A Randomized Multi-modulo RNS Architecture for Double-and-Add in ECC to prevent Power Analysis Side Channel Attacks

Jude Angelo Ambrose[†] Hector Pettenghi[‡] Darshana Jayasinghe[†] Leonel Sousa[‡]
[†]School of Computer Science and Engineering,
University of New South Wales, Sydney, Australia
{ajangelo,darshanaj}@cse.unsw.edu.au
[‡]Instituto de Engenharia de Sistemas e Computadores,
Instituto Superior Tecnico,
Lisboa, Portugal 1000-029
{hector,las}@sips.inescid.pt

Abstract

Security in embedded systems is of critical importance since most of our secure transactions are currently made via credit cards or mobile phones. Power analysis based side channel attacks have been proved as the most successful attacks on embedded systems to retrieve secret keys, allowing impersonation and theft. State-of-the-art solutions for such attacks in Elliptic Curve Cryptography (ECC), mostly in software, hinder performance and repeatedly attacked using improved techniques. To protect the ECC from both simple power analysis and differential power analysis, as a hardware solution, we propose to take advantage of the inherent parallelization capability in Multi-modulo Residue Number Systems (RNS) architectures to obfuscate the secure information. Random selection of moduli is proposed to randomly choose the moduli sets for each key bit operation. This solution allows us to prevent power analysis, while still providing all the benefits of RNS. In this paper, we show that Differential Power Analysis, cross correlation analysis and Correlation Power Analysis are thwarted using our solution.

I. Introduction

Embedded systems have become the realm of current economy, playing an integral part in our day to day life. Devices such as mobile phones and smart cards are heavily used for secure transactions. Such devices are constantly attacked by adversaries to attain secure information, e.g., secret keys which are

stored within. Side Channel Attacks (SCAs) are one of the popular attacks in embedded systems which exploit the external manifestations, such as power , timing and electro-magnetic emissions to predict the secret key [1]. These external properties are recorded while a processor in the embedded device executes cryptographic programs for encryption/decryption with the secret key. A rigorous offline analysis is then taken place to realize any correlations between the external manifestations and the actual computation, by performing predictions of the secret key. Power analysis has been the most extensively used Side Channel Attack technique to extract secret keys during the execution of cryptographic algorithms [1], [2]. Simple Power Analysis (SPA), a simple kind of power analysis, predicts the Hamming weights (i.e., number of 1's set in the output) of the executed data by directly correlating the power magnitude. This is based on the hypothesis that the higher the Hamming weight, the higher the power magnitude [2]. Differential Power Analysis (DPA), a more advanced attack, utilizes statistical analysis by correlating the predictions with the actual power measurements. A significantly high correlation, if exists for a specific prediction, is considered as the correct prediction. The hypothesis behind DPA is the considerable difference in power dissipation when processing '0's and '1's [1]. Detailed explanations on DPA can be found in [1], [2].

The Elliptic Key Cryptography (ECC) is a well known public key cryptographic algorithm, which has taken over the embedded systems market from its contender RSA [3]. ECC uses smaller key size with faster computation to suit small devices. Power analysis attacks have been the most successful attacks on ECC [4]. As described in [4] unprotected point scalar multiplication of ECC (details explained in Section III-A) has been successful for SPA and DPA attacks, especially the point Double-and-Add function. The most investigated countermeasure for Double-and-Add to prevent power analysis is random masking [5], [6], where random computations are performed together with the actual computation (mainly in software or algorithm) to obfuscate the secret key. Other countermeasures include constant path execution [5], protected new algorithms [7], software balancing [8] and hardware implementations [9]. These solutions either involve significant algorithmic and software modifications or incur high cost, sacrificing performance. Hence, in order to eliminate software modifications and not sacrifice performance, while still providing security against power analysis, we propose a randomized Multi-modulo residue number system architecture for the point Double-and-Add in ECC. Residue Number System (RNS) is widely used in signal processing and recently adopted for ECC [10]. RNS allows decomposition of operations which can be executed in parallel, carry-free execution, reducing complexity and power for large arithmetic units. However, RNS circuits require an extra cost to convert the binary number to a residue number and then translate again to binary after processing. Multi-modulo architectures (MMA) are recently adopted for RNS processors [11] (even though the core idea is proposed in 1994 [12]),

to improve performance and power, allowing operations with different moduli in series and in parallel. The conventional operation of ECC would be sequential Doubling and then the Addition, as shown in Figure 1(a). As presented in Figure 1(b) our Double-and-Add RNS (DARNS) proposal converts the input data (using the *DIRECT MMA* converter) into residue numbers, which are then processed in parallel for *DOUBLING MMA* and *ADDITION MMA*. Finally the processed residue numbers are converted from residue to binary, using the *REVERSE MMA* component. We propose to randomly change the moduli of *DOUBLING MMA* and *ADDITION MMA* to protect from both SPA and DPA attacks. As far as we aware, this is the first time the MMA is introduced in ECC to prevent power analysis. Our approach is much simpler and cost effective, providing the same degree of security, compared to prior countermeasures.

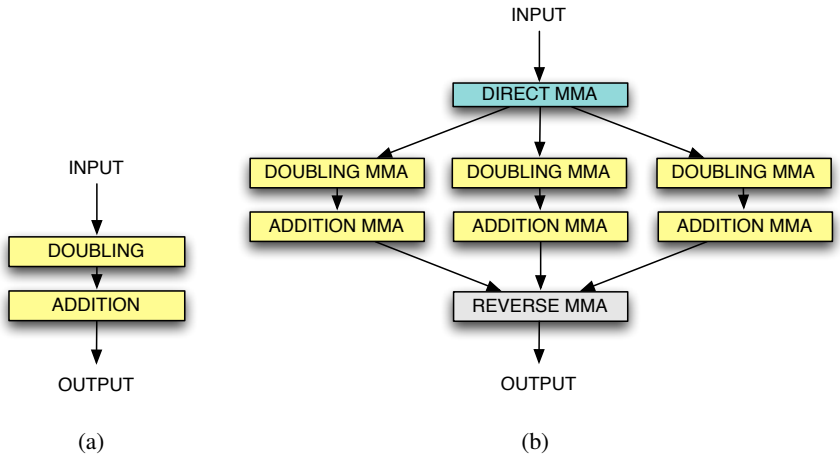


Fig. 1. (a) Standard Vs. (b) MMA RNS for Double-and-Add

The rest of the paper is organized as follows; Section II discusses the related work and Section III provides background on Double-and-Add. The proposed DARNS architecture is detailed in Section IV, whereas the integration of the DARNS architecture into ECC is demonstrated in Section V. Experimental setup is briefed in Section VI. DPA attack and analysis are provided in Section VII. The paper is concluded in Section VIII.

II. Related Work

We categorize the related work on power analysis countermeasures for the Double-and-Add in ECC into: 1), random masking, 2), constant path execution, 3), new algorithms, 4), software balancing and; 5), hardware solutions. Detailed list on general countermeasures for power analysis can be found in [13].

One of the first researchers to propose several masking countermeasures for the Double-and-Add is Coron [5], who introduced random computations with the actual point multiplication to randomize the

power profile. The author's first technique was to randomize the private component (i.e., secret key) and undo the randomization at the receiver end. Even though the solutions are considered sound, the author in [5] acknowledges that the solutions were neither implemented nor proven secure. Furthermore, Okeya and Sakurai [6] found vulnerabilities in Coron's approaches [5] and demonstrated successful attacks. Hence the authors in [6] proposed an improved scalar multiplication, using projective coordinates. Izumi et al. [14] proved that the randomizing addressing countermeasure proposed in [15] was still vulnerable.

