

$\Phi = 3, \lambda = 10$
$\delta = \{160, 126, 172, 2, 20, 132, 130, 99\}^T$
$\delta^{-1} = \{190, 132, 62, 106, 98, 2, 112, 141\}^T$

Therefore, one can make a random isomorphism by generating different sets of $\{\Phi, \lambda, \delta, \delta^{-1}\}$ and randomly choosing them in each block encryption/decryption. As described, two ciphers are isomorphic if they produce the same output for the same input. Hence, from the architectures shown in Figs. 4 and 5, it is obvious that we can have many SBoxes with similar input/output pairs and different internal structures. As it was shown in section 4, it is simple to realize and select different architectures of SBox based on different values of $\{\Phi, \lambda, \delta, \delta^{-1}\}$ since such changes can be easily realized by Boolean gates only (please see figures 5-10). As a result, for a specific input, there will be different power consumption patterns while the output remains the

same. Since different binary values are used each time, power traces collected for DPA will be weakly correlated to the data being manipulated and it will be harder to extract the secret key or any other sensitive information, even if many runs are performed for the same inputs. The proposed technique effectively reduces the signal to noise ratio (SNR) of the performed operations. As we know, the SNR quantifies how much information is leaking from a power trace. The most important features of the proposed method are that it does not change the mathematical properties of SBox at all, do not decrease the working frequency and has very low area overhead because any change in the architecture of Figs. 5 and 6 can be easily realized by Boolean gates only. In addition, it is very simple compared with those presented in the literature and fully complies with the published standard. No additional parameters than the secret key and the data to be processed are needed. The operation of the system is as follows: at first all 32 (or more) possible sets of values for $\{\Phi, \lambda, \delta, \delta^{-1}\}$ are produced and stored. In our prototype, a linear feedback shift register (LFSR) is used as a pseudorandom number generator for proof of the concept. Encryptor and decryptor agree on the initial state of this LFSR as they agree on the cipher key. In each block encryption/decryption, one set is chosen by pseudorandom number generator and one of the 32 corresponding SBox architectures (as was shown in Figure 5) is selected randomly. This results in very low area overhead with no decrease in speed or clock frequency since no hardware module has been added to the critical path. In order to prevent any power information leakage when the output of SBox is stored in the output register, another random number generator selects two other δ^{-1} matrices randomly and multiplies the pre-final value to them concurrently in a parallel path. This make power consumption pattern of the final stage of SBox computation indistinguishable for the attacker and prevents the adversary to mount a successful attack on this point. This concurrent multiplication can be easily performed in FPGA since it is one of the inherent advantages of FPGA. The simplicity of this approach makes it suitable for both hardware and software applications. Since the mentioned parameters could be easily stored inside the device, the proposed approach could be used in smart cards, digital signal processors or other security tokens.

7 Attack on a Real System

To investigate the effect on efficiency of implementing a device using the new style, both regular AES and the new design were realized in a loop architecture in which only one block of data is being processed at a time, using Verilog HDL and were synthesized on a Xilinx Spartan-II FPGA. Both used 128-bit keys. The encryptor core structure in regular AES occupied 983 CLB slices (41%), on targeting Xilinx Spartan-II FPGA device while this number for the modified implementation with 32 randomly selected SBoxs was 1050. This means that the total area cost for the implementation is almost 7%, much lower than the results reported in the literature so far. The clock speed in both cases was almost 42.2 MHz. Our measurement setup, as it is shown in Figure 12, consists of the FPGA board, an Agilent MSO7034A sampling oscilloscope with a 2 GS/sec and BW=350 MHz. The board uses two separate power supplies, a 3.3 V supply for I/O and a 2.5V supply for the core cells. Only the core power supply was measured. A small resistor (10 ohm) was inserted between the FPGA board and the power supply. In order to reduce switching noise and to improve the accuracy, the working frequency of FPGA board was lowered to 1 KHz and all measurements were averaged over ten times.

In order to improve the SNR and accuracy of the attack, instead of considering a mono-it, a set of four bit was considered, i.e. our selection function returns one when hamming weight of the output of SBox is greater than four, and otherwise it returns zero. It has been shown that the efficiency of the attack is increased in such a case since the ghost peaks and secondary peaks are lowered [21].

The experimental results for the differential power traces for the correct and a wrong subkey guesses are shown in Fig. 13 and Fig. 14 respectively. As it is seen, the plots confirm the assumption about the measurability of Hamming-Weights leakage. We need approximately 1,000 measurements to identify the correct key.

