First-order DPA Vulnerability of Rijndael: Security and Area-delay Optimization Trade-off

Monjur Alam[1], Santosh Ghosh[2], Dipanwita Roy Choudhury[2], and Indranil Sengupta[2]
(Corresponding author: Monjur Alam)

R&D Division, Cadence Design Systems (I) Pvt. Ltd., Noida 201305, India[1]
Department of Computer Science and Engineering, Indian Institute of Technology[2]
Kharagpur, West Bengal 721302, India
(Email:alammonjur@gmail.com, {santosh, drc, isg}@cse.iitkgp.ernet.in
(Received Mar. 28, 2011; revised and accepted Nov. 23, 2011)

Abstract

Differential Power Analysis (DPA) attack for smart card, ASIC or micro controller based on crypto-systems have been demonstrated by several authors. Masking is a very well known approach as a DPA countermeasure. Due to cascading architecture of masked multiplier, the existing masking schemes increase timing and area complexity. Balanced masked architecture brings poor security guaranty. In this paper, we propose a masked multiplier which reduces path delay as compared to the existing ones in the literature. The proposed masked S-box has two level of area optimization. One is avoiding transformation cost and other is using masked bits in sharing mode. We explore security issues in the context of first-order and second-order DPA attacks. We have demonstrated that our approach indeed prevents the first order DPA attack of the Rijndael circuit implemented on FPGA. The proposed masked AES S-box is the most compact one (in terms of area and path delay) as well as secure in the context of first order DPA attack.

Keywords: AES-Rijndael, DPA attack, FPGA, masking

1 Introduction

Power analysis attacks have been demonstrated to be very powerful attacks for most straightforward implementations of symmetric and public key ciphers. Differential Power Analysis (DPA) attack is one of the most powerful side channel attacks, yet it can be mounted using very little resources. This type of attack was first introduced by Paul Kocher *et al.* in [9]. Subsequently various works [14, 21, 26] have been described to exploit side-channel information based on power leakage to break crypto-devices. Experimental results with power analysis attacks on smart cards were reported by [2]. In [26] the first practical power analysis of AES hardware implementation was reported.

The Correlation Power Analysis (CPA) attack of Rijndael[1] implemented on FPGA device was presented by Standaert *et al.* [19]. It was the first successful experiments against an FPGA implementation of Rijndael. It evaluated the effect of pipelining and unrolling techniques in terms of resistance against power analysis.

Several research has been carried out to develop design alternatives to overcome power based side-channel leakages of an AES implementations. The first class of approaches against power analysis attacks tries to remove the root cause for side-channel leakage information. Badman *at el.* [5] proposed a DPA countermeasure by using dynamic voltage and frequency scaling. The authors of the papers [18, 21] came up with a design of masked dual-rail pre-charge logic (MDPL) style which can be implemented using common CMOS standard cell libraries without routing constraints. But it is almost 2 times slower and 3-4 times larger than common CMOS masked implementations of AES S-box [20].

Masking approaches, the second class of countermeasures, counteract DPA by randomizing intermediate results occurring during the execution of the cryptographic algorithm. The idea behind this approach is that the power consumption of operations on randomized data should not be correlated with the actual plain intermediate data. Masking schemes for AES have been presented in [6, 8, 16, 22, 24]. The first one [24] of these schemes has turned out to be susceptible to so-called zero-value attacks [6] and the second one [8] is even susceptible to standard DPA attacks [1]. The third scheme [6] is quite complex to implement and there are no published implementations of this approach so far. The masking schemes proposed in [6], [16] and [22] are provably secure against

[1]The terms AES, Rijndael and AES-Rijndael are used in the same meaning throughout this paper

DPA attacks. This is why these masking schemes are the most reliable for implementing AES in hardware platform. But the drawback is that due to cascading architecture of masked multiplier, the existing masking schemes increase timing complexity. Balanced masked architecture brings poor security guaranty. For *transformation* cost [17] and using masked bits in non sharing mode the existing designs ([16, 22]) lead to more hardware overhead.

In this paper, we demonstrate the power analysis attack of an FPGA implementation of Rijndael and a suitable countermeasure. The elegance of this paper is in two folds. First one is that it is the first analysis of DPA vulnerability of AES-Rijndael which is implemented on an FPGA device using rolling (or looping 10 rounds) architecture. As effectiveness, area requirements of hardware design (FPGA) are of primary importance, and so we have chosen the most compact AES-Rijndael [4] for DPA attack. We demonstrate the approach by which secret key (or part of keys) can be obtained. The second one is that the proposed masked AES S-box is the most optimal one in terms of area and path delay. We find out the source of leakage current from the circuit and take countermeasure for this part only. It can be observed that the S-box is the main source of side-channel information, and thus we have taken necessary countermeasure only for S-boxes instead of full AES circuit. The countermeasure is based on masking technique, where the masked bits (8 bits) are generated by a separate circuit called *random number generator*. We propose a masked multiplier which leads one XOR less path delay comparing the existing ones. The proposed masked S-box has two level of area optimization. One is avoiding transformation cost and other is using masked bits in sharing mode. We have proved that masked bit can be shared without compromising security. We explore security issues in the context of first-order and second-order DPA attacks. We have demonstrated that our approach indeed prevents the first order DPA attack of the Rijndael circuit implemented on FPGA. As a result the proposed scheme is the most compact one (in terms of area and path delay) as well as secure.

The outline of this paper is as follows: Section 2 deals with DPA attack of Rijndael on FPGA device. Section 3 proposes a suitable countermeasure. Section 4 critically analyzes security issues of our masked S-box. Section 5 demonstrates DPA results of our proposed scheme. Finally section 6 concludes the paper.

2 Differential Power Analysis Attacks

The basic idea of the power analysis attack is that the power consumption of a device is statistically correlated to the operations it performs. By monitoring the power usage during cryptographic operations, it is possible to obtain information correlated to the keys. Let us put emphasis on differential power analysis (DPA).

In DPA, measured power traces are compared with a prediction on the power consumption. For DPA, we consider a predictable power consumption model:

$$W_s = W(K_s, PT) \tag{1}$$

where K_s is a small portion of key, PT is a random plaintext, and W is a function of estimated power dissipation. For N given plaintexts, the predicted power W_s can be estimated using Equation (1). Corresponding real power traces P_j can be measured at different time j. Then the correlation coefficient is calculated using the formula:

$$C_s = \frac{E(W_s.P_j) - E(W_s).E(P_j)}{\sqrt{Var(W_s).Var(P_j)}}$$

where W_s and P_j are N-dimension vectors, E denotes the expectation (average) operation, and Var denotes the variance operation. When K_s is not the correct key, the corresponding W_s and P_j have little correlation and the obtained correlation factor is small; when K_s is the correct key, the corresponding W_s and P_j have the highest correlation, and we can find the correct partial key according to values of the correlation factor.

2.1 Power Analysis Attack on FPGA Implementation of AES Rijndael

For power analysis attacks on AES circuit attacker conceives that the power consumption directly depends on data (key and text) on which encryption/decryption takes place. We have collected power traces of AES circuit during its encryption operation. [4] proposed a reconfigurable AES implementation for 128, 192 and 256 bits of key and data. For power analysis attack, a similar type of architecture has been implemented for 128-bit block length only.

