DOI: 10.21276/sjet.2016.4.6.5

Scholars Journal of Engineering and Technology (SJET) **ISSN 2321-435X (Online)**
Sch. J. Eng. Tech., 2016; 4(6):284-288 **ISSN 2347-9523 (Print)**

(An International Publisher for Academic and Scientific Resources)
www.saspublisher.com

Research Article

Overcoming the Leakage Power Analysis Attack Using Higher Order DPA-Resistant AES-Masking

Malini S, Manju Priya K, Shiney Immaculate S
SVS College of Engineering, Coimbatore, Tamilnadu, India

***Corresponding author**
Manju Priya K
Email: priyamanju79@gmail.com

Abstract: The analysis of the effectiveness of Leakage Power Analysis (LPA) attacks to cryptographic VLSI circuits on which circuit level countermeasures against Differential Power Analysis (DPA) are adopted. Security metrics used for assessing the DPA-resistance of crypto core implementations is AES encipher and decipher operations and masking is a common method used to prevent differential power analysis (DPA) attack. First, reordering the execution sequence of SubBytes and ShiftRows and partition new critical path of the masked SubBytes followed by the masked MixColumns, and transform computations from GF (2^8) to GF $(2^4)^2$ that efficiently reduces the area. Second, developing an algorithm to search for an optimal transformation matrix of the map function to reduce the critical path of the masked Mix Columns. Third, reusing first order masked SubBytes for higher order masked SubBytes to optimize area without compromising performance. The LPA attacks can be successfully carried out on Higher Order DPA-Resistant AES in presence of process variations.
Keywords: Leakage Power Analysis (LPA), Differential Power Analysis (DPA), SubBytes, ShiftRows.

HISTORICAL REVIEW

In the nanometer regime, the power contribution due to leakage is increasing faster than the dynamic power at each technology node; hence chip power consumption is no longer dominated by the dynamic power. In fact, for a typical 65 nm CMOS chip, leakage power is in the order of half the total power consumption, and it is expected to be an even greater fraction in future technologies. Under these conditions, the leakage power can be easily measured in the same way as the dynamic power is measured in traditional Power Analysis attacks [2]. One Side Channel Attack in particular, namely the Differential Power Analysis (DPA), is of great concern. Side channel attacks can reveal confidential data (i.e. cryptographic keys and user PIN's) exploiting the information leaked by the hardware implementation of cryptographic algorithms. In particular, power analysis attacks, simple and differential, are based on the fact that logic operations feature a power consumption profile dependent on the processed data: with simple statistical analyses of a sufficient number of power traces, the correlation between the circuit switching activity and the key material can be revealed [3].

The logic styles to make devices resistant against SCA attacks are dual-rail pre-charge (DRP) logic styles that consume an equal amount of power and its power consumption is constant or independent of the processed data. In a dual rail precharge (DRP) logic style (e.g., sense amplifier based logic (SABL), wave dynamic differential logic (WDDL), dual spacer (DRP), signals are spatially encoded as two complementary wires and power consumption is constant under the assumption that the differential outputs of each gate drive the same capacitive load.

The delay based Dual Rail Precharge Logic (DDPL) which exploits the time domain data encoding. During the precharge$_V$ phase both differential lines are charged to DD and, $_V$ in the evaluation phase, are both discharged to SS. The information is encoded in the order with which the lines are discharged. For logic'1', the negated line is discharged after a delay with respect to the asserted one. Conversely, for logic'0', the negated line is discharged first. Since over the operating cycles both lines are charged and discharged once, the total current consumption is data-independent.

The proposed architecture to protect "data" in storage area networks from the risk of differential power analysis attacks without degrading performance is a high-throughput masked advanced encryption standard (AES) engine. Masking is a common method used in embedded systems to prevent differential power analysis (DPA) attack. It will mask the plaintext and the

private key at the beginning of the AES computation, and remove redundant masks to recover the cipher text at the end of this computation. In the Boolean masking implementation, the intermediate value x is concealed by exclusive-ORing it with the random mask m. In the round function of the AES, ShiftRows, MixColumns, and AddRound Key are linear transformations, while SubBytes is the only nonlinear transformation of the AES.

LITERATURE REVIEW

A. ANALYSIS OF LPA ATTACKS

Encryption algorithms have been designed to be secure against cryptanalysis that has access to plaintext and cipher text. The physical implementation however, provides the attacker with important information. Numerous attacks have been presented that use 'side channels', such as time delay and power consumption, as an extra source of information to find the secret key.