Coron [5] proposed a constant path execution approach to prevent the SPA on Double-and-Add. The control flow within the algorithm was eliminated by rewriting the software using array elements. A careful study of the algorithm and complete rewriting of the code is necessary to perform this task. Similar limitation exists for the new point multiplication algorithm proposed in [7] to prevent power analysis. Sakai and Sakurai [8] proposed a software balancing approach where the software code was modified in such a way that complementary events are coded to negate the effects of the actual computations.

Koschuch et al. [9] designed a hardware/software based ECC implementation which eliminated the dependency in Hamming weights of the key to the Double-and-Add operations. As the authors indicate, this provides protection against SPA, however security against DPA was not demonstrated. Residue Number Systems (RNS) have recently gained interest to prevent side channel attacks. Bajard et al. [16] optimizes the Montgomery algorithm using the RNS multiplication, claiming that their design is secure against side channel attacks. However, no attacks were performed to justify this claim. Similarly, the work in [17] proposed an RNS enhanced microprocessor for ECC, indicating that randomness was introduced to thwart power analysis. The authors used the masking technique from [5], which was already proved vulnerable by [6].

Compared to the random masking approaches [5], [18], [6], [14], our DARNS architecture does not hinder performance (compared to a standard multi-modulo RNS) and not require any software or algorithmic modifications, while providing better security than some of the techniques (i.e., protecting against both the SPA and DPA). The algorithm does not have to be modified in contrast to the constant path execution [5] or the new algorithmic solutions or the software balancing approaches [8]. For the first time, we are proposing a MMA RNS based architecture, DARNS, to prevent power analysis based side channel attacks for the Double-and-Add in ECC. Moduli sets are randomly selected to create random behavior in the power profile. Our solution achieves benefits of using the RNS, while still preventing SPA and DPA.

This journal is an extension of the work presented in [19] and the improvements made are as follows;

- An improved random pattern generation to alleviate security is presented

- The DARNS architecure is extended to support the new secure patterns
- Correlation Power Analysis is attempted on DARNS to evalute its security

III. Preliminaries

An embedded device, D_k, is produced with a hardware key, $k \in K$. The hardware keys, K, will be unique per device (example include mobile phones and gaming devices) which are utilized to perform encryption and decryption for secure transactions. An adversary who knows k can replicate the device and act as the owner, causing severe financial damage and other security related problems. Side channel attacks (SCA) exploit the external manifestations, $M^m_{D_k}$, via a medium, m, from D_k. The adversary measures this leaked manifestations via probes or monitoring devices (e.g., power profiles are measured using voltage measurements across ground and Vcc in smart cards and recorded in a Storage Cathode-ray Oscilloscope [20]). An offline analysis with the help of Mutual Information is performed on the measured information to predict k. The adversary predicts a key k' and computes the $\mathrm{I}_{k'k}$(Y;X), by feeding the input variable **X** and observing the output variable **Y**. $\mathrm{I}_{k'k}$(Y;X) is the Mutual Information when the correct key is k and the predicted key is k'. The $\mathrm{I}_{k'k}$(Y;X) will be significantly higher when k' matches k and much smaller (almost negligible and zero) when they are different. More information on predicting the correct key using this approach can be found in [21]. We briefly explain the formulation of our DPA attack. The adversary measures the power trace, $P_k[j]$, of the DARNS architecture by providing the input $I[j]$, with a known key K_k. Power value observed when processing with $K_k[i]$ key bit is $P_k[j][i]$. Similarly, N_I number of inputs are fed and respective power traces are recorded. During the attack stage, the adversary does not know the actual secret key K_u. Same input sets I (used before) are fed again and power traces P_u are obtained. The power value at $K_u[i]$ key bit for input $I[j]$ is $P_u[j][i]$. The DPA value per key bit position i is then formulated as shown in Equation 1. As shown, the $DPA_T[i]$ is computed by subtracting the means of the power values when measuring with the known key, K_k, and the power values when measuring with the unknown key, K_u. The final DPA_T will contain points for each bit of the secret key (both K_u and K_k are of the same size and both the power profiles P_k and P_u are sampled at fixed points). If the known key bit $K_k[i]$ does not match with the unknown key bit $K_u[i]$, a significant peak is expected at $DPA_T[i]$. A value close to zero should be observed when the bits match. Based on this hypothesis, the differential trace DPA_T is analyzed for significant peaks which will indicate different bit values being used in K_u and K_k, if not, same values. For example, if the known key is b'1111' and the unknown key is b'1010', the differential trace will indicate significant peaks at positions 2 and 4 (bit positions starting from 1 and from most significant end) and almost zero for positions 1 and 3.

$$DPA_T[i] = \frac{1}{N_I}\sum_{j=1}^{N_I} P_k[j][i] - \frac{1}{N_I}\sum_{j=1}^{N_I} P_u[j][i] \tag{1}$$

A. ECC and Double-and-Add

Elliptic Key Cryptography (ECC) is a public key cryptographic algorithm where senders will use a private key to encrypt the data and receivers will use the public key for decryption. A sender chooses a private key k in the interval between 1 to $n-1$ where n is a prime number, of length l bits. A public key P_k is created, such that $P_k = k.G$, where G is a known point in the elliptic curve:

$$E : y^2 = x^3 + ax + b \tag{2}$$

The operation of computing the public key ($P_k = k.G$) involves an addition of point G to itself k times. This operation is called scalar multiplication, where the binary version of Double-and-Add approach is used, as shown in Algorithm 1. The algorithm takes in the point G and the private key k as inputs. The output will be the public key P_k. For each bit of k, where the length of k is l, the Q is doubled at the start of each iteration. When the bit value d_i is one, G is added after doubling. Itoh et al. [22] showed that this unprotected Double-and-Add in ECC can be attacked using SPA and DPA respectively.

Algorithm 1: Double-and-Add from [5]

```
Input: G, k
Output: P_k = k.G
Q ← G
for i from l-2 to 0 do
    /* Double portion */
    Q ← 2.Q
    if d_i == 1 then
        /* Add portion */
        Q ← Q + G
P_k ← Q
return Q
```

IV. DARNS ARCHITECTURE

The proposed DARNS architecture has three major components: 1), *DIRECT*, to convert the binary to residue, 2), *ARITHMETIC*, to perform parallel doubling/addition MMAs; and 3), *REVERSE*, to convert residue back to binary. The underlying idea is to randomly choose the moduli sets for each iteration of the DARNS, which still makes the circuit properly function, while changing the behavior of the logic. *control_R* signal is randomly set (i.e., '0') or cleared (i.e., '1') to choose the moduli set at runtime. However, to minimize the power consumption with the change of moduli, a pattern of possibilities for the *control_R* is used. This pattern, which maintains the random characteristic in DARNS, will be detailed in Section 5. In order to reduce the power consumption, *ENABLE* signals are included to disable the direct and reverse converters when there is no change in *control_R* (i.e., '0'). *control_K* is the secret key bit per iteration as explained in Algorithm 1, used to decide whether to perform the doubling/addition operations. Figure 2 shows the DARNS for a couple of moduli sets $\{2^{2n}, 2^n \pm f_1\}$ and $\{2^{2n}, 2^n \pm f_2\}$, with $f_1 = 1$ and $f_2 = 3$. It is important to emphasize that this technique can be extended for any odd value of f_i, and it is not restricted to a two moduli set. The following describes the steps for designing the three stages of the DARNS.