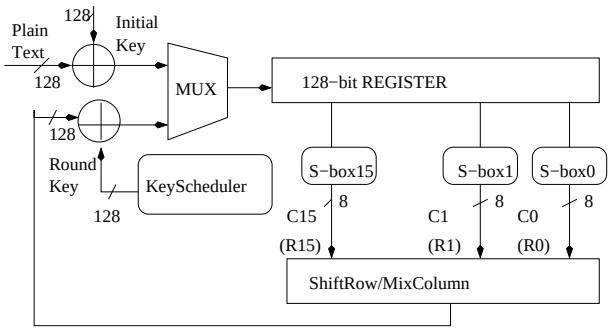

Figure 1: AES architecture for CPA attack

Figure 1 depicts the simplest block diagram of 128-bit AES architecture on which DPA is performed. Here power consumptions are observed at each clock cycle and the consumed power can be calculated as:

$$P = (a \times \sum_{i=0}^{15} H(C_i \oplus R_i) + \sum_{i=0}^{15} b_i$$

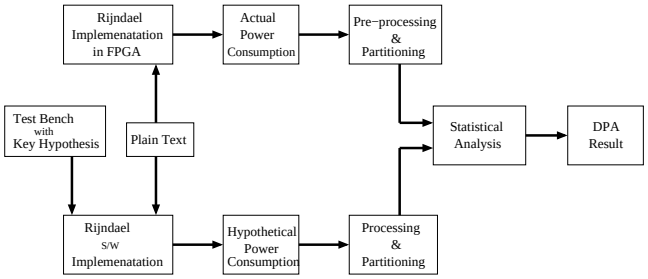

Figure 2: DPA attack flow for our AES circuit

where C_i and R_i are the current and previous 8-bit output values directly related to S-boxes, H(X) denotes the Hamming weight of X, a is a constant and b_i represents the related power for i^{th} S-box. To make our analysis simpler, we neglect the power consumption of KeyScheduler, and noise. ShiftRow does not consume any power as it is a simple rewiring. The power consumed by one MixColumn operation depends on the output values of four 8-bit S-boxes. Therefore, the power consumption of MixColumns and succeeding operations are relevant to the value of C_i and R_i.

2.2 DPA Attack Flow of Rijndael Circuit

Figure 2 depicts DPA flow on FPGA implementation of AES circuit. DPA attack is primarily based on chosen plain text attack with side-channel information. The attacker first chooses plain text inputs to AES circuit assuming that the encryption key is fixed and it is not accessible. The attacker measures actual power consumption P_i of the AES circuit for encryption process as a side-channel information. The information $[P_1, P_2, ...P_N]$ are then stored with respect to the applied plain texts.

Let us predict the power consumption for each possible key for the same plain text, referred to as Hypothetical power consumption. For example, $[H_{11}, H_{12}, ..., H_{1N}]$ for key K_1, $[H_{21}, H_{22}, ..., H_{2N}]$ for key K_2 and so on. The hypothetical (predicted) power consumption does not necessarily represent exact values; it is the relative difference between them [9].

The attacker correlates the hypothetical power consumption ($[H_{i1}, H_{i2}, ..., H_{iN}]$) of each key to that of the actual power consumption ($[P_1, P_2, ...P_N]$). Correlation for the actual key would be higher than the other keys. The detailed theory behind the power analysis attacks has been presented in [9]. As DPA relies on statistical analysis, the quality of analysis increases with number of plain texts.

The consumed power (Captured Data) is then processed to implement a DPA attack. As shown in DPA flow (Figure 2), the processing is performed by following two modules:

- Processing on captured data module which performs some pre-processing on actual and hypothetical power values.
- Statistical analysis module which performs DPA attack discussed in Section 2.

2.3 An Attack Using Practical Measurements

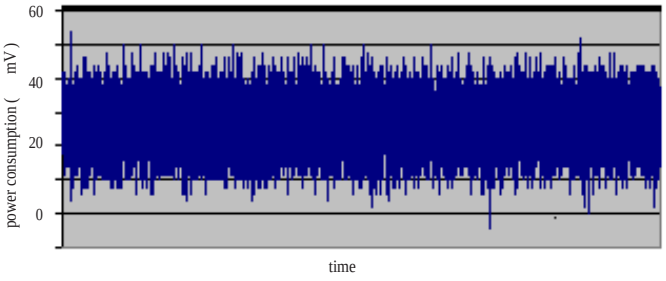

Figure 3: Power consumption of Rijndael circuit for 1000 data

We first perform DPA on our Rijndael test circuit without any countermeasure. Our Rijndael circuit was implemented on Spartan-3 XC3S400 FPGA device. DPA has been done after place and route in the design flow. We have captured the consumed power by a storage oscilloscope. One S-box at a time can be targeted for DPA attack and it may be performed iteratively on 16 S-boxes. In our experiment, the DPA attack first targets the S-box15, i.e., the 8 MSBs (most significant bits). The corresponding hypothetical power consumption is generated. The power analysis attack that we have implemented is similar to the one described in [5]. Detailed setup is mentioned in Appendix.

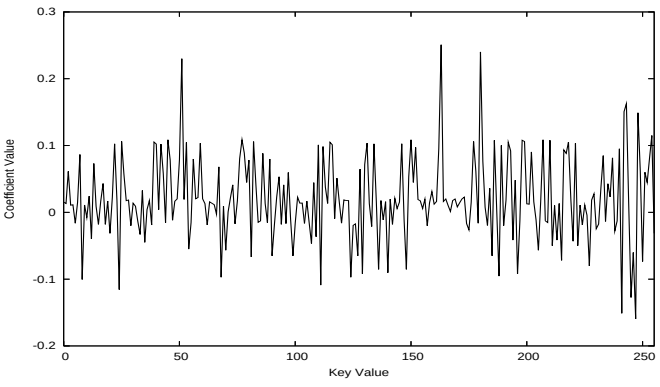

Figure 4: Correlation coefficient values of AES-Rijndael for different keys on 1000 plain texts

Figure 3 shows a direct power trace of 1000 successive encryptions. This power is drawn from the FPGA device for AES encryption. This power trace does not portray any significant meaning regarding secret encryption key. That is why simple power analysis is not effective for

breaking AES, and thus more sophisticated power analysis attacks are applied.

We choose 128-bit plain text with a fixed 128-bit key and collect the consumed power. We take the power 5 times by using the same data and take the average to rectify noise. In order to perform DPA attack, the actual power consumptions for encryption of 1000 random plain texts are collected. These are considered as actual power consumption for encryption process collected by the attacker.

For the hypothetical power value, for each plain text let us consider all possible key combinations. As our target is 8 MSBs, we have total of 256 combinations of keys for each plain text. We have taken consumed power for 1000×256 encryptions.

The correlation power analysis (DPA) is then performed on aforementioned actual and hypothetical power values. The respective DPA result is plotted in Figure 4. The X-axis represents possible 8-bit target key values (0 to 255) and the Y-axis represents the correlation of a particular key's hypothetical power consumption to the actual power consumption. The key with highest correlation represents the correct key. As there are more than one peak (three peaks) present in the curve, we cannot find out the exact key.

