A Cryptographic Coarse Grain Reconfigurable Architecture Robust Against DPA

Daniel Mesquita[1], Benoît Badrignans[2], Lionel Torres[2], Gilles Sassatelli[2], Michel Robert[2], Fernando Moraes[3]

[1]Instituto de Engenharia de Sistemas e Computadores – INESC-ID
Lisboa – Portugal
mesquita@inesc-id.pt

[2]Université Montpellier II, LIRMM
Montpellier –France
{surname}@lirmm.fr

[3]PUCRS – FACIN
Porto Alegre – Brasil
moraes@inf.pucrs.br

Abstract

This work addresses the problem of information leakage of cryptographic devices, by using the reconfiguration technique allied to an RNS based arithmetic. The information leaked by circuits, like power consumption, electromagnetic emissions and time to compute may be used to find cryptographic secrets. The results issue of prototyping shows that our coarse grained reconfigurable architecture is robust against power analysis attacks.

1. Introduction

The main idea of this work is that reconfigurable techniques may be used to give flexibility and improve security of cryptographic systems. Flexibility is an obvious necessity in the security domain, because cryptanalysis evolves as fast as cryptographic methods. However, nowadays is no longer sufficient make use of proven robust cryptographic algorithms. Now it is imperative to take care of the hardware implementation of these algorithms.

In 1998 an attack based on the information leaked by cryptographic circuits was proposed [1]. This kind of attack makes use of information like time to compute, electromagnetic emissions, temperature, power consumption, etc. They are called Side Channel Attacks (SCA) and can be used to retrieve secret keys of cryptographic devices.

For instance, the power consumed by a cryptographic engine has a strict relation with the data computed. This characteristic is due that logic gates consume current differently when switches from zero to one than from zero to zero. Then, an encryption process generates a power signature depending on the text encrypted and the cryptographic key used to do it. If the cryptographic hardware has no protection, a Simple Power Analysis (SPA) attack can be performed. But even with some countermeasures present, the Differential Power Analysis (DPA) may be still efficient.

This paper firstly gives an overview about DPA attacks and shows some countermeasures. Then, the Leak Resistant Reconfigurable Architecture (LR^{2}A) is explained. Afterwards a discussion about the security of the LR^{2}A is made, considering the arithmetic background of this approach. Finally some results, conclusions and further work are presented.

1-4244-0910-1/07/$20.00 c2007 IEEE.

2. DPA Attack

Differential power analysis consists not only visual (as the SPA), but also statistical analysis and error-correction methods to recover keys of hardware implementation of cryptographic algorithms [1].

The most of power consumption of an integrated circuit is due to the logical gates and the parasitic capacitance of the internal wires. But the variant part of power consumption is given by the data processed. So, the DPA has the ability to find the correlated data in hardware's power consumption, without requiring any information about the implementation details.

For a typical attack, an adversary repeatedly samples the target device's power consumption through each of several thousand cryptographic computations. These power traces can be obtained using high-speed analog-to-digital converters. Figure 1 illustrates this method used to attack a smartcard device, but experiences were made also against FPGA implementations.

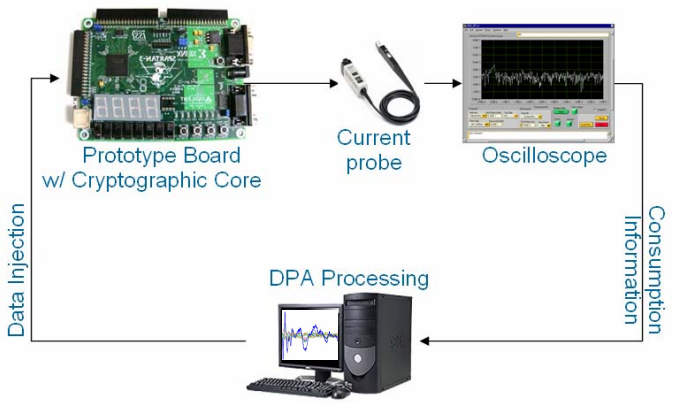

Figure 1 - A DPA attack platform

As an illustration, a DPA attack against a device implementing the DES algorithm is given. The Data

Encryption Standard (DES) [2] is used because of its widespread use and simplicity. The DES is composed by 16 rounds of substitutions, with initial and final permutations. In each of the 16 rounds, the DES encryption algorithm performs eight S-box operations. The 8 S-boxes take as input the XOR between the six key bits and the six bits of the R register and produce four output bits.