B. SENSE AMPLIFIER BASED LOGIC

Sense Amplifier Based Logic is a logic style that uses a fixed amount of charge for every transition, including the degenerated events in which a gate does not change state. In every cycle, a SABL gate charges a total capacitance with a constant value. SABL is based on two principles. First, it is a Dynamic and Differential Logic (DDL) and therefore has exactly one switching event per cycle and this independently of the input value and sequence. Second, during a switching event, it discharges and charges the sum of all the internal node capacitances together with one of the balanced output

C. WAVE DYNAMIC DIFFERENTIAL LOGIC (WDDL)

The input signals, which are the outputs of dynamic gates, precharge to '0'. Whenever the inputs of an input AND or OR gate are precharged to '0', the outputs are automatically at '0'. There is no need to force them to 0. Consequently, performing the precharge operation inside the SDDL any input AND gate and the SDDL any input OR gate can be omitted. The dynamic differential cells are now implemented with half the resources required previously. WDDL gates are freely interconnected. Every compound standard cell has only one switching event per cycle. As a result, the differential gates at logic depth '1' switch once per cycle. Differences in input arrival time are not of any influence and do not cause glitch.

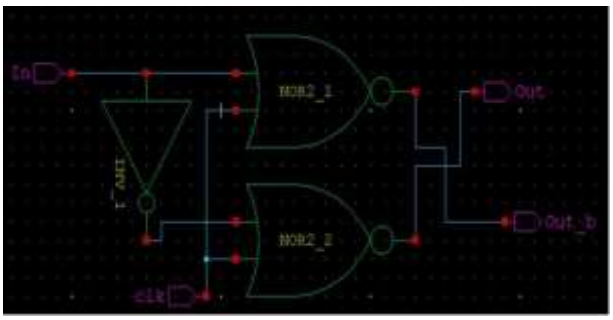

Fig. 1 WDDL INVERTER

D. THREE PHASE DUAL-RAIL PRECHARGE LOGIC

A three phase dual rail pre-charge logic (TDPL) where, during the first phase (pre-charge), the output lines of a generic logic gate are both charged to VDD, then (second phase-evaluation) the proper line is discharged to VSS according to the input data, thus generating a new output data. Finally, during the last phase (discharge), the other line is discharged too [7]. As a consequence, since both wires are pre-charged to VDD and discharged to VSS, a TDPL logic gate shows constant energy consumption over its operating cycle (independent of the input data), even if unbalanced capacitive loads to VDD and/or VSS are taken into account. An inverter is shown in Fig 2, where two additional pull-down NMOS transistors (N1, N4) and a PMOS switch (P1) have been added to the SABL inverter in order to implement the discharge phase.

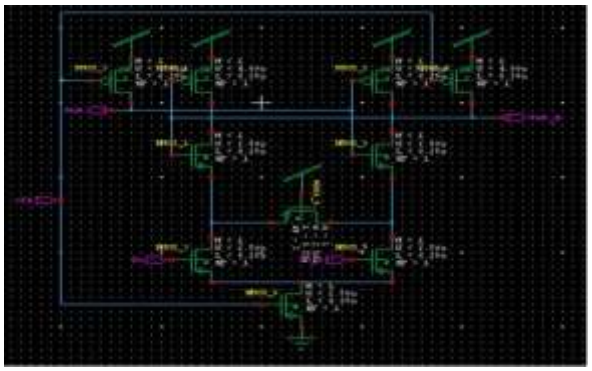

Fig. 2 TDPL INVERTER

E. DELAY BASED DUAL-RAIL PRECHARGE LOGIC

Delay Based Dual Rail Precharge Logic (DDPL) which exploits the time domain data encoding shown in Fig. 3 during the precharge phase both differential lines are charged to V_{DD} and, in the evaluation phase, are both discharged to V_{SS} . The information is encoded in the order with which the lines are discharged. For $_{\Delta}$logic '1', the negated line is discharged after a delay with respect to the asserted one. Conversely, for logic '0', the negated line is discharged first. Since over the operating cycles both lines are charged and discharged once, the total current consumption is data-independent [8].