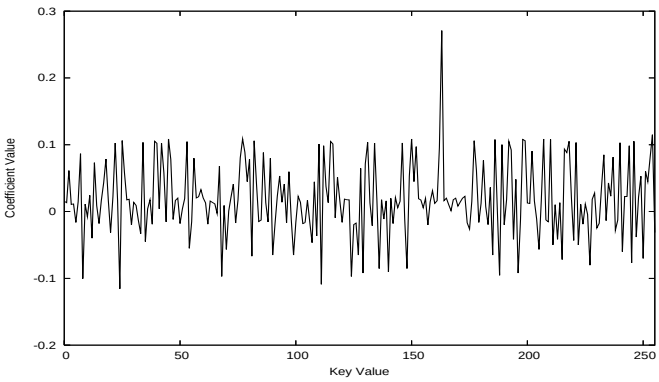

Figure 5: Correlation coefficient values of AES-Rijndael for different keys on 5000 plain texts

[11] mentions that the correlation peak and overall performance of DPA attack is directly depends on the size of the data set. Motivated by this fact, we perform the aforementioned experiments on 50000 plain texts. The correlation for each 8-bit key again calculated on new actual and hypothetical power values. The resulting DPA in this case is plotted in Figure 5. There is only one significant correlation peak in this plot. The peak is found at the position where key value in X-axis is 163. As expected this is the correct 8 key bits $(A3)_h$ (which is equivalent to 163 in decimal).

As the operations of AES algorithm are in terms of bytes, we have targeted 8 bits of 128-bit key at a time. We have seen that for finding 8 key bits the attacker should be able to collect power consumption values for encrypting at least $50000 \simeq 2^{16}$ chosen plain text. The attacker should also be able to perform $256 \times 50000 \simeq 2^{24}$ encryptions on the same AES device for obtaining hypothetical power values. Let us consider the 128-bit AES key as a collection of sixteen bytes, say K_{15}, K_{14}, $\cdots$, K_0, which are relevant to the input of respective S-boxes. The most significant S-box is targeted first and recursively the total key is found out. Thus, aforementioned correlation power analysis need to perform sixteen times for finding the whole 128-bit AES key. In such scenario, the attacking complexity is $16 \times 50000 = 800000 \simeq 2^{20}$ chosen plain text encryption and $16 \times 2^{24} = 2^{28}$ encryption for computing hypothetical power values. This corresponds to $2^{20} + 2^{28} \simeq 2^{29}$ encryption along with some additional statistical analysis required for finding out the whole 128-bit AES key.

3 Masking as a DPA Countermeasure

Within the AES algorithm the plain text is gradually processed using the round keys derived from the cipher key. At any point within the data path where the data (derived from the plain text) and the round key (derived from the cipher key) enter a logic gate, the dynamic power consumption of this gate depends on both the cipher key and the plain text. If this information can be sampled, a power analysis attack can be performed. As such, the output of any function in AES is a candidate point for an attack. The DPA vulnerability of the intermediate results is heavily influenced by the specific implementation of the data path.

The $ShiftRow$ function is a simple bit permutation. The 128-bit parallel data path for this function is realized using rewiring only, and hence does not have any weakness against power analysis attack. Non-linear functions increase the efficiency of differential as well as correlation power analysis attacks. Therefore, the outputs of S-boxes are usually attacked in practice. Attacking the output of $MixColumn$ is very costly as this function is defined for 32 bits. Any attack on this function requires key hypotheses 32 (i.e., 2^{32} keys), which is not impossible but too costly compared to other alternatives. Finally the $AddRoundKey$ function is realized using a bit-by-bit XOR operation of data and key. This is the only function in AES where data and key enter a direct operation together and can be targeted to attack any subset of the cipher key.

It has been observed by [11, 12] that the S-box is the only source of significant leakage signal, and hence responsible for DPA attack of AES circuit. So, as a countermeasure we have implemented a masked AES S-box. This S-box is substituted in Figure 1 keeping other sub parts unchanged. The masking scheme for the inversion operation is based on composite field arithmetic.

3.1 Compact S-box Implementation

Before describing masked S-box implementation, we first describe unmasked S-box implementation. We introduce relevant formulae borrowed from [4, 25] to represent S-box in composite field $GF((2^4)^2)$. The bijection from an element $X(x_0, x_1, \cdots, x_7)$ to a two-term polynomial $X_h x + X_l$ is given by a function δ where $X_h, X_l \in GF((2^4)^2)$, $X \in GF(2^8)$ and δ is defined as

$$\delta(X) = T.X \tag{2}$$

where T is a transformation matrix and defined in [4].

Inversion of two terms polynomial $\Phi(x) = (X_h x + X_l)^{-1}$ can be derived as

$$\begin{aligned}\Phi(x) &= X_h.(X_h^2.\lambda + X_h.X_l + X_l^2)^{-1}x \\ &\quad +(X_h + X_l).(X_h^2.\lambda + X_h.X_l + X_l^2)^{-1}\end{aligned}$$

where λ is a primitive element of $GF(2^4)$. The elements of Φ are calculated as

$$\begin{aligned}\Phi_0 &= x_A + x_0 + x_0.x_2 + x_1.x_2 + x_0.x_1.x_2 \\ \Phi_1 &= x_0.x_1 + x_0.x_2 + x_1.x_2 + x_3 + x_1.x_3 + x_0.x_1.x_3 \\ \Phi_2 &= x_0.x_1 + x_2 + x_0.x_2 + x_3 + x_0.x_1 + x_0.x_2.x_3 \\ \Phi_3 &= x_A + x_0.x_3 + x_1.x_3 + x_2.x_3\end{aligned} \tag{3}$$

where $x_A = x_1 + x_2 + x_3 + x_1.x_2.x_3$. Addition of two terms polynomial is given by

$$(X_h x + X_l) + (Y_h x + Y_l) = (X_h + Y_h)x + (X_l + Y_l).$$

Multiplication of two terms polynomial is given by

$$\Delta(x) = X(x).Y(x) = X(x).Y(x) \bmod m(x)$$

where $\Delta(x), X(x), Y(x) \in GF(2^4)$ and $m(x)$ is irreducible polynomial of degree 4, defined as $m(x) = x^4 + 1$. The elements of Δ can be expressed as

$$\begin{aligned}\Delta_0 &= x_0.y_0 + x_3.y_1 + (x_2 + x_3)y_2 + (x_1 + x_2)y_3 \\ \Delta_1 &= x_1.y_0 + (x_0 + x_3)y_1 + x_2.y_2 + x_1.y_3 \\ \Delta_2 &= x_2.y_0 + x_1.y_1 + (x_0 + x_3)y_2 + (x_2 + x_3)y_3 \\ \Delta_3 &= x_3.y_0 + x_2.y_1 + x_1.y_2 + (x_0 + x_3)y_3\end{aligned} \tag{4}$$

where (+) indicates Galois field addition (bit wise XOR, assuming an extension of the binary field), $(\cdot)$ indicates Galois field multiplication[2]. Squaring in $GF(2^4)$ is a special case of multiplication and computed as

$$\Psi(x) = X(x)^2 \bmod m(x)$$

where $\Psi(x), X(x), m(x) \in GF((2^4)^2)$. The elements of Ψ can be expressed as

$$\Psi_0 = x_0 + x_2, \Psi_1 = x2, \Psi_2 = x_1 + x_3, \Psi_3 = x_3.$$

[2](+) indicates $GF(2)$ addition which is equivalent to two input XOR operation. $(\cdot)$ indicates $GF(2)$ multiplication which is equivalent to two input AND operation. $a \cdot b$, ab and $a \times b$ signify the same. This convention is followed throughout the paper

3.2 Proposed Masking Scheme

When implementing a masked design, designers must pay special attention to the gate structure of the masking scheme. In this scheme, a so-called fresh mask must be added to some intermediate values. This fresh-mask is fundamental for the security of the masking scheme. However, it is logically redundant, e.g. $(A + B) + (C + A) = B + C$. Therefore, during all steps of the design flow, we ensure that this fresh mask is never removed during an optimization step.