The attack consists in to guess the sub-keys at the input of the S-box, and predict the output. For example, a typical prediction is that the 6 bits entering S-box 1 are "100101". If correct, it allows the attacker to compute four bits entering the second round of the DES computation. If the assertion is incorrect, however, an effort to predict any of these bits will be wrong nearly to half the time.

For any of the four predicted bits, the power traces are divided into two subsets: one where the predicted bit value is 0, and one set where the predicted value is 1. Next, an average trace is computed for each subset, where the nth sample in each average trace is the average of the nth samples in all traces in the subset. Finally, the adversary computes the difference of the average traces.

If the original assumption is incorrect, the criteria used to create the subsets will be approximately random. Any randomly-chosen subset of a sufficiently-large data set will have the same average as the main set. As a result, the difference trace will be near to zero, and the adversary repeats the process with a new guess.

If the assumption is correct, however, choice of the subsets will be correlated to the actual computation. In particular, the second-round bit will have been '0' in all traces in one subset and '1' in the other. When this bit is actually being manipulated, its value will have a small effect on the power consumption, which will appear as a statistically-significant deviation from zero in the difference trace.

The main idea behind this method is that prediction of a single output bit leads the attacker to the 6 bits of the input sub-key, and then, to the rest of the key bits.

3. DPA Countermeasures

Several general approaches for reducing the flow of information through power consumption have been proposed. This might be accomplished by adding a secondary architecture to the chip that would do calculations on random numbers. This could mask the power consumed by the other part of the chip handling the encryption. But it is unclear whether enough randomness could be created to resist the more thorough statistical techniques used to break the cards' codes. Random calculations tend to average out over time and are easy for differential power analysis to remove.

At the architectural level, one solution could be to add parallel circuits to the chip that would mirror the real encryption calculations. For instance, if the real circuit is multiplying by the binary number 101, then the mirror circuit might multiply by 010 [3]. This would smooth out the power consumption because the power consumed by both parts together should be more constant. Still, it is unclear if all information can be blocked by this solution, because the mirroring is not perfect, due to physical synthesis aspects.

Some efficient algorithmic countermeasures have been presented both at the hardware (logic gates, analog) and algorithmic levels.

A. Hardware Countermeasures

The hardware methods to counteract DPA attacks differ expressively from the algorithmic ones. For the hardware approach the intermediate results of the cryptographic algorithm computation are not affected. As an alternative, the contribution of the hardware approach is to hide the attackable part of the power consumption with different noises. The noise addition has a direct relation with the needs of measurement. It does not avoid DPA attacks, but makes it quite more difficult. The effectiveness of the countermeasures against DPA is due to the fact that cryptographic devices are typically protected by a combination of algorithmic and hardware techniques, or only the hardware one [4].

In order to decrease the correlation between data inputs and the power consumption of a given circuit, we must be able to increase the samples needed in DPA. Two major hardware countermeasures in this sense have been proposed.

The lower signal to noise ratio (SNR) is, the lower is the correlation between the correct hypothetical current consumption and the real power consumption of the device. To reduce SNR there are some works that use special logic to minimize the data dependency of the current consumption.

In [5], [6] and [7] the balanced dual-rail logic is proposed. The basic idea is that a logic gate must consume an equivalent power, independently from the incoming input values. The SNR is reduced by this data-independent switching of the standard cells. Unfortunately, the experiments show that this goal is only partially reached. Dual-rail approach is not sufficient to guarantee a complete data independent power signature. One potential problem is that the gate loads may differ due to differences in routing. The design of each dual-rail gate must ensure equal input pin loads and balanced power usage. To achieve this, the process of grouping cells in the placement must be done carefully, which implies a high development effort. Besides that, the final circuit with dual-rail logic takes about tree times the area and two times the consumption of the original circuit.