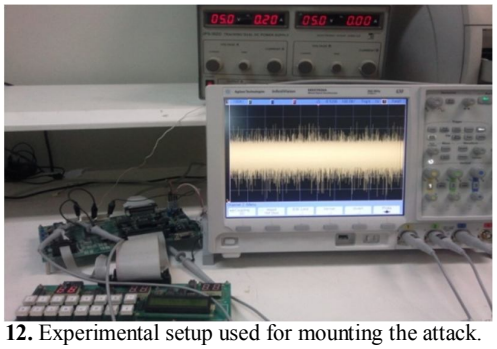

Fig. 12. Experimental setup used for mounting the attack.

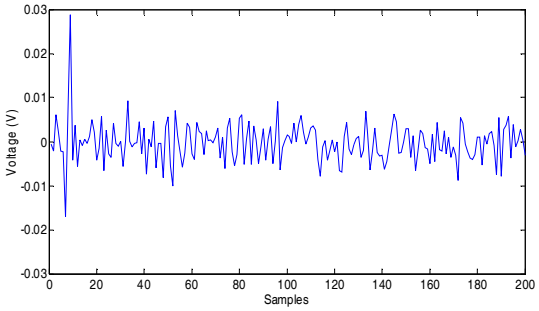

Fig. 13 Differential power traces for the correct subkey guess in the unprotected implementation.

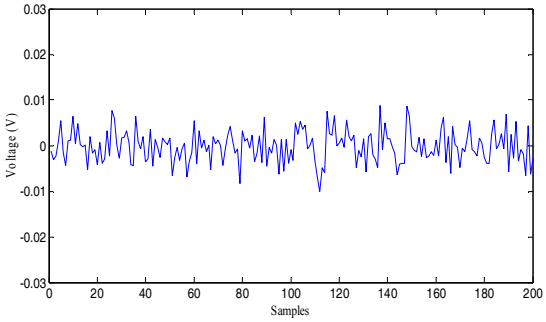

Fig. 14 Differential power traces for a wrong subkey guess in the unprotected implementation.

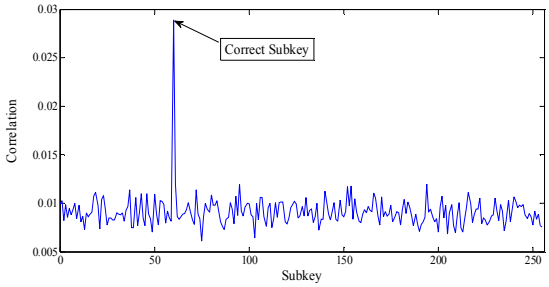

Fig. 15 Recovering the correct subkey using a correlation attack with real measurements.

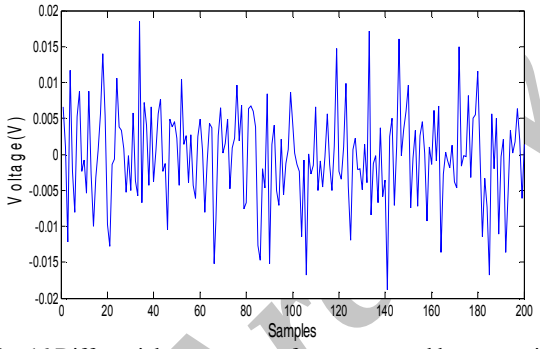

Fig. 16 Differential power traces for a correct subkey guess in the modified implementation.

The results of calculation of correlations for the first subkey are shown in Fig. 15, in which the correct value, *0x3C*, appears as a clear peak. Our experiments showed that recovering the full 128-bit key with this method takes almost two hours with an Intel 2.5 GHz quad processor computer.

The setup was repeated using an implementation of the new algorithm. Fig. 16 and Fig. 17 show the differential power traces for the correct and wrong key guesses in the protected version respectively. As it is seen the correct key cannot be distinguished from the wrong key as there is more much noise in the system. Fig. 18 and Fig. 19 show two different patterns of powertraces when inputting the same plaintext. Fig. 20 shows the correlations for each possible key guess in the modified implementation after 1000 traces. This time the correct value is completely obscured by other values and there are no clear peaks, just a band of random values. The experiment was repeated with a new key schedule, this time 6000 power traces were recorded. Again, the correct value was not distinguishable from the incorrect ones.