The intricate computation structure of the masking schemes causes additional problems for the designers. Typically, designers try to balance the combinational paths. Unbalanced paths have the drawback that the clock period cannot be exploited entirely. Long combinatorial paths also increase the power consumption due to more glitches. The combinatorial delay of the masked implementations, for example, are very long compared to the unmasked implementations of [22]. Therefore, it seems natural to pipeline these structures. However, pipelining in the implementation of [22] is not straightforward. Due to the complicated structure, a considerable amount of pipeline registers are required [7]. So, reducing memory over head is another objective in masking architecture.

From the previous section we have seen that all operations in composite field require only addition (+) and multiplication $(\cdot)$ in $GF(2)$, i.e bit wise XOR and AND. To perform (+) and $(\cdot)$ operations in $GF(2)$, we need fresh masks m_x, m_y, m_r for x, y, r respectively defining the masked variables by

$$x_m = x + m_x, y_m = y + m_y, r_m = r + m_r. \tag{5}$$

As + is a linear operation,

$$x + y = (x_m + m_x) + (y_m + m_y) = (x_m + y_m) + (m_x + m_y)$$

to compute + on two masked data bits, we do not need to unmask these bits

As $\cdot$ is a non-linear operation, our main focus is to mask the $GF(2)$ multiplication

$$xy = r.$$

The masked multiplier constructed by all of the author [22] define as

$$r_m = x_m.y_m + (m_x.y_m + (x_m.m_y + (m_x.m_y + m_r)))$$

the parenthesses indicate the order of operations to avoid any intermediate result with a distribution dependent on the data x and/or y. To reduce the path delay and area overhead we have reconstructed the masked multiplier keeping the same functionality. Our masked multiplier can be represented as

$$r_m = x_m.y_m + (((y_m.m_x + x_m.m_y) + (m_x.m_y + m_r))). \tag{6}$$

Figure 6 depicts our GF(2) mask multiplier. The masked multiplier essentially generates 1-bit output $r_m =$

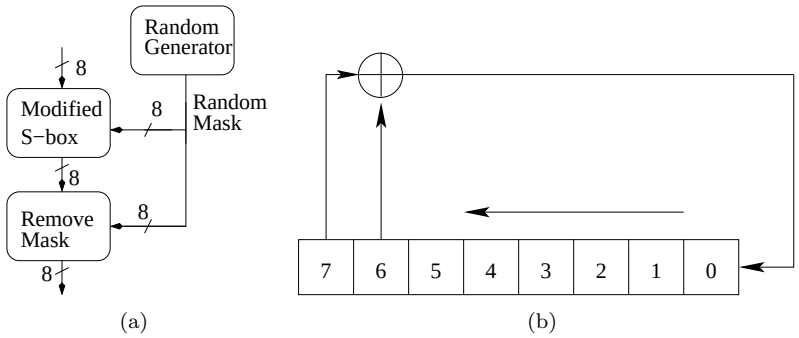

Figure 7: (a) Basic block diagram of our masking scheme (b) LSFR based random number generator

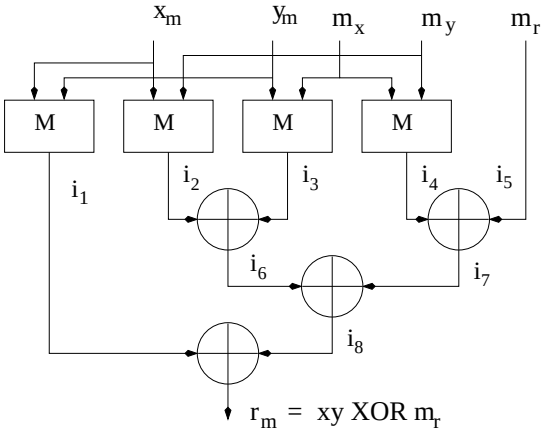

Figure 6: Our proposed architecture of a balanced masked multiplier

$xy + m_r$. The output r_m is the function of intermediate outputs i_1, i_2, i_3, i_4, and i_5, and it is computed as

$$r_m = i_1 + i_2 + i_3 + i_4 + i_5.$$

Comparing to the cascading circuit defined in [12, 22] our balanced circuit is less *glitch* prone and it reduces area over head comparing with synchronous design proposed by Santosh *et al.* [7]. As FPGA implementation does not have *glitching* effect, so analysis of *glitch* [3] over security is out of scope in this article.

3.3 Random Number Generator

To avoid any possible hardware overhead and achieve best efficiency, we stick to a very simple design which ensures required randomness in mask bits. In Figure 7(a), random numbers are used as mask bits. The noise engine, in our case, is based on an 8-bit Linear Feedback Shift Register (LFSR). This LFSR generates pseudo random numbers which are used as input to S-boxes. A LFSR is a finite state machine that operates over finite field F_p, where p is a prime. LFSR generators are made up of two parts, the shift register and the feedback function. The specification for shift register comes down to a bit sequence with a shift usually triggered by a clock drive. To generate a new sequence, the register is shifted 1 bit to the left, and the rightmost bit is computed from the remaining bits in the register. The method used by the feedback function is called tap sequence. Figure 7(b) represents the simplest block diagram of the proposed LSFR which uses $x^7 + x^6 + 1$ as primitive polynomial. At each clock cycle, all memory units except the one driven by the output of feedback function (input tap bit) copy the last bit value from the unit on their right respectively. The input tap bit, as mentioned earlier, gets its value from the external function block which basically consists of units 6 and 7 XOR ed together to produce one bit of feedback. We have proved in Section 4.2 that masked bits generated from the proposed LSFR are independent and uniformly distributed.

The design of [1] requires an additive and a multiplicative mask, [23] uses an additive mask, and [16] uses additive masks. The number of random bits which are required for the masks clearly influences the performance as well. The design of [23] requires 20 random bits per data bit, [1] requires 2 and [16] requires 1 random bit per data bit. In other words, for the encryption of one 128-bit block, [22] needs ten 128-bit random masks.

Instead of using sixteen 8-bit or one 128-bit random number generator, we have used a 8-bit random number generator to implement 128-bit masked AES circuit. As a result there is an 8-bit common input to all S-boxes used (in Figure 1). By this way, we can reduce hardware complexity of our masked AES circuit. In the next section, we have proved that our masking scheme indeed prevent the DPA attack of AES circuit.

4 Security Analysis of Proposed Masking Scheme

We adopt the notion of *perfect masking* given by [6].

Definition 1. *A system is said to be secure if for all adversaries X and all realizable distributions P_1 and P_2, $P_1 = P_2$.*

This definition is equivalent to the *perfect masking* condition given in [6] for standard differential SCA.

Definition 2. *Let a circuit C be comprised of sub circuits $c_1, c_2, \cdots, c_i$. Let $a_1, a_2, \cdots, a_n$ be the input vectors to C. If all the input vectors are statistically independent of the original data, then output of $c_i(\forall i)$ is also statistically independent of the original data.*

Definition 1 does imply that regardless of the hypothesis, the distributions based on this hypothesis are identical. Definition 2 does imply that every operation that is performed by sub circuits c_i in our masking scheme, leads to an output whose distribution does not depend (in a statistical sense) on the input. Our proof is divided into two parts. In the first part, which consists of the two Lemmas by [6] which are re-used.