The second hardware approach to prevent DPA attacks is to reduce the correlation between input data and power

consumption by randomly disarrange the moment of time at which the attacked intermediate result is computed. If the time tc is different in every power trace, the correlation between the hypothetical power consumption and the real one is highly reduced. The countermeasure proposed by [8] lies on the insertion of random delays. The countermeasure proposed in [4] counteracts the DPA by using power-managed blocks to mask the power consumption. Both approaches, with the [9] and [10] works, difficult the DPA attack. But, as shown in [11], even if a direct calculation of the maximum probability of a given power consumption occurring at a given time is not practical, it is always possible to approximate it empirically based on a software model of the countermeasure.

Another way to decrease the SNR is presented in [12], and consists in to mask the power consumption not by randomizing the consumption or creating noise but by generating, at the transistor level, a constant consumption. This approach is a little similar with the work proposed by Adi Shamir in [13], but the circuit described in [13] considers only if the attacker probes the Vcc, because the Gnd line remains vulnerable, when [12] provides a full masking in both Vcc and Gnd wires.

B. Algorithmic Countermeasures

There are several algorithmic (or software) countermeasures to thwart DPA attacks. Some of the first ones were proposed in [14], and the three proposed countermeasures are efficient against SPA and classical DPA attacks. For RSA cryptosystems [15] the first method described by Coron is applicable, and the second one is just an adaptation of the Chaum's blind signature [16]. The third method is only suitable for ECC (Elliptic Curve Cryptosystems). But the recently proposed Refined Power Analysis (RPA) [17] overrules these countermeasures.

The BRIP method counteracts the RPA but is also targeted to ECC, not tailored to work with the widely used RSA algorithm [18]. The message blinding proposed by P. Kocher [19] seems to be an efficient countermeasure against the MRED [20] (an attack targeting CRT implementation of RSA).

In general, the countermeasures protecting the RSA algorithm of DPA attacks relies on message or exponent blinding. These methods contribute or not to the security of the system, depending on the way they are implemented and the kind of attack. Is not rare that defend against one attack may benefit another kind of attack.

So, the best way to counteract DPA attacks is to target the DPA principle: the correlation between the data computed and the power consumption. Differently of the works that generally proposes CRT to accelerate RSA, like [23], another proposes a full RNS representation to compute RSA [24], [25]. Besides the acceleration, a full RNS implementation of RSA can intelligently be used to counteract DPA and DFA attacks, as is shown in Section V.

4. The Leak Resistant Reconfigurable Architecture

We conceived a coarse grained reconfigurable architecture to execute the Leak Resistant Arithmetic (Section V) called "Leak Resistant Reconfigurable Architecture" (LR²A). Besides, the LR²A intends to be a flexible solution for cryptographic algorithms; also it can be viewed as a coarse grain reconfigurable architecture, because it is possible to reconfigure this architecture to perform other cryptosystems based on modular arithmetic, like ECC or RC6.

Furthermore, the LR²A can be easily modified to run supplementary applications, like data compression or image treatment, but the discussion about the flexibility exceeds this paper's scope.

The LR²A is built around three main structures: a configuration and data injection controller, some homogeneous processing elements (PE), and memory resources. A controller is charged of data injection and configuration control. There is k PE, where k is given by the number of bases used for the LRA. The LR²A memory schema is the non-uniform memory access, i.e. the memory is distributed can be viewed as a local memory for each PE, but accessible for all PEs and the controller.

The following subsections describe the controller, the PE, the memory structure and the configuration model, where the robustness of the LR²A is highlighted.

A. The Controller

The main difference of the LR²A and some common reconfigurable architectures is that the LR²A is not only a loop-core for a specific algorithm class. The proposed reconfigurable architecture includes a controller to bring the configuration specific to each node, to inject the data into the distributed memory, and after that, to recover the computed data. The Figure 2 shows an overview of the LR²A.

To perform these operations the Plasma processor was chosen, due to its availability and the fact it has a C compiler, which makes easier the configuration model programming. The Plasma CPU is a open source processor, based on the MIPS R3000 instruction set, ant it has as an advantage the software compatibility with others processors used for embedded systems.

At the beginning, the controller brings the configuration for each PE, and saves into a data structure information about the status of each node. After, the initial data supposed to be computed is charged into the local memories. Since configuration and data are in place, the controller starts the computation. Finally, the controller is the element charged of to recuperate the computed data and to store it into the main memory.