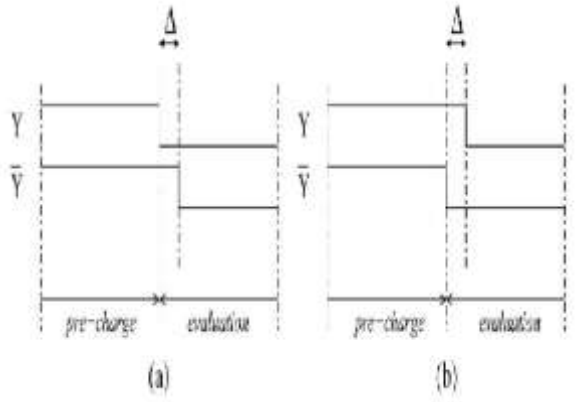

Fig. 3 Time domain data encoding (A) Logic '1'; (B) Logic '0'

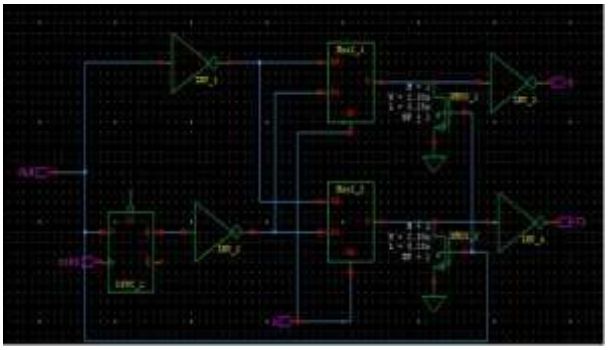

Fig. 4 DDPL INVERTER

F. RIJNDAEL ALGORITHM

With the development of information technology, protecting sensitive information via encryption is becoming more and more important to daily life. In 2001, the National Institute of Standards and Technology (NIST) selected the Rijndael algorithm as the Advanced Encryption Standard (AES), which replaced the Data Encryption Standard (DES) [8]. Since then, AES has been widely used in a variety of applications, such as secure communication systems, high-performance database servers, digital video/ audio recorders, RFID tags, and smart cards [4]. To satisfy different applications requirements, numerous hardware implementations of AES have been reported. Verbauwhede et al described the first AES implementation on silicon, which can provide a 2.29 Gbps throughput with non pipeline architecture. Mukhopadhyay and Roy- Chowdhury improved their AES system to 8 Gbps with pipelining which is a common technique used to enhance the performance of a system. The first AES implementation with a throughput over 10 Gbps was proposed by applying T-box, which is a combination of the SubBytes, ShiftRows, and MixColumns phases in the AES algorithm. Furthermore, the area-throughput tradeoffs of fully pipelined AES processors with throughputs between 30 and 70 Gbps have been presented

III. IMPACT OF PROCESS VARIATIONS ON LPA ATTACKS EFFECTIVENESS

A. LPA ATTACKS AND PROCESS VARIATIONS

In well-defined five steps procedure for LPA attacks has been presented and is summarized in the following. The attack aims to recover the secret key k of a cryptographic device where the processed data X under attack are a function (or a portion) of k. In the first step of LPA attacks, the adversary chooses an internal m-bit signal X that is physically generated within the cryptographic circuit under attack. In the second step, the adversary applies two different input values and measures the corresponding leakage current I of the cryptographic chip at the point of time in which X is physically evaluated. In the third step, the physical value of X within the chip is estimated for each input. For each possible guess k of the secret key, the resulting value of X under the generic input is found. As a result of this step, correct guess can be found.

On the other hand, if the key guess is wrong (i.e., kj is not equal t ok), the measured leakage is no longer linearly related to the estimated H(X), hence the measured leakage and H(X) are loosely correlated and the correlation coefficient ($I_{leak,i}$,Hij) is lower. This means that the correct guess of k leading to the highest value of ($I_{leak,i}$,Hij) among all possible guesses kj. Hence, the adversary must identify the value j*of j that maximizes j and the secret key is simply equal to $k=k_{j,}$ and $\mathbf{P_J=maxP_J}$.