Lemma 1. *Let $X \in GF(2^n)$ be arbitrary. Let $M \in GF(2^n)$ be uniformly distributed in $GF(2^n)$ and independent of X. Then $f(X) = X + M$ is also uniformly distributed regardless of X. Moreover, the distribution of $f(X)$ is independent of X.*

Lemma 2. *Let $X, Y \in GF(2^n)$ be two arbitrary elements. Let $M_1, M_2 \in GF(2^n)$ be uniformly distributed in $GF(2^n)$ and independent of of each other. Then the probability distribution of $Q = f(X, Y, M_1, M_2) = (X+M_1)\cdot(Y+M_2)$ which is given by*

$$\begin{aligned} Pr[Q] &= Pr[f(X, Y, M_1, M_2) = i] \\ &= \begin{cases} (2^{n+1} - 1)/2^{2n} & \text{if } i = 0, \\ (2^n - 1)/2^{2n} & \text{if } i \neq 0, \end{cases} \end{aligned} \quad (7)$$

is a random product distribution.

In our case the circuit essentially consists of 8 sub circuits. Four multipliers (M) which produce the outputs i_1, i_2, i_3, i_4 and 4 XORs which produce i_6, i_7, i_8, r_m. Input vectors are x_m, y_m, m_x, m_y, m_r. We have shown in Section 4.2 that m_x, m_y, m_r are uniformly distributed. For the convenient let

$$P_1 = x_m.y_m, P_2 = y_m.m_x, P_3 = x_m.m_y, P_4 = m_x.m_y. \quad (8)$$

From Lemma 1

$$x_m = x + m_x, y_m = y + m_y, r_m = r + m_r$$

are uniform distributions. So x_m, y_m and r_m are independent to x, y and r respectively.

From Lemma 2

$$i_1 = P_1, i_2 = P_2, i_3 = P_3, i_4 = P_4 \quad (9)$$

are random product distributions which give a data independent distribution. From Lemma 1

$$i_5 = m_r \quad (10)$$

is uniform distribution.

$$i_6 = y_m.m_x + x_m.m_y = P_2 + P_3 \quad (11)$$

gives uniform distribution of two random product.

$$i_7 = m_x.m_y + m_r = P_4 + m_r \quad (12)$$

which is an uniform distribution of a random product and a random variable. The Equation 12 also can be derived in the form:

$$i_6^{'} = m_x.y + m_y.x \quad (13)$$

for a particular case when $x = y = 0$, the masking does not have any effect for that sub-circuit.

$$i_8 = i_6 + i_7 = x.m_y + y_m.m_x + m_r = x.m_y + P_2 + m_r. \quad (14)$$

The result is of the form $xY + P + M$ where Y and M are independent and uniform, P is *random product*. All are independent of x. If $x = 0$, we get uniform distribution of a *random product* P and R which is independent and uniform. But if $x \neq 0$ we get a uniform distribution (by Lemma 1) which is independent of x.

$$r_m = xy + m_r = r + m_r. \quad (15)$$

In the form $r + M$ where M is independent and uniform. So the distribution $r + M$ is uniform and independent to r (by Lemma 1).

From Equation 9 to Equation 15 our conclusion is that the proposed masked multiplier circuit satisfies the conditions by Definition 2, and hence the conditions by Definition 1. For a very corner case, when $x = y = 0$ only the sub part of the given circuit (i_6) is not perfectly secure. We can consider that this is a trade-off between area optimization and security requirement. It is true fact that the intermediate results can not be added always in secure way.

Lemma 3. *Let $X_i \in GF(2^n)$ be an arbitrary element. Let $M_i \in GF(2^n)$ be uniformly distributed in $GF(2^n)$ and independent of X_i $(\forall i)$. If P_i is a random product and can be expressed as $P_i = f(f(X_i, M_i), M_j)$, then the distribution of $\sum_i P_i$ is not always independent of X_i.*

Proof. Let us define $f(X_i, M_i) = X_i + M_i$ and $f(f(X_i, M_i), M_j) = (X_i+M_i)\cdot M_j$. Linear property of Galois field addition (bit wise XOR) says that if $f(x) = x+m$ then $x = f(x) + m$. Using a counter example we can proof the lemma. Considering the Equation 8, $\sum_{i=i}^{4} P_i = P_1 + P_2 + P_3 + P_4 = x \cdot y$. □

Lemma 3 shows that for secure implementations, the order in which computations of subparts of a circuit is very important. In case of re-structuring of Equation 6

$$r_m = m_r + ((x_m.y_m + x_m.m_y) + (y_m.m_x m_x.m_y +)).$$

Leads to one XOR path delay comparing to [12, 22], but it is no more secure. Oswald *et al.* [16] warns that "every summation of variables must start with the addition of an independent mask M".

4.1 Security Analysis for Hardware Sharing

To get the most optimized masked S-box we have considered for reusing of masking agents. We can re-use m_x or m_y in place of m_r. Let us use m_x in place of m_r. Instead of Equation 5 we will get

$$x_m = x + m_x, y_m = y + m_y, r_m = xy + m_x.$$

The required output of th masked circuit is

$$r_m = xy + m_x.$$

Equation 6 can be written as

$$r_m = x_m.y_m + (((y_m.m_x + x_m.m_y) + (m_x.m_y + m_x))).$$

i_1 and i_6 follow distribution of Equation 9 and Equation 11 respectively.

$$i_7 = m_x.m_y + m_x = m_x \tag{16}$$

which is an uniform distribution and independent to x.

$$i_8 = i_6 + i_7 = y_m.m_x + x_m.m_y + m_x = m_x + x_m.m_y. \tag{17}$$

The result is of the form $P+M$ where M are independent and uniform, P is *random product*. All are independent of x. We get uniform distribution of a *random product* P and M which is independent and uniform.

$$r_m = xy + m_x. \tag{18}$$

In the form $r + M$ where M is independent and uniform. So the distribution $r + M$ is uniform and independent to r (by Lemma 1). From Equation 16 to Equation 18, our conclusion is that the proposed masked multiplier circuit satisfies the conditions by Definition 2, and hence the conditions by Definition 1.

4.2 Uniformness of Masked bits

Now, we define basic terminologies followed by two lemmas to prove that the distribution of the sequences generated by this LSFR is uniform in statistical sense.

Definition 3. *A polynomial $\prod(x)$ in $F[x]$ is called* **irreducible over** *F if it is non-constant and cannot be represented as the product of two or more non-constant polynomials from $F[x]$, each of degree lower than that of $\prod(x)$.*

Definition 4. *a polynomial $\prod(x)$ with coefficients in $GF(p) = \mathbf{Z}/p\mathbf{Z}$ is a* **primitive polynomial** *if it has a root α in $GF(p^m)$ such that $\{1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{p^m-1}\} \in GF(2^n)$ is the entire field $GF(p^m)$, and moreover, $\prod(x)$ is the smallest degree polynomial having α as root.*

Lemma 4. *LFSR produces a maximum-length (pseudo-noise) binary sequence if and only if its characteristic polynomial is a primitive polynomial.*

Proof. By definition, $\prod(x)$ is a primitive polynomial of degree n, if it is irreducible and x is generator for the group G of invertible polynomials *modulo* $\prod(x)$. The group G consists of all polynomials $\prod_i(x)$ of degree j where $j < n$ and $j \neq 0$. $\prod(x)$ is irreducible hence $\prod_i(x)$ *modulo* $\prod(x)$ are a field, $\forall i$. If the LFSR has n slots, the possible internal states are 2^n. Stepping the LFSR is equivalent to multiplying or dividing by x *modulo* $\prod(x)$. x being *generator* means that it goes through the whole group of invertible polynomials regardless of the start position, as long as it is not zero. □