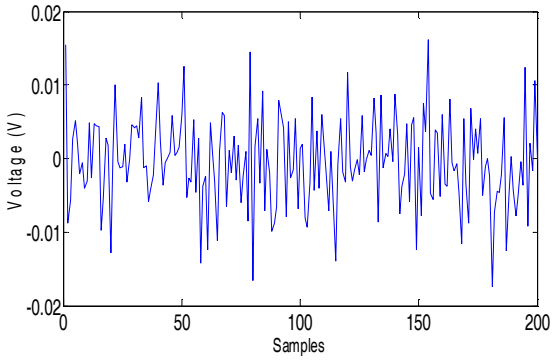

Fig. 17. Differential power traces for a wrong subkey guess in the modified implementation.

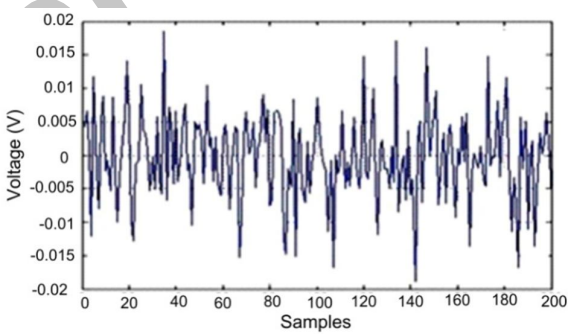

Fig. 18. Differential power traces for a known plaintext in the modified implementation with a specific set of parameters $\{\Phi, \lambda, \delta, \delta^{-1}\}$.

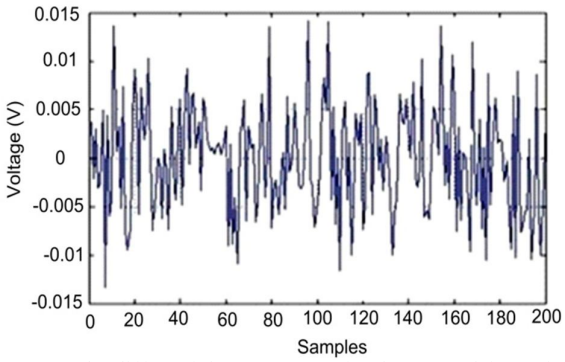

Fig. 19. The differential power traces for the same plaintext in the modified implementation with another set of parameters $\{\Phi, \lambda, \delta, \delta^{-1}\}$.

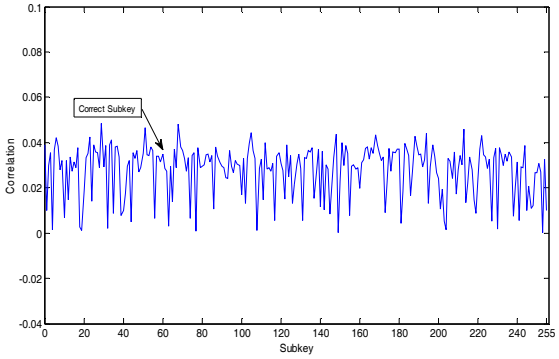

Fig. 20 Result of correlation analysis for recovering the correct subkey in the modified implementation.

8. Countermeasures and Challenges

In recent years, there has been a lot of work done to prevent side-channel attacks. The methods can generally be divided into software and hardware countermeasures, with the majority of proposals dealing with software countermeasures. "Software" countermeasures refer primarily to algorithmic changes, such as masking of secret keys with random values, which are also applicable to implementations in custom hardware or FPGA. Hardware countermeasures often deal either with some form of power trace smoothing or with transistor-level changes of the logic. Neither seems to be easily applicable to FPGAs without support from the manufacturers. However, some proposals such as duplicated architectures might work on today's FPGAs. Many of the mitigations that are intuitively put forward, such as the randomization of the execution sequence or the addition of a random power consuming module or a current sink, hardly improve the resistance against the power attacks [22]. In the present state-of-the-art, the countermeasures try to make the power consumption of the cryptographic device independent of the signal values at the internal circuit nodes by either randomizing or flattening the power consumption. None of the techniques, however, provides perfect security instead they increase the required number of measurements. As mentioned, randomizing the power consumption is done with masking techniques that randomize the signal values at the internal circuit nodes while still producing the correct cipher text. This can be done at the algorithmic level where a random mask is added to the data prior to the encryption and removed afterwards without changing the encryption result (*e.g.* [23]) or at the circuit level where a random mask-bit equalizes the output transition probabilities of each logic gate [24], [25]. Flattening the power consumption is done at the circuit level such that each individual gate has a quasi data-independent power dissipation. This is done with dynamic differential logic, sometimes also referred to as dual rail with precharge logic to assure that every logic gate has a single charging event per cycle [26]. In self-timed asynchronous logic [27], the terminology refers to dual rail encoded data interleaved with spacers.