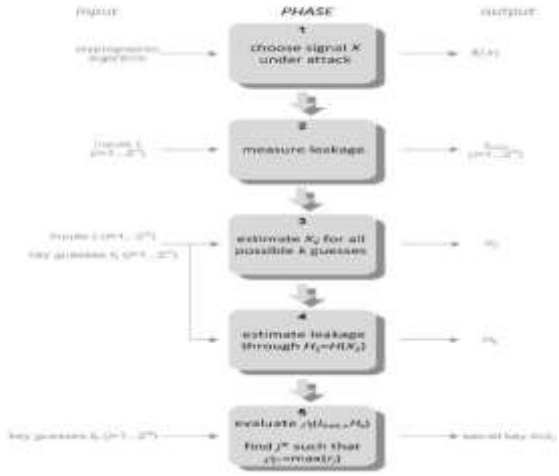

Fig. 5 LPA attack procedure

IV. PROPOSED EFFICIENT IMPLEMENTATIONS FOR DPA-RESISTANT AES

In this section, we propose a novel instruction extension for DPA-resistant AES designs. We detailed the design flow for the first-order masked AES. In order to shorten the critical path, we move the map and the inverse map functions outside AES iterative rounds. An optimization has been done on the masked Sub- Bytes

followed by the MixColumns. This method can also be extended to support second-order and third-order masked AES designs.

A. Proposed First-Order Masked AES

In order to resist against DPA attacks, we need to remove the correlation between intermediate results and the secret key. Unlike the extension for the AES design, the masked AES is more complicated due to extra area required to perform masking functions. In the masked implementation, the intermediate value X is concealed by exclusive-oring (XOR) it with the random mask. In the round function of the AES design, ShiftRows,MixColumns and AddRound Key are linear transformations, while SubBytes is the only nonlinear transformation. We define the linear transformation as Linear_Op, therefore, masking Linear_Op with m is shown as follows:

Linear_Op(x xor m) =Linear_Op(x) xor Oper(m)

The nonlinear operation SubBytes is defined as S-box, which should have the following character:

S-box(x xor m) $\neq$ S-box(x) xorS-box (m)

In order to mask the nonlinear transformation, a new S-box, denoted as S-box$'$, is recomputed as follows:

S-box$'$(x xor m) =S-box(x) xor m$'$

Where m and m$'$ are the input and the output masks of the Sub- Bytes. It holds

m$'$=S-box(m)

Generally, to mask a 128-bit AES, it needs 6 byte random values. This masking method is called first-order masking. It can be broken by higher-order DPA attacks. It has been pointed out that every AES S-box with 1<j<16 should use a different set of d-1 input masks and output masks for each round to thwart DPA attacks of orders < d. This requires having enough mask sets to properly conceal sensitive information. These refreshing processes of mask sets are very costly when implementing them on software platform. In order to have a trade-off between area and performance, the required mask sets are generated and maintained within the hardware domain, which avoids information leakage from software and only the masked intermediate results are sent to software for computations. In the proposed design, we use 16 different masks for 16 S-boxes for one round. M_j and M'_j represent the input and the output masks for the SubBytes transformation at the j^{th}round j ε (0, 1, 2…10), respectively. Mx_j and Mx_j represent the input and the output masks for the MixColumns transformation, respectively. These masks are refreshed at each round. However, masking the SubBytes over GF (2^8) is area consuming. We will transform this operation from GF (2^8) to GF $(2^4)^2$ to optimize area.

The field GF (2^8) is an extension of the field GF $(2^4)^2$, over which to perform modular reduction needs an irreducible polynomial of degree 2 and another irreducible polynomial of degree 4. In order to reduce the hardware resources, we calculate the masked AES engine mainly over GF $(2^4)^2$. Fig. 6 shows the architecture of the proposed first order masked 128-bit AES, which moves the map and the inverse map functions outside the AES iterative round. The plaintext and masks are mapped once from GF (2^8) to GF $(2^4)^2$ at the beginning of the computation

All intermediate operations are computed over GF $(2^4)^2$. Finally, the cipher text is mapped back from GF (2^8) to GF $(2^4)^2$. All masks need to be mapped from GF (2^8) to GF $(2^4)^2$, and we denote that m_j=map (M_j and m_{xj}=map(Mx_j), where represents the AES round. Similar to the above instruction extension example for the AES, we partition into hardware the critical part of first-order masked AES, the masked SubBytes followed by the masked MixColumns. Therefore, the ShiftRows transformation is moved before the SubBytes transformation and it is kept at the software level. In this particular design, two new instructions are required:

- Computing the masked SubBytes followed by the masked MixColumns; Computing the masked SubBytes only (the last round of AES computation does not need the MixColumns transformation).