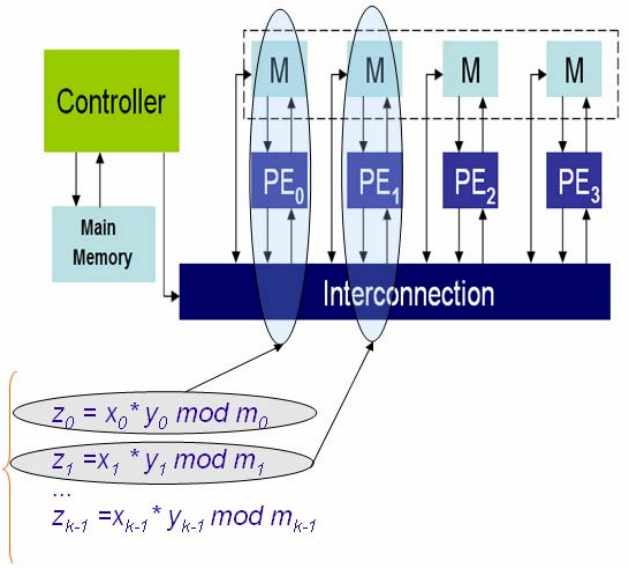

Figure 2 - The LR²A overview

For the next changes (I.e, data arriving, or other configuration needed), the controller consults current status of the PEs and its respective memories before send the new information. In fact; to control the LR²A consists in to run the configuration framework explained in the subsection III.C.

B. The Processing Element

The LR²A's processing element has a load-store architecture: the logic and arithmetic instructions are executed among internal registers only, while the memory access instructions execute either the reading from (load) or the writing to (store) one memory position.

Due to the load/store architecture option, the processor must have a relatively large set of data manipulation general-purpose registers, to reduce the number of memory accesses (this always represents a time penalty with regard to the processor internal operation). Regarding the instruction format, all instructions have exactly the same size, occupying one memory word each. The instruction contains the operation code and the operands specification, in case they exist.

Other characteristics present at the LR²A are common to the most risc-like processor:

1. Address and data size are of 32 bits.
2. Memory addressing is performed on a word basis.
3. The register bank contains 16 general-purpose registers, each of 32 bits.
4. There are 4 status flags named: negative, zero, carry, and overflow.
5. The instruction execution takes place in 2 to 4 clock cycles.

The processor datapath includes dedicated cryptographic specific operators. Because the LRA's basic operations are the modular ones, the following operations are hardwired and incorporated to the ALU:

- Modular reduction (X mod M)
- Modular reduction by 2^k factor (X mod 2^k)
- Modular addition ($X+Y$ mod M)
- Modular subtraction ($X-Y$ mod M)
- Modular multiplication ($X \times Y$ mod M)
- Modular multiplication by 2^k factor ($X \times Y$ mod 2^k)
- Shifts and rotations performed trough a Barrel shifter.

C. The Configuration Control Model

Frequently the efficiency of a reconfigurable architecture is compromised due to the absence of methods to control the reconfiguration process and the data injection.

To fill this gap a configuration model associated with the LR²A is proposed. The model is composed by several modules: configuration memory (CM), reconfiguration monitor (RM), reconfiguration dispatcher (RD), configuration scheduler (CS) and the central configuration control (CCC). This reconfiguration model is based on the Reconfigurable System Configuration Manager (RSCM) idea presented at [26]. The RSCM adapted for the LR²A can be depicted from Figure 3.

As the name indicates, the configuration memory is a reserved part of the controller's memory, which stores all configurations that could be needed to execute an application.

The Reconfiguration Monitor (RM) detects situations where reconfigurations need to be performed, the so-called reconfiguration events, and notifies CCC, which acts appropriately.

All information about PEs allocation and its corresponding tasks is stored at the Table of Resources Allocation (TRA), associated with the CCC. The CCC receives requests of the RM and dispatches the necessary services at the CS and RD.

The Configuration Scheduler (CS) module is responsible to determine which configuration is the next to be configured. This module receives service requests from the CCC. It stores a data structure with information about configurations dependence, called Table of Dependencies and Descriptors (TDD).