The increased power attack resistance does not come for free. The algorithmic level masking has a factor 1.5 overhead when compared with a regular (unprotected) design [23]. The masked logic styles have a factor 2 and 5 area overhead [24], [25]. The dual rail logic styles have a factor 3 area overhead [28]. Yet, the figures for the algorithmic and logic masking do not include the random number generator. It is thus important that the full implementation cost of a countermeasure is clearly communicated and taken into account for evaluation. Several techniques have been proposed to reduce the area overhead. For instance, custom logic cells can be made more compact than compound standard logic cells and security partitioning reduces the part of the design that has to be protected [28].The fundamental question here is that can these techniques be optimized further or can a breakthrough mitigation technology be developed with a lower overhead?

Yet, one has to be careful to declare a (new) mitigation as secure. A visual inspection, or even the standard deviation of the power consumption, does not provide any indication [29]. Thus far, the best figure of merit is probably the required number of measurements for a successful attack on a realistic circuit. The success of an attack, however, depends both on the information in the power consumption and on the strength of the attacker, which encompasses the measurement setup but also the leakage estimation and the statistical technique used. Indeed, if the power estimation is more accurate, the attack will be more successful. The statistical analysis technique to compare the measurements with the estimations is also important: the difference of means test requires more measurements then the correlation test, which requires more measurements then the Bayesian classification. Some work has been done to distinguish the quality of an implementation from the strength of a side-channel adversary [24], but it is not clear how in a practical way a design can be evaluated without an attack and with abstraction of the statistical tool or distinguisher. Another important question here is that can an expression be found that based on design parameters such as the activity factor or a power consumption profile indicates the strength of a design? These are some examples of open questions and open research areas about the use of FPGA as a module of security. Of course, replying these questions is an important, challenging and motivational task.

9. Conclusions

As a conclusion, it can be said that even an encryption standard that has proved its robustness theoretically, may be vulnerable when implemented on hardware devices such as FPGAs. This work confirmed that power analysis has to be considered as a serious threat for FPGA security. We were able to recover the 128-bit secret key in almost two hours. Although certain

features of FPGAs make the practical implementation of power attacks significantly harder than in the smart card context, we have conduced relevant experimental tests that led us to find the encryption key. In addition, a novel AES implementation with a simple and integrated countermeasure against DPA was presented. The countermeasure is based on mathematical properties of Rijndael algorithm and retains perfect compatibility with the published Standard. The new design permits the construction of actual cores with very efficient area and speed characteristics, while still keeping a high protection level. Many solutions would allow improving our measurements and a lot of questions concerning the physical security of FPGAs remain open. The efficiency of the attack may be improved by combining this method with other side-channel technique. As many state-of-the-art cryptosystems use FPGA and processors simultaneously, successful implementation of DPA on such systems may be an interesting research subject for both academia and industry. Protecting against the attacks exploiting the information, however, can be a challenge, is costly and must be done with care.

References

[1] Kocher P., Jaffe J. and Jun B., "Differential power analysis", *Advances in Cryptology- Crypto 1999*, LNCS, Vol. 1666, pp. 388-397, Springer-Verlag, 1999.

[2] Coron, J., Prouff E. and Rivain M., "Side channel cryptanalysis of a higher order masking scheme", CHES 2007, *LNCS*, Vol. 4727, pp. 28–44, Springer-Verlag.

[3] Masoomi M., Masoumi M. and Ahmadian M., "A practical differential power analysis attack against an FPGA implementation of AES cryptosystem", *IEEE I-Society*, London, UK, 2010.

[4] Standaert F. X., Peeters E., Mace F., and Quisquater J. J., "Updates in security of FPGA against differential power attacks", *ARC2006*, LNCS 3958, pp. 335-346, Springer-Verlog, 2006.

[5] Fereidunian A., Lesani H., Lucas C., Lehtonen M. and Nordman M. M., "A Systems Approach to Information Technology (IT) Infrastructure Design for Utility Management Automation Systems", *Iranian Journal of Electrical and Electronic Engineering*, Vol. 2, No.3, pp. 91-104, 2006.

[6] Courtois N. T. and Goubin L., "An Algebraic Masking Method to Protect AES against Power Attacks", *ICISC 2005*, LNCS 3935, pp. 199–209, Springer-Verlag, 2005.

[7] Mirhosseini S. and Ayatollahi A., "A low-voltage, low-power, two-stage amplifier for switched-capacitor applications in 90 nm CMOS process", *Iranian Journal of Electrical and Electronic Engineering*, Vol. 6, No. 4, pp. 199-204, 2010.