A. Direct VMAs

A standard direct Single Modulo $A = \{2^n \pm f\}$ Architecture (direct SMA), f odd, transforms an integer G with m-bit inputs $(\{g_0, g_1..., g_{m-1}\})$, into a residue word R of a-bit outputs $(\{r_0, r_1, ..., r_{a-1}\})$, with $a = \lceil log_2(A) \rceil$, $i.e.$ $a = n$ and $a = n + 1$ for modulo $A = \{2^n - f\}$ and $A = \{2^n + f\}$, respectively. The input value G is converted from $G = \sum_{j=0}^{m-1} 2^j \cdot g_j$ into:

$$|G|_A = \sum_{j=0}^{a-1} 2^j \cdot r_j = \left| \sum_{j=0}^{m-1} 2^j \cdot g_j \right|_A . \tag{3}$$

A technique to avoid the division operation to compute the remainder was presented in [23]:

$$|G|_A = \left| \sum_{j=0}^{m-1} \left|2^j\right|_A \cdot g_j \right|_A . \tag{4}$$

If the weight values $\left(\left|2^j\right|_A\right)$, associated to the inputs are directly available, $|X|_A$ can be computed simply by adding the weight terms $\left|2^j\right|_A$ for which $g_j = 1$. The importance of the Eq. (4) is that the weights of the m-inputs, g_j, are reduced from m to a. The weight selection of the inputs (g_j), is of key importance in designing RNS architectures in order to improve performance. In case of direct Variable

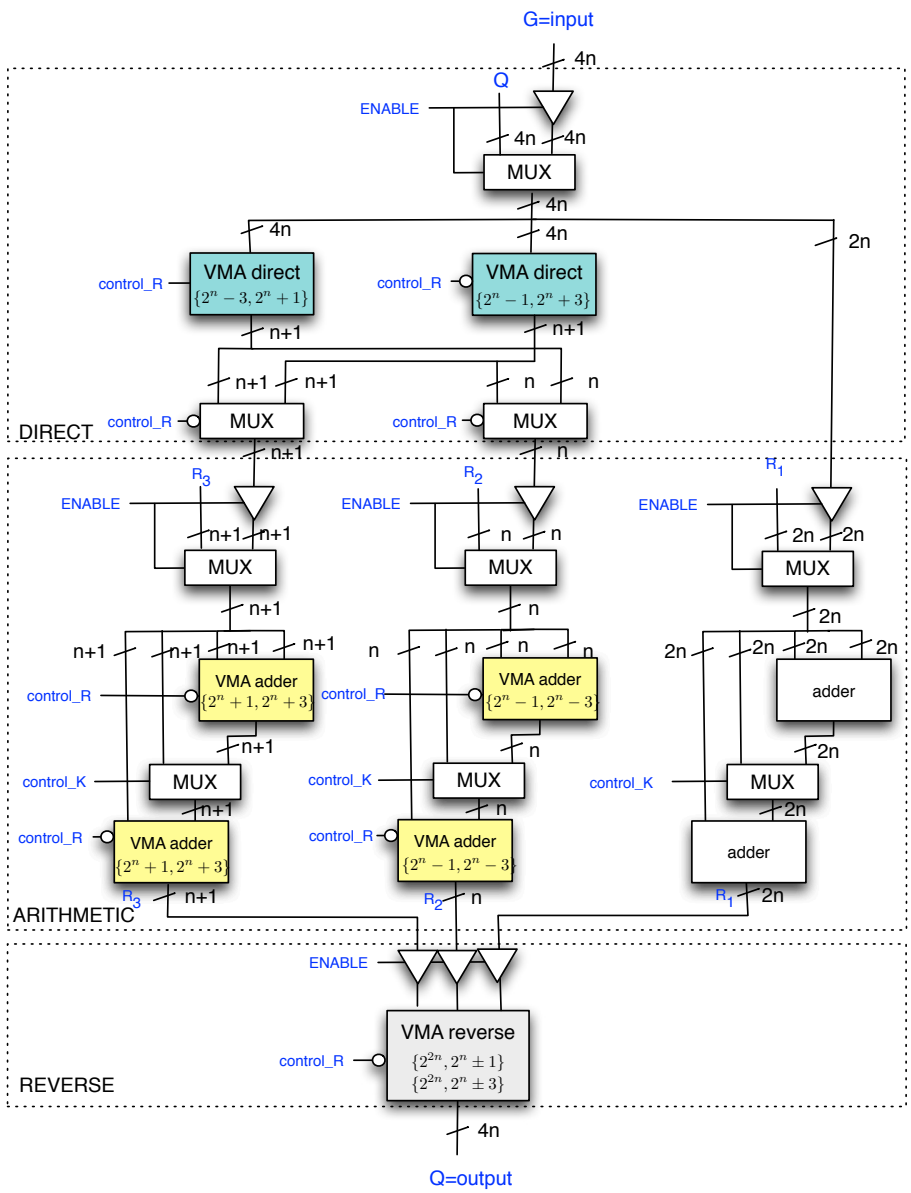


Fig. 2. Proposed DARNS Architecture

Multi-moduli Architectures (direct VMAs), several moduli shares the common hardware and they can be selected in series by a control [12]. Let us denote this q-set of moduli as $\{A_1, A_2, ..., A_q\}$. Thus, Eq. (4) is needed to be applied q times in order to obtain $\{|G|_{A_1}, |G|_{A_2}, ..., |G|_{A_q}\}$.

The herein proposed direct VMA structure using a couple of modulus A_1 and A_2 is shown in Figure 3(a), which is applied in the proposal DARNS architecture presented in the next section. It essentially consists of two stages, where the addition of the precomputed inputs and the final modulo

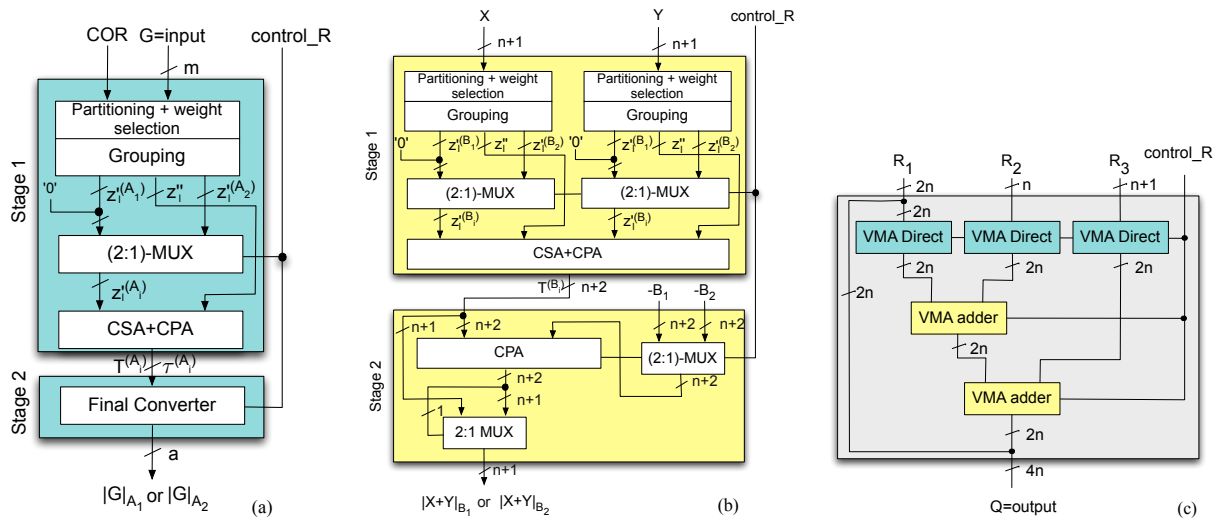


Fig. 3. Proposed (a) Direct, (b) Adder and (c) Reverse VMA architectures included in the proposed DARNS.

addition are carried out in the first and second one, respectively. A control, herein denoted as *control_R* is also included to select the required modulo computation, modulo A_1 or A_2.