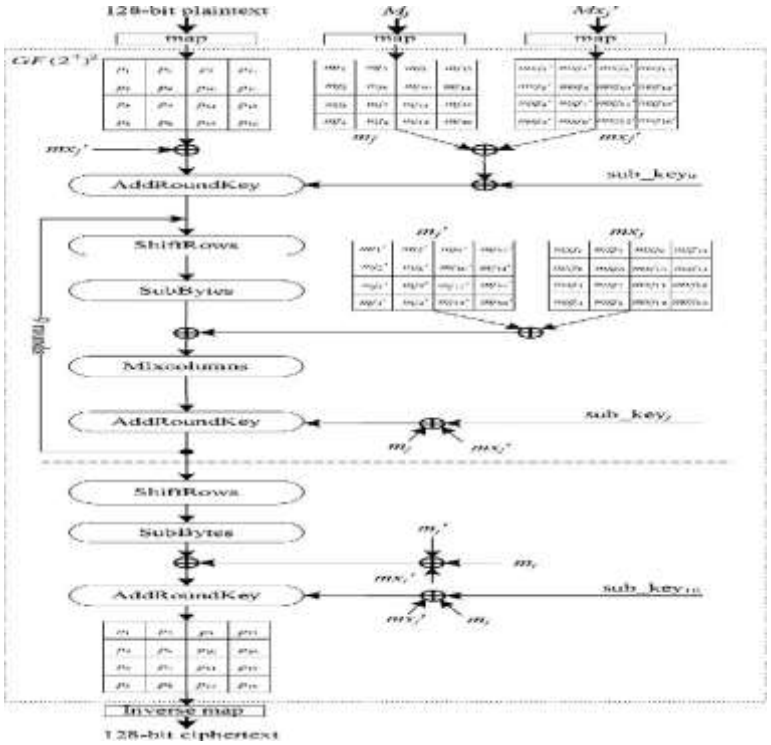

Fig. 6 Structure of the proposed first-order masked AES design

Fig.7 shows the architecture of the proposed instruction extension for the first-order masked AES. It could also be extended to support the second-order AES and the third-order AES designs. Note here, the masked SubBytes and the masked MixColumns are computed over GF $(2^4)^2$. The proposed architecture consists of four parts: the masked SubBytes, the masked MixColumns, random number generator (RNG) and masks. RNG is designed to generate the corresponding mask sets for the masked AES. In order not to affect the original data path, an optimization has been done on the masked SubBytes followed by the masked MixColumns, which will be detailed in the following section.

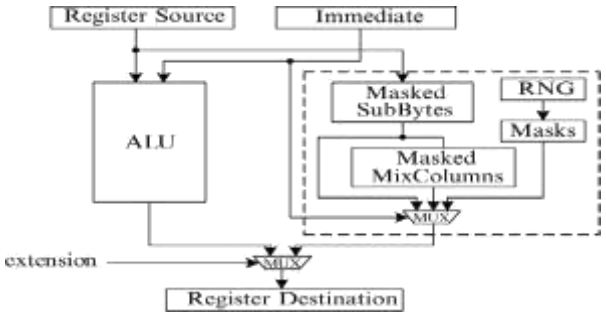

Fig. 7 Structure of the proposed extension in the

AES-M1 design

Fig. 8 shows the execution flow of the proposed first-order masked AES. In order to be compatible with the original C program and properly generate mask sets, three extra instructions are developed. They are: 1) generating mask sets for the proposed design; 2) masking plaintext with mx'_j; 3) generating the sub_key's mask (mx_j+mx'_j).

The procedure of the proposed first-order masked AES design is shown as follows, where SW represents software and HW represents hardware.

1. SW requests HW to generate the mask sets and maps these sets from GF (2^8) to GF $(2^4)^2$.
2. SW requests HW to generate the sub_keys and maps them from GF (2^8) to GF $(2^4)^2$.
3. SW maps plaintext from GF (2^8) to GF $(2^4)^2$ and sends it to HW.
4. HW masks plaintext and sends back to SW.
5. SW requests the sub_keys mask sets from HW.
6. SW masks sub_keys with the received mask sets and performs AddRound Key.
7. SW sends the intermediate result to HW after ShiftRows.
8. HW performs the masked SubBytes and the masked MixColumns and sends the results back to SW.
9. SW performs AddRound Key.
10. At the last round, HW only performs the masked SubBytes followed by correction of the results.
11. After AddRound Key, SW maps the result at Step 11back from GF $(2^4)^2$ to GF (2^8) to retrieve the correct cipher text.