[8] Karimi, G. R. and Mirzakuchaki S., "Behavioral modeling and simulation of semiconductor devices and circuits using VHDL-AMS", *Iranian Journal of Electrical and Electronic Engineering*, Vol. 4, No. 4, pp.165-175, 2008.

[9] Moore S., Anderson R., Mullins R., G. Taylor, and Fournier J. J. A., "Balanced self-checking asynchronous logic for smart card applications", *J. of Microp. and Microsystems*, Vol. 27, No. 9, pp. 421-430, 2003.

[10] Ors S. B. and Oswald E., "Power analysis attacks against FPGA, first experimental results", *CHES 2003*, LNCS 2779, pp. 35-50, Springer-Verlag, 2003.

[11] Daemen J. and Rijmen V., *AES Proposal Rijndael*, National Institute of Standards and Technology, July 2001.

[12] Zhang X. and Parhi K. K., "High-Speed VLSI architectures for the AES algorithm", *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, Vol. 12, No. 9, pp. 957-967, 2004.

[13] Rostovtsev A. G. and Shemyakina O. V., "AES side channel attack protection using random isomorphisms", Available on: *http://eprint.iacr.org/2005/087.pdf*.

[14] Ghellar F. and Lubaszewski M. S., "A novel AES cryptographic core highly resistant to differential power attack", *BCCI'08*, pp. 29-35, Gramado, Brazil, 2008.

[15] Blömer J., Guajardo J. and Krummel V., "Provably secure masking of AES", *in Selected Areas in Cryptography Workshop*, pp. 69-83, 2004.

[16] Oswald, E., Mangard, S. and Pramstaller, N., "Secure and efficient masking of AES -amission impossible?", *eprint.iacr.org/2004/134*.

[17] Oswald, E., Mangard, S., Pramstaller, N. and Rijmen, V., "A side-channel analysis resistant description of the AES S-Box", FSE 2005, pp. 413-423, 2005.

[18] Schramm K., *Advanced methods in side-channel cryptanalysis*, Ph. D. Thesis, University of Bochum, Germany, 2006.

[19] Xiao L. and Heys. H. M., "Hardware design and analysis of block cipher components", *in 5th Int. Conf. on Information Security and Cryptology (ICISC'02)*, Seoul, Korea, November, 2002.

[20] Lidl R. and Niederreiter H., *Introduction to finite field and applications*, Cambridge University Press, 1986.

[21] Ha Lee, Canovas C. and Cledier J., "An overview onside-channel analysis attacks" *ASIACCS'08*, pp. 33-43, March, 18-20, 2008.

[22] Clavier C., Coron J. and Dabbous N., "Differential power analysis in the presence of hardware countermeasures" *CHES 2000*, pp. 252-263, 2000.

[23] Pramstaller N., Gürkaynak F., Häne S., Kaeslin H., Felber N. and FichtnerW., "Towards an AES crypto-chip resistant to differential power analysis", *ESSCIRC*, pp. 307-310, 2004.
[24] Suzuki D., Saeki M. and Ichikawa T., "Random switching logic: acountermeasure against DPA based on transition probability", *IACR ePrint*, rep. 2004/346, 2004.
[25] Popp T. and Mangard S., "Masked dual-rail pre-charge logic: DPA resistance without the routing constraints" *CHES 2005*, pp. 172-186, 2005.
[26] Tiri K., Akmal M. and Verbauwhede I., "A dynamic, differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards", *ESSCIRC*, pp. 403-406, 2002.
[27] Moore S., Anderson R., Mullins R. and Taylor G., "Balanced self checking asynchronous logic for smart card applications", *J. Microprocess. Microsyst.*, Vol. 27, pp. 421-430, 2003.
[28] Tiri K., Hwang D., Hodjat A., Lai B.-C., Yang S., Schaumont P. and Verbauwhede I., "AES-Based cryptographic, biometric security coprocessor IC in 0.18-µm CMOS resistant to side-channel power analysis attacks", *VLSI Sympusium*, pp. 216-219, 2005.
[29] Tiri K. and Verbauwhede I., "Simulation models for side-channel information leaks" *DAC*, pp. 228-233, 2005.

Massoud Masoumi received his BSc from Guilan University, Rasht, Iran in 1996 and MSc and PhD from K.N. Toosi University of Technology, Tehran, Iran, in 1999 and 2006 respectively, all in electronics engineering and all with honor degree. In 2009, he finished a post doctoral research in the context of power analysis attack to symmetric key cipher systems in K. N. Toosi University of Technology. His research interests include side-channel attacks and related countermeasures and efficient VLSI architecture design for digital signal processing systems, error-correcting codes, and cryptosystems.