Stage 1: The precomputation of the inputs is carried out to obtain the non-common and common bits, $z_l^{'(A_i)}$ and $z_l^{''}$, respectively as well as the required correction factor COR. This precomputation is performed in two levels: *i)* partitioning and weight selection, and *ii)* grouping, as is detailed explained in [11]. The summation of the residues $\left|2^j\right|_{A_i} g_j$, for each modulus A_i, is defined as:

$$T^{(A_i)} = \sum_{j=0}^{m} \left| \left|2^j\right|_{A_i} \right| g_j. \tag{5}$$

Thus, the summation given in Eq. (5) is maximum when all inputs are one. In this particular case

$$T_{max}^{(A_i)} = \sum_{j=0}^{m} \left| \left|2^j\right|_{A_i} \right|, \tag{6}$$

which has a size of

$$\tau^{(A_i)} = \lceil log_2(T_{max} + 1) \rceil. \tag{7}$$

Therefore, a reduction from m to $\tau^{(A_i)}$ bits is achieved after the CSA + CPA stage [11].

Stage 2: The calculation of $|G|_{A_i} = \left|T^{d(A_i)}\right|_{A_i}$ in Eq. (4) is carried out by means of a memory-less Final Converter (FC) designed using subtracters and a MUX. The main difference of the VMA proposal in comparison with [11] is that our proposal avoids the use of recursive (CSA+CPA) stages that minimizes the number of bits at the input of the FC. This reduction of the delay in our proposal is obtained at cost

of an area increase of the FC. Our proposed FC consists of a conditional corrective addition, by means of comparisons, which improves the idea presented in [24]. The total number of required comparisons associated to the modulo value A_i, $\varepsilon^{(i)}$, depends on the maximum value at the FC input, $T_{max}^{(A_i)}$, which is $\varepsilon^{(i)} = \lceil \frac{T_{max}^{(A_i)}}{A_i} \rceil$.

B. Adder VMAs

The proposed VMA adder for a couple of modulus $B_1 = \{2^n + f_1\}$ and $B_2 = \{2^n + f_2\}$, for f_1 and f_2 $odd \in [1, 2^n - 1]$, is shown in Figure 3(b). The design of the VMA adder for a q-set of moduli as $\{B_1, B_2, ..., B_q\}$ is described as follows. In the case of the addition $X + Y$, with $X = \{x_0, x_1, ..., x_n\}$ and $Y = \{y_0, y_1, ..., y_n\}$ is:

$$X + Y = T^{(B_i)} = \sum_{j=0}^{n} \left|2^j\right|_{B_i} \cdot x_j + \sum_{j=0}^{n} \left|2^j\right|_{B_i} \cdot y_j \tag{8}$$

The modulo computation of the addition can be derived by:

$$|X + Y|_{B_i} = \begin{cases} T^{(B_i)} - (B_i), & \text{if } T^{(B_i)} \geq B_i \\ T^{(B_i)}, & \text{otherwise,} \end{cases} \tag{9}$$

The condition of one single comparison to derive the modulo computation in Eq. (9) is always set because the maximum value of $T^{(B_i)}$, herein denoted as $T_{max}^{(B_i)}$ is:

$$T_{max}^{(B_i)} < 2 \times (B_i) \tag{10}$$

for a positive weight selection of the inputs. Therefore, the use of a correction term is not needed for this proposal. The derivation of the terms of $T^{(B_i)}$ in Eq. (8) can be explained in the same way than $T^{(A_i)}$ in Eq. (5). Therefore, the stage 1 of direct VMA is applied twice in the VMA adder to derive the residue terms associated to X and Y as is shown in Figure 3(b). The last stage consist on a subtraction of B_i, Eq. (9), selected by a (q:1)-MUX. The $|X + Y|_{2^n+k}$ computation is carried out by means one CPA for which its processed $S_{[n+1]}$ controls the following (2:1)-MUX that select the correct arithmetic operation.

C. Reverse VMAs

Once that the doubling and addition operations are derived in the arithmetic unit by means VMA adders, is needed to carry out the conversion from RNS to the final binary solution, Q. In this subsection

is presented the design of the reverse SMA conversion from the moduli set $\{2^{2n}, 2^n \pm f\}$ to binary, where the modulus are $m_1 = 2^{2n}$, $m_2 = 2^n - f$ and $m_3 = 2^n + f$. The corresponding reverse VMA is derived from the reverse SMA converter herein proposed. For the defined set are defined, $\left(M = \prod_{i=1}^{3} m_i\right)$, $\hat{m}_i = M/m_i$, and $\left|\hat{m}_i^{-1}\right|_{m_1}$ that represents the multiplicative inverse of $\hat{m}_i$ with respect to modulus m_i. These definitions are required to decode the RNS representation into binary, Q, from the residues R_i, $1 \leq i \leq 3$, of the arithmetic unit output by using the CRT [25]:

$$Q = \left|\sum_{i=1}^{3} \hat{m}_i \left|\hat{m}_i^{-1}\right|_{m_i} R_i\right|_M . \tag{11}$$

Eq. (11) can be rewritten as:

$$Q = \sum_{i=1}^{3} \hat{m}_i \left|\hat{m}_i^{-1}\right|_{m_i} R_i - MA(Q), \tag{12}$$

where $A(Q)$ is an integer that depends on the value of Q.

The values of the multiplicative inverses satisfy the condition $\left|(\hat{m}_i)(\hat{m}_i^{-1})\right|_{m_i} = 1$ for $(i = 1, 2, 3)$. This proof, due to the lack of space, is not herein presented.

To get the value of Q we can also use the equation:

$$Q = \left\lfloor \frac{Q}{m_1} \right\rfloor m_1 + R_1, \tag{13}$$

which is obtained by concatenating $\left\lfloor \frac{Q}{m_1} \right\rfloor$ to R_1.

To compute $\left\lfloor \frac{Q}{m_1} \right\rfloor$ we can divide (12) by m_1 and take the floor value of both sides modulo $\hat{m}_1$:

$$\begin{aligned}
\left\lfloor \frac{Q}{m_1} \right\rfloor &= \left|\left\lfloor \frac{Q}{m_1} \right\rfloor\right|_{\hat{m}_1} = \left|\left|\left\lfloor \sum_{i=1}^{N} \left|\hat{m}_i^{-1}\right|_{m_i} \frac{\hat{m}_i}{m_1} R_i \right\rfloor\right|_{\hat{m}_1} - \overbrace{\left|\frac{M}{m_1} A(Q)\right|_{\hat{m}_1}}^{=0}\right|_{\hat{m}_1} = \left|\left|\left\lfloor \left|\hat{m}_1^{-1}\right|_{m_1} \frac{\hat{m}_1}{m_1} R_1 \right\rfloor\right|_{\hat{m}_1} + \right. \\
&\left. + \left|\overbrace{\left|\hat{m}_2^{-1}\right|_{m_2} \frac{\hat{m}_2}{m_1}}^{V_2} R_2\right|_{\hat{m}_1} + \left|\overbrace{\left|\hat{m}_3^{-1}\right|_{m_3} \frac{\hat{m}_3}{m_1}}^{V_3} R_3\right|_{\hat{m}_1}\right|_{\hat{m}_1} = \left|\left|\left\lfloor \left|\hat{m}_1^{-1}\right|_{m_1} \frac{\hat{m}_1}{m_1} R_1 \right\rfloor\right|_{\hat{m}_1} + v_2 + v_3\right|_{\hat{m}_1} ; \\
\left\lfloor \frac{Q}{m_1} \right\rfloor &= \left|\sum_{i=1}^{3} v_i\right|_{\hat{m}_1} ,
\end{aligned} \tag{14}$$