Lemma 5. *Sequence produced by a LFSR which uses primitive characteristic polynomial is uniformly distributed.*

Proof. Let us consider any time frame $\Delta = n \times T + \delta$ where T is the LFSR period, $n \in Z$ (integer set) and $0 \leq \delta \leq T$. Each sequence appears at least n times with a few δ patterns appearing once more. Let the random variable X describes the process of sequences generation from this LFSR. So statistically, based on this interval Δ we can define the cumulative distribution function (CDF) for δ sequence value as

$$P[(X = x] = \frac{n}{n.T + \delta}$$

and for the rest of the values of the field, we can have CDF defined as

$$P[X = x] = \frac{n+1}{n.T + \delta}.$$

Since we have assumed the distribution to be statistical, for a large value n; $n \sim (n + 1)$ and thus these two probability functions converge to the same characteristics. Hence the discrete distribution appears to be statistically uniform. □

Thus, we choose one such polynomial $x^7 + x^6 + 1$ to ensure every possible output except all zeros are in uniform distribution. As the generated masked bits (8-bit) is divided in two parts having 4-bit each, we can assure that the divided masked bits are also uniformly distributed and independent.

Lemma 6. *Given m uniformly distributed over a finite set S, and a one-to-one mapping $g : S- > S$, then $y = g(m)$ is also uniformly distributed.*

Proof. For a finite set, any one-to-one mapping is a bijection; i.e, there is a one-to-one correspondence between those sets. So a uniform distribution is unchanged. In fact, any isomorphism of a finite field is a bijection. □

Lemma 7. *Given $m = [m_1, m_2, \ldots, m_{2n}]$ uniformly distributed over a set S^{2n} of ordered 2n-tuples from a finite set S, then equally divided $y_1 = [m_1, m_2, \ldots, m_n]$ and $y_2 = [m_{n+1}, m_{n+2}, \ldots, m_{2n}]$ of m are independent and uniformly distributed over S^n.*

Proof. m is uniform if and only if m_i is independently uniform over S $\forall i$. Then $y_1 = m_i$ (for $i = 1, 2, \ldots, n$) is independent and uniform; and $y_2 = m_i$ (for $i = n+1, n+2, \ldots, 2n$) is independent and uniform. Our conclusion is for a given mask of n-bit, any sub-mask is also uniform. □

4.3 High-order DPA Analysis of Propose Scheme

A high-order DPA attack defined by Kocher *et al.* [9] as a DPA attack that combines one or more samples with a single power trace.

Definition 5. *An nth-order DPA attack makes use of n different samples in the power consumption that corresponds to n different intermediate values calculated during the execution of an algorithm.*

We have already proved that our scheme indeed secures against first order DPA. Let us concentrate the proposed scheme on high-order, in particular second-order, DPA vulnerabilities. There are several articles [13, 15, 24] on high-order DPA attacks available in literature. Among those the articles of [13, 15] come up with theoretical proofs as well as practical implementations. The second-order DPA attack proposed by Oswald *et al.* [15] will have two steps. Let $\beta_{t_1} = (P + K + M)$ be the intermediate value at time instance t_1, where P is plaintext, K is key and M is mask-bit. Let input of masked S-box is β_{t_1} and the output of masked S-box is $\beta_{t_2} = S^{'}(\beta_{t_1}) = S(P + K) + M$ which is produced at time t_2 $(t_1 > t_2)$. In the first step, an educated guess of time frame (t_1, t_2) is drawn when β_{t_1} and β_{t_2} are computed. At these time stamps actual power traces for processing are obtained from the device. In the second step, predictions are performed with the help of

$$\begin{aligned} C(\beta_{t_2}) - C(\beta_{t_1}) &= C(S(P+K)+M) - C(P+K+M) \\ &= HW(S(P+K)+(P+K)) \end{aligned}$$

where $C(x)$ stands for power consumption for precess x, $HW(x)$ stands for Hamming-weight of value x. For known plaintext attack only K is here unknown. Now one can apply normal DPA by predicting K described in Section 2.

This approach is not applicable for our model. We have already mentioned that masking is applied only for S-box implementation and it works in gate level. As there is no additive making at the *add round key* sub part, the predictions can be obtained as

$$\begin{aligned} &C(S(P+K)+M) - C(P+K) \\ &= HW((S(P+K)+M)+(P+K)) \end{aligned}$$

It is absolutely infeasible for prediction, as M is uniformly distributed and changes its value with time. Moreover question comes, from device perspective for practical attack, how to identify the intersection points (t_1, t_2) in the power trace?

The attack described by Messerges *et al.* [13], targets the + operation of a byte of the key and a byte of masked data. It is assumed that in the implementation under attack, the mask is generated and subsequently added with the data prior to the + operation that involves the key byte. Let $C_i(M)$ stands for consumed power at time $t = 1$ when masked bits are generated, $C_i(P + M)$ stands for consumed power at time $t = 2$ when plaintext P is added with mask M and $C_i(P + M + K)$ stands for consumed power at time $t = 3$ when masked plaintext is added with key.The mean power consumption for every bit of plaintext is calculated as

$$\begin{aligned} \mu_0 &= \Sigma_i |C_i(M) - C_i(P+M+K)| \quad \text{if } P_j = 0 \\ \mu_1 &= \Sigma_i |C_i(M) - C_i(P+K+M)| \quad \text{if } P_j = 1 \end{aligned}$$

where P_j stands for j^{th} bit of plain text. The key bit is calculated like

$$K_j = \begin{cases} 1 & \text{if } (\mu_0 - \mu_1) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

As per our model, generating mask-bit and addition of mask-bit with plaintext (masked S-box operation) compute parallel. So, power trace for $C_i(M)$ and the power trace for $C_i(P + M)$ coincide. It is absolutely difficult to find out the trace for $C_i(M)$ separately.

5 DPA Results of Proposed Masking Scheme

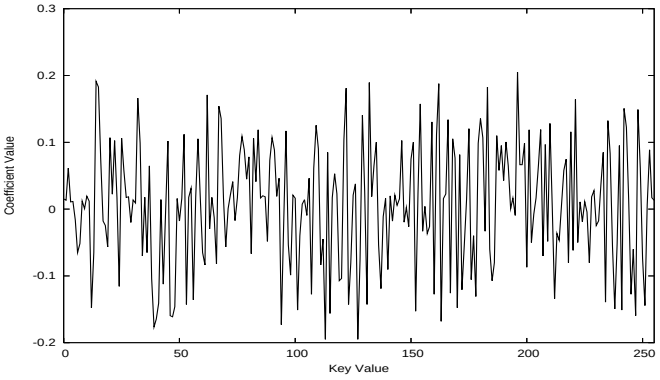

Figure 8: Correlation coefficient values of masked AES-Rijndael for different keys on 50000 plain texts.

If a masking scheme is secure, power consumption of all input transitions is randomized. In other words the expected values for peak and mean power consumption should be equal and independent of input transitions. The CPA result plot for 50000 plain text of our masked AES circuit is shown in Figure 8. As there are no significant peaks present in the curve, we cannot guess the encryption key byte. We have also tested our masked circuit for 100000 data set, and again we could not find the target

key byte. CPA results for 100000 data set is portrayed in Figure 9.