since the terms in this equation for $2 \leq i \leq 3$ are integers. Let us to denote from now on the integer terms associated to the floor operation as v_i. In order to derive the v_1 it is important to notice that the multiplicative inverse $\left|\hat{m}_1^{-1}\right|_{m_1}$ needs to satisfy:

$$\left|\left|\hat{m}_1^{-1}\right|_{m_1} \hat{m}_1\right|_{m_1} = \left|\left|\hat{m}_1^{-1}\right|_{m_1} (2^{2n} - f^2)\right|_{m_1} = \left|\left|\hat{m}_1^{-1}\right|_{m_1} (\overbrace{\left|2^{2n}\right|_{m_1}}^{=0} - f^2)\right|_{m_1} = \left|-\left|\hat{m}_1^{-1}\right|_{m_1} f^2\right|_{m_1} = 1. \tag{15}$$

Due to $\left|-(\gamma 2^{2n}-1)\right|_{m_1} = \left|-(-1)\right|_{m_1} = 1$, being γ a positive integer, the multiplicative inverse $\left|\hat{m}_1^{-1}\right|_{m_1}$ can be expressed as:

$$\left|-\left|\hat{m}_1^{-1}\right|_{m_1} f^2\right|_{m_1} = \left|-(\gamma 2^{2n}-1)\right|_{m_1} = 1;$$
$$\left|\hat{m}_1^{-1}\right|_{m_1} = \frac{\gamma 2^{2n}-1}{f^2}. \tag{16}$$

The term associated to $i=1$ can be rewritten as:

$$\left|\left\lfloor \left|\hat{m}_1^{-1}\right|_{m_1} \frac{(2^{2n}-f^2)}{m_1} R_1 \right\rfloor\right|_{\hat{m}_1} = \left|\left|\frac{\cancel{2^{2n}}}{\cancel{m_1}} \left|\hat{m}_1^{-1}\right|_{m_1} R_1\right|_{\hat{m}_1} - \left\lfloor \frac{(\gamma 2^{2n}-1)}{m_1} R_1 \right\rfloor\right|_{\hat{m}_1} =$$
$$= \left|\left|\overbrace{(\left|\hat{m}_1^{-1}\right|_{m_1} - \gamma)}^{V_1} R_1\right|_{\hat{m}_1} + \overbrace{\left\lfloor \frac{R_1}{m_1} \right\rfloor}^{=0}\right|_{\hat{m}_1} = \left|V_1 R_1\right|_{m_1} = v_1. \tag{17}$$

The term $\left\lfloor \frac{R_1}{m_1} \right\rfloor = 0$, due to $\frac{R_1}{m_1}$ is a non integer term lower than one. Therefore, the v_i terms are processed by direct SMA converters for each channel. The corresponding v_i are added by a tree of SMA adders to derive the floor value $\left\lfloor \frac{Q}{m_1} \right\rfloor$ of Eq. (14). Finally the $\left\lfloor \frac{Q}{m_1} \right\rfloor$ is concatenating to R_1 to derive the output Q in Eq. (13).

The reverse VMA can be easily derived by substituting the SMA direct and adders by the corresponding VMA ones previously presented. The addition of a control, *control_R*, to select the moduli set is required. We present the VMA for the two moduli sets $\{2^{2n}, 2^n \pm f_1\}$ and $\{2^{2n}, 2^n \pm f_2\}$ in Figure 3(c). These moduli sets are the ones required by the proposal DARNS in the case study presented in the experimental results section. It important to emphasize that the reverse VMA is able to compute the modulo of the v_i not for two moduli sets, but for three or more.

V. DARNS IN ECC

The integration of our DARNS architecture into the ECC is articulated in Figure 4. Set of inputs I_1, I_2,..., I_m are fed to perform encryption, which are then executed to perform lines 4-7 of Algorithm 1, using multiple DARNS logics (it can be also a single DARNS architecture with a feedback based logic). Based on whether *control_K* or $d_k[i]$ is '0' or '1', the DARNS architecture performs its operations. We propose a pattern of sequences, herein denoted as s, containing possible candidates for $d_r[0,1,...,l-1]$, which have only one change of state in the bit position h, $0 \leq h \leq l-1$. Figure 4 indicates that the number of such candidates per pattern is $2l-2$, resulting in $s[0,...,2l-2]$. A random pattern of only one transition is chosen to minimize power consumption. This transition has proven enough to mask the

power profile and it is indeed possible to allow more transitions in d_r to increase security at an extra cost in power.

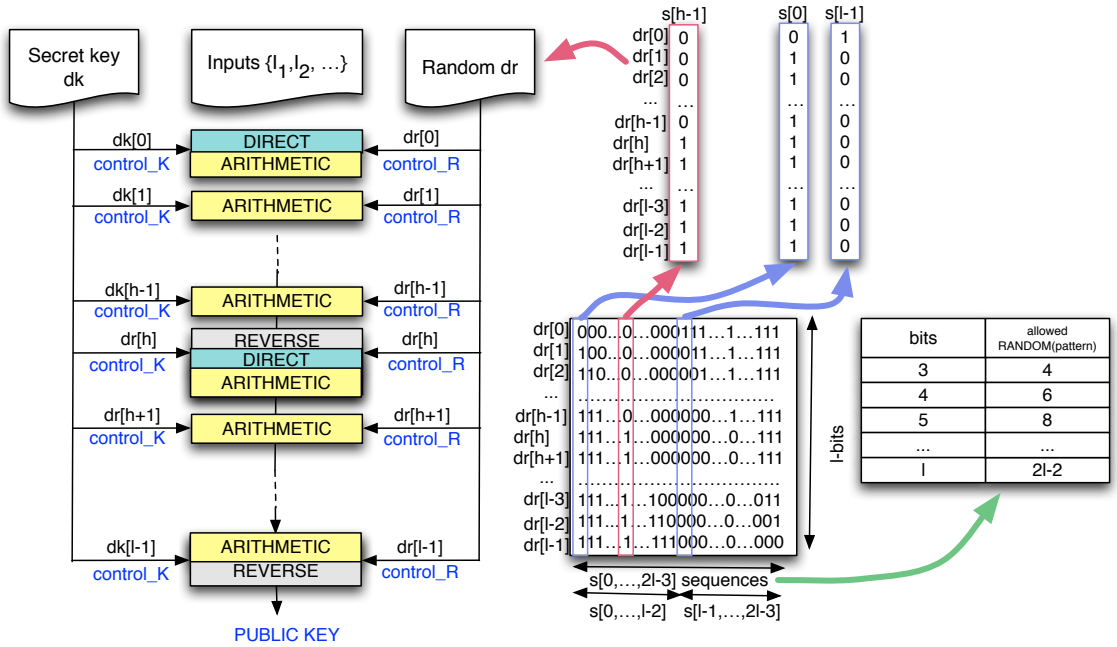


Fig. 4. The DARNS Architecture into ECC

It is worth noting that these patterns can be easily implemented by the modified Johnson (or Moebius) counter presented in Figure 5(a). However, this random number generator (RNG) is deterministic, hence might be vulnerable to attacks. Our RNG structure to generate the required pattern using RNS processors is presented in Figure 5(b). The functionality of the proposed RNG is presented as follows: *i)* The RNG provides a random sequence of numbers, R, larger than $(2l-2)$, of size $r \geq \lceil log_2(2l-2) \rceil$ bits; *ii)* The random value, R, is processing by a direct RNS, such as $|R|_{2l-2}$, in order to obtain $(2l-2)$ possible numbers into the range $[0, 2l-2]$. The values of R could be deterministic as the RNG output is also masked at the output of the direct RNS. Therefore, deterministic RNGs can be chosen, such as the modified Johnson counter presented in Figure 5(a), since they are less complex and faster than the non-deterministic ones; *iii)* The $u = \lceil log_2(2l-2) \rceil$ bits associated to $|R|_{2l-2}$ contains $(2l-2)$ possible values that are retrieved from a ROM, which contains the $s[0, ..., 2l-2]$ allowed sequences. Due to the fact that the half of the sequences are the complement of the other half, $s[0, ..., l-1] = \bar{s}[l-2, 2l-2]$, it is possible to reduce the number of words required in the ROM to half. In this way $(l-1)$ sequences of d_r are selected for storage in ROM. The rest of $2^u - (l-1)$ words are set to zeros since they are

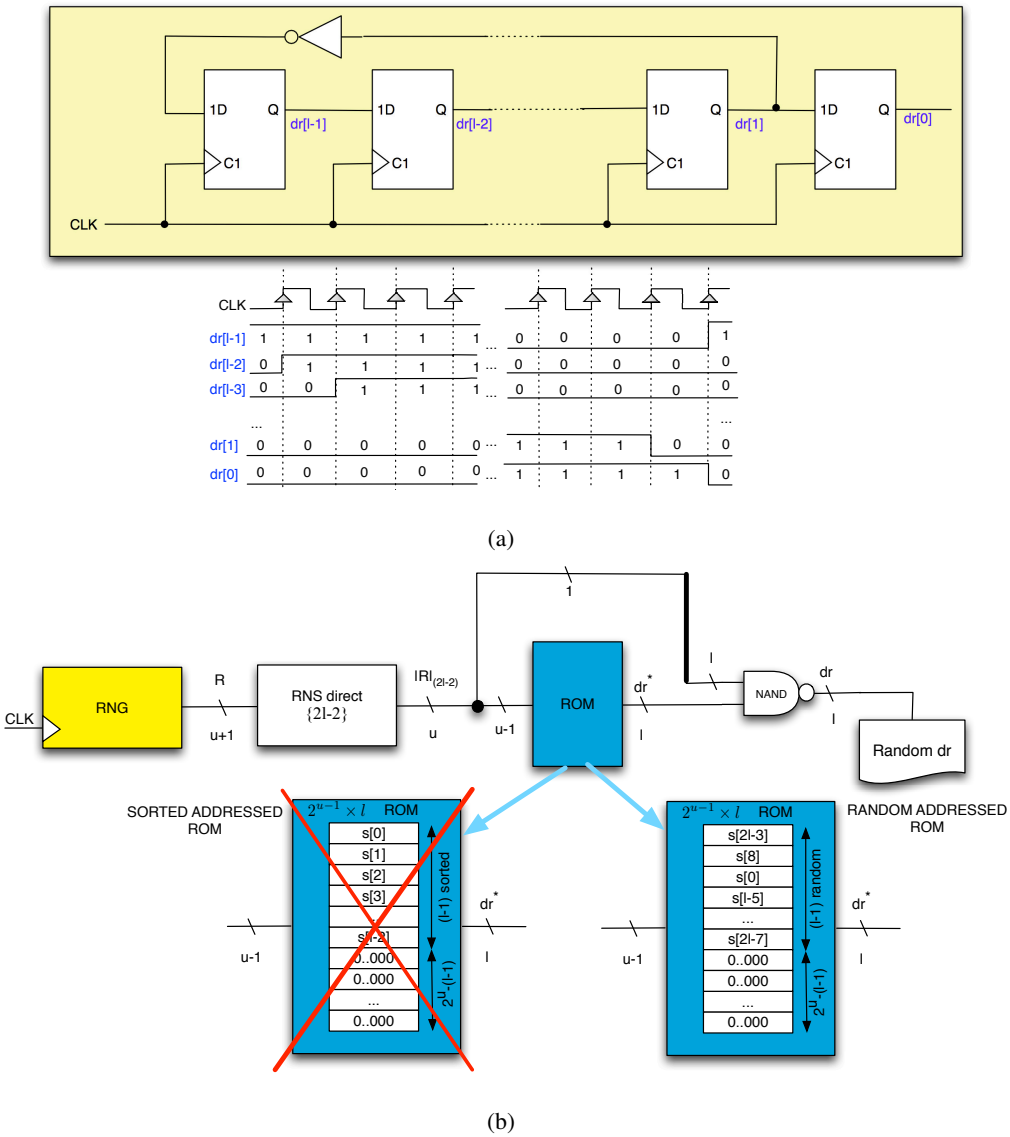


Fig. 5. (a) Modified Johnson counter, (b) proposed RNG.

invalid inputs. The $(l-1)$ sequences are chosen in order to derive secure values to prevent possible power attacks based on power patterns (e.g., template attacks [26]). Sorted $(l-1)$ sequences in the ROM addressing will provide distinctive patterns in the power profiles. Therefore, random $(l-1)$ sequences for ROM addresses are chosen as presented in Figure 5(b); and, *iv)* The remaining complemented $(l-1)$ sequences are obtained the MSB of $|R|_{2l-2}$ is active in the NAND operation with the output of the ROM, herein denoted as d_r*. The above explains our random generation of d_r with $l-1$ bits for the required pattern with secure RNG.

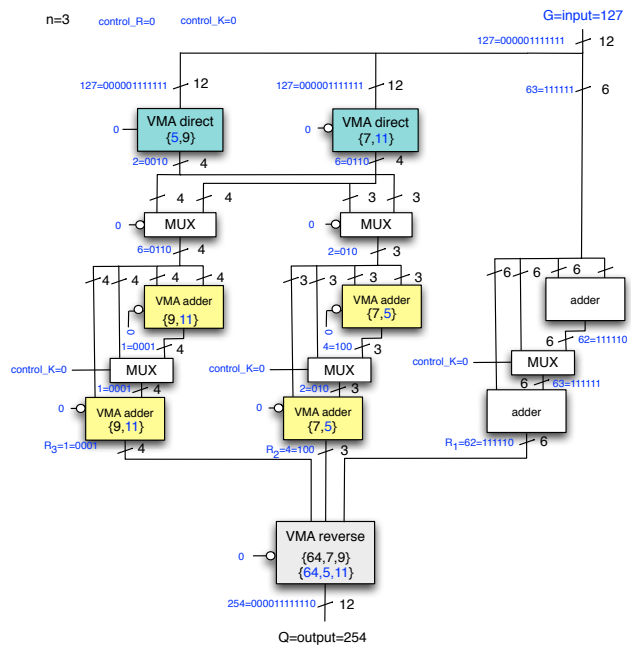


Fig. 6. DARNS with $n = 3$, control_R=0 and control_K=0

An illustrative example is provided in Figure 6 for $n = 3$, and moduli sets $\{2^{2n}, 2^n \pm 1\}$ and $\{2^{2n}, 2^n \pm 3\}$. The signals *control_R* and *control_K* are set to '0', and the circuitry associated with the *ENABLE* signal is not shown to avoid complexity. The input G, of value 127, with 12-bit length is directed to both direct VMA components. These components compute the modulo operation $\{2^n - 3, 2^n + 1\} = \{5, 9\}$ and $\{2^n - 1, 2^n + 3\} = \{7, 11\}$. Since *control_R* is 0, the modulo operations 5 and 11 for 127, marked in blue, are computed in these converters, $|127|_5 = 2$ and $|127|_{11} = 6$. Both residues are processed by a first level of MUX'es to derive a channel of $(n + 1)$-bits associated to $m_3 = \{2^n - 3\}$, and another with n-bits, associated to $m_2 = \{2^n - 3\}$ in the arithmetic unit. A third channel of $2n$-bits, associated to $m_1 = \{2^{2n}\} = 64$, is included by inserting the $2n$-LSB array of the input 127, $|127|_{64} = 63$.