But the benefit in security comes at the cost of hardware cost and number of random bits required. To implement a byte inversion, it is required 5 multiplications and one inversion in $GF(2^4)$. From Equation 3 our unmasked inversion in $GF(2^4)$ requires 12 (+) and 9 (×) operations in $GF(2)$. For masked data we need 4 (+) and 4 (×) operations in $GF(2)$ for each (×) operation in $GF(2)$, which makes in total of 48 (+) and 36 (×) operations in $GF(2)$ to invert one element of $GF(2^4)$. Multiplication in $GF(2^4)$ r requires 12 (+) and 16 (×) operations in $GF(2)$. As there are 3 such multiplications unit in S-box, which results in total of 228 (+) and 192 (×) operations in $GF(2)$.

For timing analysis, the path delay of our unmasked S-box is the resultant path delay of 3 multiplications in $GF(2^4)$, 1 inversion in $GF(2^4)$ and 1 (+) operation in $GF(2)$. From the Equations 3 and 4, $GF(2^4)$ multiplication leads 5 (+) and 4 (×) delay and $GF(2^4)$ inversion leads 7 (+) and 6 (×) delay. We have assumed that 3-input AND delay is equivalent to 2 (×) delay in $GF(2)$. This makes total of 23 $GF(2)$ (+) delay plus 18 $GF(2)$ (×) delay. In case of masked S-box, $GF(2)$ multiplier leads 4 $GF(2)$ (+) delays more. We have reduced 1 $GF(2)$ (+) delay for each masked $GF(2)$ multiplier, resulting 77 $GF(2)$ (+) delay plus 18 $GF(2)$ (×) delay. Moreover we have not used transformation as well as inverse transformation matrix matrix (Equation 2) in S-box, which reduces 10 $GF(2)$ (+) delay. As a result the proposed mask ed AES S-box leads 28 $GF(2)$ (+) delay reduction comparing to conventional masked S-box implementations [22, 23].

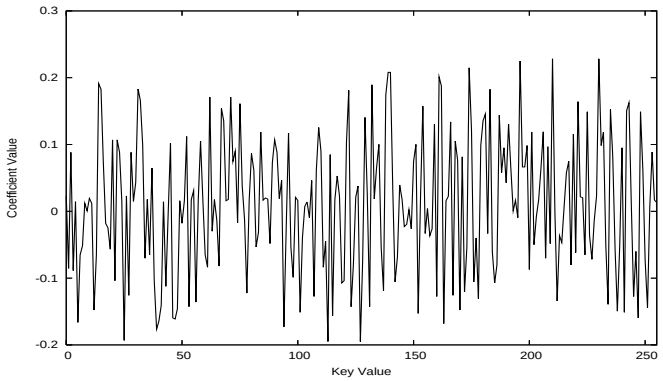

Figure 9: Correlation coefficient values of masked AES-Rijndael for different keys on 100000 plain texts.

Table 1 tabulates the hardware cost, in terms of $GF(2)$ addition (XOR gates) and $GF(2)$ multiplication (AND gates) for $GF(2)$, $GF(2^4)$ multipliers and also the AES S-Box. /, for example 4/6, implies that the component used by the first author is 4 and the component used by the second author is 6.

6 Conclusion

In this paper, we have investigated the security issues of AES-Rijndael implemented by fully rolled fashion on FPGA device. We have performed DPA attacks of this circuit and successfully found the correct 8 bits of AES key. Then we have proposed a suitable countermeasure based on randomness of data-bit which is computed during S-box operation. The advantage of this approach is that it reduces path delay of masked S-box due to restructuring of masked multiplier. Moreover sharing of random bits and avoiding transformation cost, it reduces hardware complexity of the circuit without compromising security.

Acknowledgements

We are grateful to the Department of Information Technology (DIT), Govt. of India for funding us to fulfil this work. We also would like to thank the anonymous reviewers for their critical suggestions that greatly improved the quality of this paper.

A Measurement Setup

A dedicated FPGA board has been developed for mounting the CPA attacks. CPA attack setup of Rijndael for FPGA device is shown in Figure 10. Measurements are perform as follows. First, FPGA is programmed with Rijndael circuit through *Parallel JTAG Cable*. The source voltage of the *FPGA Kit* is 3.5 volts. This voltage branches through FPGA Chip, *RAM* and *I/O*. 1.8 volts is the effective supply voltage to the *FPGA Chip*. The current flowing through *Vcc* Pins of *FPGA Chip* is the required leakage current which is measured by *Current Probe*. The captured values are displayed on the oscilloscope. To have off line analysis by choosing a selection function (as discussed in Section 2), the consumed power traces are stored in a file.

References

[1] M. L. Akkar, R. Bevan, and L. Goubin, "Two power analysis attacks against one mask methods," *FSE, LNCS*, vol. 3017, pp. 332–347, 2004.

[2] M. L. Akkar, R. Bevana, P. Dischamp, and D. Moyart, "Power analysis, what is now possible...," in *Asiacrypt*, vol. 1976, pp. 489–502, 2000.

[3] M. Alam, S. Ghosh, M. Jagonmohan, D. Mukhopadhyay, D. Raychoudhury, and I. Sengupta, "Effect of glitches against masked aes s-box implementation and countermeasure," *IET Information Security*, vol. 3, no. 1, pp. 33–34, 2009.

[4] M. Alam, S. Ray, D. Mukhopadhayay, S. Ghosh, D. Roychowdhury, and I. Sengupta, "An area optimized reconfigurable encryptor for aes-rijndael," *DATE*, pp. 1116–1121, 2007.

Table 1: Comparision of the masked circuits

No of Components	Conventional Masking [10, 23]			Proposed Masking		
	$GF(2)$ Multiplier	$GF(2^4)$ Multiplier	AES S-box	$GF(2)$ Multiplier	$GF(2^4)$ Multiplier	AES S-box
$GF(2)$ $(+)$	4/6	124/222	336/1124	4	76	228
$GF(2)$ $(\times)$	4/4	64/116	312/736	4	64	192
Random bits	3/4	16/32	78/142	2	8	8
Path Delay	4/2 XOR 1/1 AND	21 XOR 4 AND	105 XOR 18 AND	3 XOR 1 AND	17 XOR 4 AND	77 XOR + 18 AND