The doubling is carried out by modulo adding twice the residue in each channel, whereas the extra addition is selected by the *control_K* in a second level of MUX'es. Due to the *control_K* is 0, the extra addition is not derived. The doubling operation derives into $R_3 = |6 + 6|_{11} = 1$, $R_2 = |2 + 2|_5 = 4$ and $R_1 = |62 + 62|_{64} = 62$.

Taking into account that $\hat{m}_1 = 55$, $\hat{m}_2 = 88$, $\hat{m}_3 = 44$, $\left|\hat{m}_1^{-1}\right|_{m_1} = 7$, $\left|\hat{m}_2^{-1}\right|_{m_2} = 4$ and $\left|\hat{m}_3^{-1}\right|_{m_3} = 1$ it is possible to derive $V_1 = 6$, $V_2 = 44$ and $V_3 = 5$ from Eq. (14) and Eq. (17). By substituting these V_i, $1 \leq i \leq 3$, values in the Eq. (14) are derived $v_1 = |6 \times 62|_{55} = 42$, $v_2 = |44 \times 4|_{55} = 11$ and $v_3 =$

$|5 \times 1|_{55} = 5$. Therefore, by concatenating $\left\lfloor \frac{Q}{2^{2n}} \right\rfloor 2^{2n} = |42 + 11 + 5|_{55}\, 64 = 3 \times 64 = 192$ to $R_1 = 62$ as presented in Eq. (13), is obtained the final result of the doubling operation $Q = 192 + 62 = 254$.

VI. Experimental Setup

The DARNS Multi-modulo architecture, explained in Section IV, is integrated with a Random Number Generator (RNG) to create the RTL version of the synthesizable DARNS architecture, as shown in Figure 7. For proof of concept, we utilize the Linear Feedback Shift Register (LFSR) as the pseudo random number generator, providing its seed from the 'date' command in linux. Design compiler synthesizes the design and creates the synthesized DARNS architecture, together with switching information, parasitics and other necessary files. Modelsim reads these files and the synthesized DARNS architecture to create the activity file (i.e., VCD file). Power measurements are performed on the VCD file using PrimeTime. Tools to perform DPA and SPA are developed in Matlab.

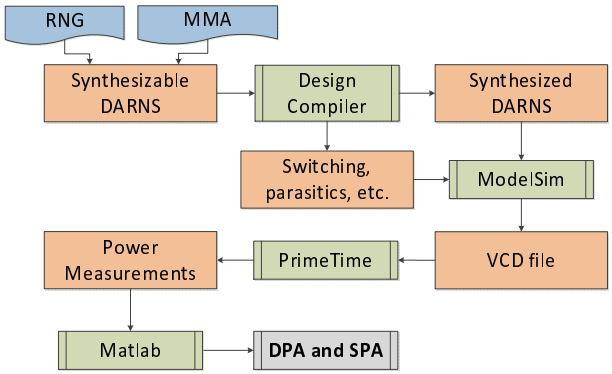
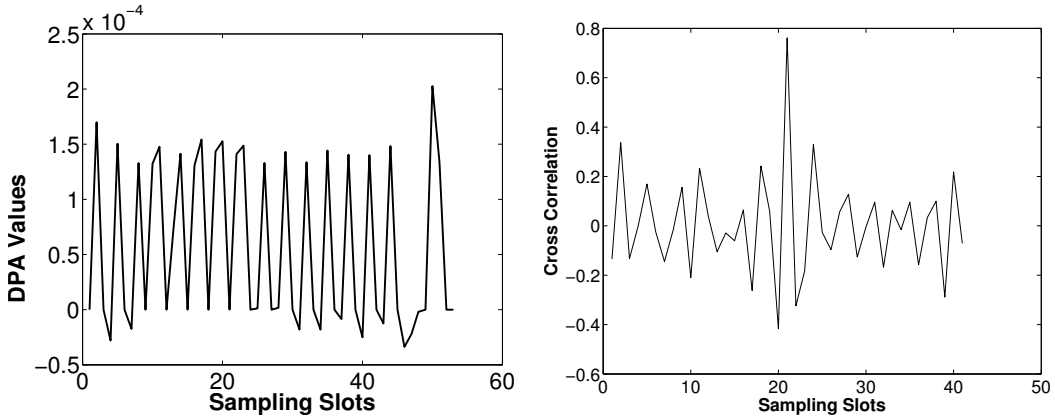


Fig. 7. Experimental Flow

VII. Results

We perform the Differential Power Analysis (DPA) on our DARNS architecture as demonstrated in [22] (referred to as SE attack). The two moduli architecture, as shown in Figure 2, with $n = 6$ is utilized for the attack. Figure 8(a) depicts the DPA plot which has the sampling slots (i.e., points) in x-axis, which is for every change in the *control_R* signal. The y-axis of the plot shows the DPA values, which are calculated based on the formulation presented in Eq. (1). As explained in Section III, we first use a key with all '1's, considering that as the known key. Similar to [22], we measure the power dissipation of DARNS with 500 power profiles (i.e., N_I = 500), for 500 random inputs. Again a key with '1' and then '0' repeated is used as the unknown key; 500 random inputs fed and power traces recorded. As shown

in [22], an unprotected version of the Double-and-Add (shown in Algorithm 1) generated peaks where the unknown key bits do not match with the known key bits. Our DPA attack in the DARNS architecture presented in Figure 2 clearly shows that there are no significant peaks, even when the bits do not match between the known key and the unknown key, because of the randomness and parallelism in DARNS.



(a) (b)

Fig. 8. (a) Differential Power Analysis, and (b) Cross Correlation Analysis for SPA.

As mentioned previously, the Double-add-Add has been attacked using Simple Power Analysis (SPA) by just observing the power profile (i.e., double and add operations consume different amount of power, hence will create significant distinctive patterns in the power profile). To evaluate such an attack for DARNS, we adopt the cross correlation analysis presented in [27]. Figure 8(b) shows the cross correlation analysis of two random power traces, one from the unknown key side and the other from the known key. The x-axis indicates the sampling slots, similar to Figure 8(a), whereas the y-axis shows the cross correlation factor. The plot indicates that there are no significant correlation between the power traces, revealing that no specific patterns exist. A slightly higher peak is observed due to the nature of the cross correlation analysis.

Correlation Power Analysis (CPA) [28] has been well known for out-performing DPA. We perform CPA analysis on our solution to further verify its security. Equation 18 depicts the formulation for CPA, where ζ_j reveals the correlation magnitude at a sampling slot j (power traces are sampled and each sample time is referred to as a slot). We feed two different keys (similar to the above DPA attack), k_{10}, a key with '1' and then '0' repeated and k_{11} all '1's. The CPA analysis should reveal significant correlation for equal key bits between both these keys for the attack to succeed. There are D number of input data fed to DARNS circuit with key k_{10}, generating D power traces. The same set of D inputs are fed with

key k_{11} to generate another D number of power traces. Power traces are denoted as h and t for k_{10} and k_{11} cases respectively. $\hat{h_j}$ and $\hat{t_j}$ represent the mean value of all the power samples at sampling slot j for k_{10} and k_{11} cases respectively. Both h and t should be of the same sample size.

$$\zeta_j = \frac{\sum_{d=1}^{D}(h_{d,j} - \hat{h_j}).(t_{d,j} - \hat{t_j})}{\sqrt{\sum_{d=1}^{D}(h_{d,j} - \hat{h_j})^2 . \sum_{d=1}^{D}(t_{d,j} - \hat{t_j})^2}} \tag{18}$$

Figure 9 depicts the resulting CPA plot. Overall, the correlations show very little difference (except one ghost peak [28]), indicating that there is no significant correlation in the power profiles when using k_{10} and k_{11}. This shows that DARNS is protected against CPA as well, showing no hints on the similarity between key bits (interested readers are referred to [28] for details on CPA).

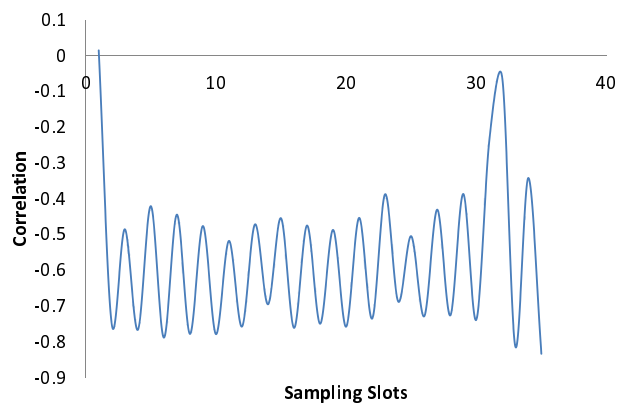


Fig. 9. Correlation Power Analysis

Our DARNS architecture (including the Inverse and Direct components) consumes around 64,043 cells in 90nm technology, whereas the DARNS with arithmetic (which is our main point of interest) takes 5,795 cells. The average dynamic power consumed in DARNS with arithmetic is 3.40mW and leakage power 19.52uW. The SMA architecture, on the other hand, has 78.44% less area than DARNS, however will not protect DPA or SPA. The standard architecture using adders, costs 1,263 cells in area and 1.09mW, 4.90uW in dynamic and leakage power, respectively, however still vulnerable to power analysis. A delay of 2.91ns is observed in DARNS, while the standard produced a delay of 1.78ns per key bit cycle. Since it is well proven that the RNS circuits are faster and consume less power for a large set of data compared to a standard implementation [12], [10], the performance of the DARNS is expected to be better than the standard one for large values of n.

ACKNOWLEDGEMENTS

This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2011

VIII. CONCLUSION

This paper presents a novel Multi-modulo parallel RNS implementation which chooses different moduli sets randomly. Such a randomness and parallelization prevents the Differential Power Analysis (DPA), Simple Power Analysis (SPA) and Correlation Power Analysis (CPA) in the Double-and-Add operation of the Elliptic Curve Cryptography. DPA, SPA and CPA analysis are demonstrated to prove the security of our DARNS architecture. Our architecture is not only secure, but perform better for large number of inputs, consume less power, benefiting from the inherent properties of the RNS.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Lecture Notes in Computer Science*, 1999.

[2] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *WOST*, 1999, pp. 17–30.

[3] R. Afreen and S. C. Mehrotra, "A review on elliptic curve cryptography for embedded systems," *CoRR*, 2011.

[4] E. Oswald, "Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems," in *CHES*, 2002.

[5] J.-S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," in *CHES*, 1999.

[6] K. Okeya and K. Sakurai, "Power Analysis Breaks Elliptic Curve Cryptosystems even Secure against the Timing Attack," in *INDOCRYPT*, 2000.

[7] A. Miyaji, T. Ono, and H. Cohen, "Efficient Elliptic Curve Exponentiation," in *IICSC*, 1997, pp. 282–290.

[8] Y. Sakai and K. Sakurai, "Simple Power Analysis on Fast Modular Reduction with Generalized Mersenne Prime for Elliptic Curve Cryptosystems." *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 2006.

[9] M. Koschuch, J. Lechner, A. Weitzer, J. Grobscädl, A. Szekely, S. Tillich, and J. Wolkerstorfer, "Hardware/software co-design of elliptic curve cryptography on an 8051 microcontroller," in *CHES*, 2006.

[10] D. Schinianakis, A. Kakarountas, and T. Stouraitis, "A new approach to elliptic curve cryptography: an RNS architecture," in *MELECON*, 2006.

[11] H. Pettenghi, L. Sousa, and J. Ambrose, "Efficient implementation of multi-moduli architectures for binary-to-rns conversion," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, 2012, pp. 819 –824.

[12] V. Paliouras and T. Stouraitis, "Systematic Design of Multi-Modulus/Multi-Function Residue Number System Processors," in *ISCAS*, 1994.

[13] Y. Zhou and D. Feng, "Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing," 2005.

[14] M. Izumi, J. Ikegami, K. Sakiyama, and K. Ohta, "Improved countermeasure against Address-bit DPA for ECC scalar multiplication," in *DATE*, march 2010, pp. 981 –984.

[15] K. Itoh, T. Izu, and M. Takenaka, "A Practical Countermeasure against Address-Bit Differential Power Analysis," in *CHES*, 2003, pp. 382–396.

[16] J. Bajard, S. Duquesne, and M. Ercegovac, "Combining leak–resistant arithmetic for elliptic curves defined over F_p and RNS representation," in *IACR Eprint archive*, 2010.

[17] Z. Lim, B. Phillips, and M. Liebelt, "Elliptic curve digital signature algorithm over GF(p) on a residue number system enabled microprocessor," in *TENCON*, 2009.

[18] E. Oswald and M. Aigner, "Randomized addition-subtraction chains as a countermeasure against power attacks," in *CHES*, 2001.

[19] J. A. Ambrose, H. Pettenghi, and L. Sousa, "DARNS: A Randomized Multi-modulo RNS Architecture for Double-and-Add in ECC to prevent Power Analysis Side Channel Attacks," in *ASP-DAC*, 2013 (accepted-to be published).

[20] C. H. Gebotys and B. A. White, "Methodology for attack on a Java-based PDA," in *CODES+ISSS*, 2006.

[21] B. Gierlichs, L. Batina, and P. Tuyls, "Mutual Information Analysis – A Universal Differential Side-Channel Attack," 2007.

[22] K. Itoh, T. Izu, and M. Takenaka, "Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA," in *CHES*, 2003.

[23] N. S. Szabo and R. I. Tanaka, "Residue arithmetic and its applications to computer technology," *McGraw-Hill, New York*, 1967.

[24] G. Alia and E. Martinelli, "Designing multioperand modular adders," *Electronics Letters*, vol. 32, no. 1, pp. 22 –23, 1996.

[25] B. Parhami, *Computer arithmetic: algorithms and hardware designs*, ser. The Oxford Series in Electrical and Computer Engineering Series. Oxford University Press, 2009. [Online]. Available: http://books.google.pt/books?id=tEo_AQAAIAAJ

[26] C. Rechberger and E. Oswald, "Practical Template Attacks," in *WISA*, vol. 3325, 2004, pp. 443–457.

[27] J. A. Ambrose, R. G. Ragel, and S. Parameswaran, "RIJID: Random Code Injection to Mask Power Analysis based Side Channel Attacks," in *DAC*, 2007, pp. 489–492.

[28] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model." in *CHES*, 2004, pp. 16–29.