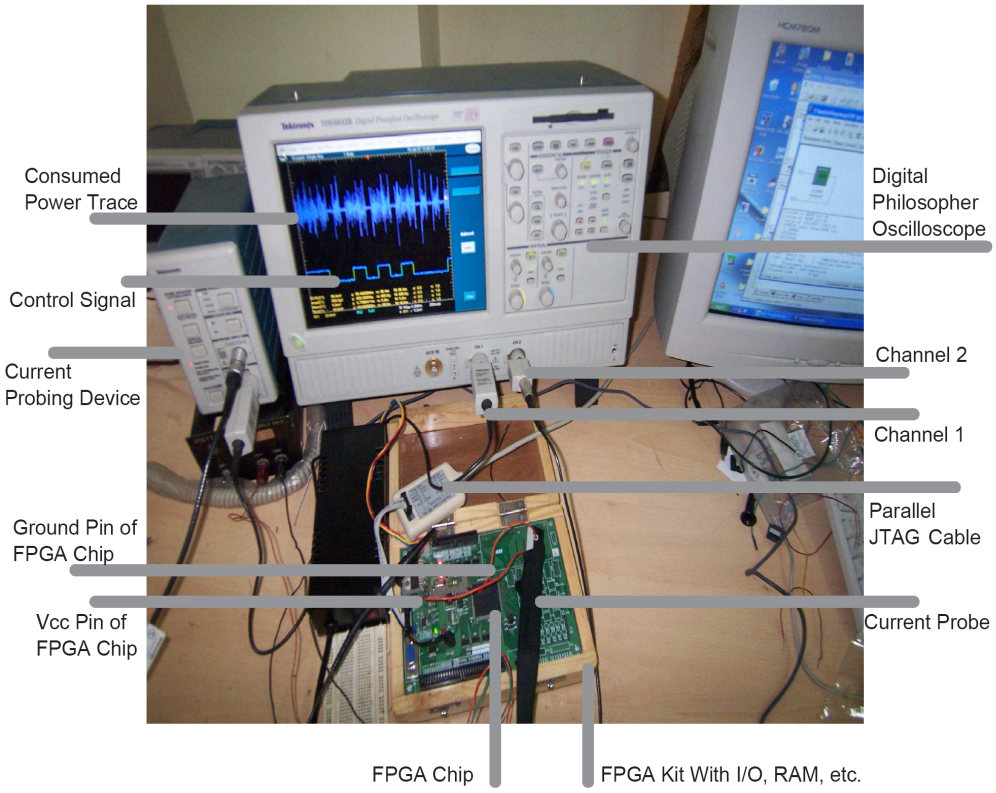

Figure 10: Measurement setup for performing CPA attacks of AES circuit on FPGA device

[5] K. Baddam and M. Zwolinski, "Evaluation of dynamic voltage and frequency scaling as a differential power analysis countermeasure," *VLSID, IEEE*, pp. 854–859, 2007.

[6] J. Blömer, J. Guajardo, and V. Krummel, "Provably secure masking of aes," *SAC, LNCS*, vol. 3357, pp. 69–83, 2004.

[7] S. Ghosh, M. Alam, K. Kumar, D. Mukhopadhyay, and D. Roychowdhury, "Preventing the side channel leakage of masked aes s-box," *ADCOM, IEEE Computer Society*, pp. 15–20, 2007.

[8] J. D. Golić and C. Tymen, "Multiplicative masking and power analysis of aes," in *CHES*, vol. 2535, pp. 198–212, 2002.

[9] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Crypto' 99*, vol. 1666, pp. 388–397, 1999.

[10] K. Kumar, D. Mukhopadhayay, and D. Roychowdhury, "Design of a differential power analysis resistant masked aes s-box," in *Indocrypt*, vol. 4859, pp. 373–383, 2007.

[11] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked aes hardware implementations," in *CHES*, vol. 3659, pp. 157–171, 2005.

[12] S. Mangard and K. Schramm, "Pinpointing the side-channel leakage of masked aes hardware implementations," in *CHES*, vol. 4259, pp. 76–90, 2006.

[13] T. S. Messergs, "Using second-order power analysis to attack dpa resistant software," *CHES, LNCS*, vol. 1965, pp. 238–251, 2000.

[14] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an asic aes implementation," *ITCC*, vol. 2, pp. 546–552, 2004.

[15] E. Oswald, S. Mangard, C. Herbst, and S. Tillich, "Practical second-order dpa attacks for masked smart card implementations of block *ciphers**," *CT-RSA, LNCS*, vol. 3860, pp. 192–207, 2006.
[16] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen, "A side-channel analysis resistant description of the aes s-box," *FSE, LNCS*, vol. 3557, pp. 413–423, 2005.
[17] C. Paar, "Efficient vlsi architectures for bit-parallel computation in galois fields," *PhD thesis, Institute of Experimental Mathematics, University of Essen, Germany*, 1994.
[18] T. Popp and S. Mangard, "Masked dual-rail precharge logic: Dpa-resistance without routing constraints," in *CHES*, vol. 3659, pp. 172–186, 2005.
[19] F. Standaert, S. Ors, and B. Preneel, "Power analysis of an fpga implementation of rijndael: Is pipelining a dpa countermeasure?," in *CHES*, vol. 3156, pp. 30–44, 2004.
[20] K. Tiri, D. Hwang, A. Hodjat, B. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "A side-channel leakage free coprocessor ic in 0.18μ cmos for embedded aes-based cryptographic and biometric processing," *DAC, ACM*, 2005.
[21] K. Tiri and I. Verbauwhede, "Design method for constant power consumption of differential logic circuits," in *Proceedings of Design, Automation and Test in Europe*, pp. 628–633, 2005.
[22] E. Trichina, "Combinational logic design for aes subbyte transformation on masked data," *Cryptology ePrint Archive*, vol. 236, 2003.
[23] E. Trichina, D. D. Seta, and L. Germani, "Simplified adaptive multiplicative masking for aes," in *CHES*, vol. 2535, pp. 187–197, 2002.
[24] J. Waddle and D. Wagner, "Towards efficient second-order power analysis," in *CHES*, vol. 3156, pp. 1–15, 2004.
[25] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An asic implementation of the aes s-boxes," in *CT-RSA*, vol. 2271, pp. 67–78, 2002.
[26] S. M. Yen, "Amplified differential power cryptanalysis on rijndael implementations with exponentially fewer power traces," in *Information Security and Privacy*, vol. 2727, pp. 106–117, 2003.

Monjur Alam is presently working as an R & D engineer at Cadence Design Systems (I) Pvt. Ltd. from March 2008. He obtained B.Tech from the department of Information Technology, Haldia Institute of Technology, Haldia, WB, India in 2005. Subsequently, he obtained his M.S. Degree in 2008 from the department of CSE, IIT Kharagpur. He has published more than 14 technical papers in International Journals and Conferences. His research interests lie broadly in embedded security.

Santosh Ghosh is presently working as post doctoral researcher at ESAT/COSIC, KU Leuven, Belgium. He received his B.Tech degree in Computer Science and Engineering from Haldia Institute of Technology, Haldia, WB, India in 2002 and received his M.S. degree in CSE from IIT Kharagpur, India in 2008. Subsequently, he obtained his PhD Degree in 2011 from IIT Kharagpur. His research interests include embedded security, side-channel analysis, fault analysis, and Secure Testing. He has published 30 technical papers in International Journals and Conferences and has served as Reviewers of several International Conferences and Journals.

Dipanwita Roy Choudhury is a Professor in the Department of CSE, IIT Kharagpur, India. She received her B.Tech and M.Tech degrees in Computer Science from University of Kolkata in 1987 and 1989 respectively, and the PhD degree from the Department of CSE, IIT Kharagpur in 1994. Her current research interests are in the field of Cryptography, Error Correcting Code, Cellular Automata, and VLSI Design and Testing. She has published more than 130 pear reviewed articles. Dr. Roychowdhury has supervised 10 PhD and 7 MS thesis and she is the Principal Investigator of several R&D projects. She is the recipient of INSA Young Scientist Award and Associate of Indian Academy of Science and is the fellow of Indian National Academy of Engineers (INAE).

Indranil Sengupta is a Professor in the Department of CSE, IIT Kharagpur, India. He has obtained his B.Tech, M.Tech and Ph.D. degrees in Computer Science from the University of Calcutta. He joined IIT Kharagpur, in 1988 as a faculty member in the Department of CSE. His research interests include cryptography and network security, VLSI design and testing, and mobile computing. Prof. Sengupta has published more than 120 technical papers in International Journals and Conferences. He has supervised more than 20 MS/PhD students and executed several research and consultancy projects.