Finally, the RD module is charged of the reconfiguration itself, sending the appropriated code for each PE, or, for a group of PEs (i.e. a broadcast of configurations is possible).

As mentioned in [26], the RSCM can be implemented in hardware, software or in a combined hardware/software approach. Because the configurations are not frequent in the LR²A, we opted to implement the RSCM in software, running at the Plasma processor.

This configuration model allows centralizing all control and synchronization, avoiding problems related with cache

coherence. The controller "knows" what is happening in each PE because each PE has a data structure associated and updated by the RSCM.

5. Leak Resistant Arithmetic

The Leak Resistant Arithmetic (henceforth called LRA) is based in the RNS representation and the RNS Montgomery's modular multiplication proposed in [22].

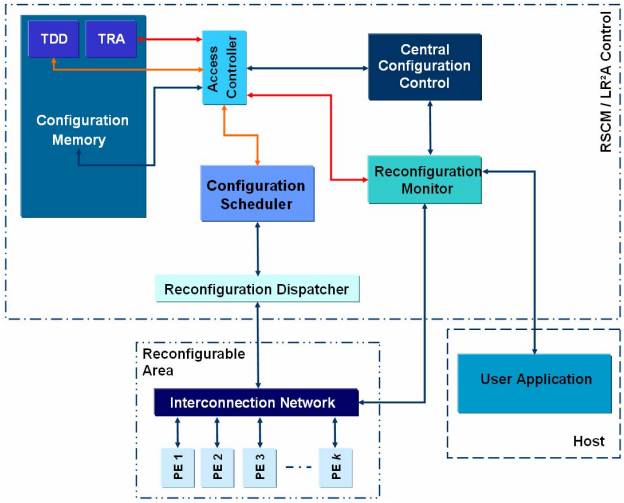

Figure 3 - The LR²A control based on the RSCM

A. Residue Number System

A Residue Number System (RNS) relies on the Chinese Remainder Theorem (CRT). This theorem indicates that is possible represent a large integer using a set of smaller integers, so that computation may be performed more efficiently.

A residue number system is defined by a set of k integers constants, $\{m_1, m_2, m_3 .. m_k\}$, referred to as the moduli. The moduli must all be co-prime; so in particular no modulus may be a factor of any other. Let M be the product of all the m_i. Any arbitrary integer X smaller than M can be represented in the defined residue number system as a set of k smaller integers $\{x_1, x_2, x_3 ... x_k\}$ with $x_i = X$ modulo m_i representing the residue class of X regarding that modulus.

So, arithmetic operations can be made using this representation, as addition and multiplication, for instance, absolutely in parallel, because once represented in RNS the operations does not generates any carry. For example, to perform an addition between A and B in RNS with a base composed by m_i $(0 < m \leq k)$ is: $Sum_i = (a_i + b_i)$ modulo m_i.

The conversion from RNS to decimal is not so trivial, but is needed only after all computations performed, so its cost is amortized. To transform an RNS number in a decimal representation the following formula is needed:

$$x = \sum_{i=1}^{k} x_i M_i \left| M_i^{-1} \right|_{m_i} \bmod M \qquad (1)$$

For cryptographic applications, modular reduction (x mod M), modular multiplication (x.y mod M) and modular exponentiation (xy mod M) are the most important operations. They can be calculated using the Montgomery's algorithm, modified to RNS representation, as described in the next sub-section.

B. Montgomery Modular Multiplication

The version of the Montgomery's modular multiplication presented below was proposed in [24].

In the RNS representation the value M is taken from:

$$M = \prod_{i=1}^{k} m_i \qquad (2)$$

So, M is chosen as the Montgomery constant instead β^k in the classical representation. Then, with A, B, R and N represented in RNS within the base $\beta_1=\{m_1, m_2, m_3, ... m_k\}$. The result of the algorithm must be: $R = A.B.M_1^{-1} \bmod N$.

However, the value M_1^{-1} cannot be computed in β_1. So another base β_2 is defined as an extension of β_1, with k extra moduli all co-primes among them and with β_1. So, before calculate (Algorithm 1, in Figure 2, points 3 and 5) M_1^{-1} a base extension from β_1 to β_2 is performed.

So, the Equation 2 describes the algorithm for Montgomery's modular multiplication in RNS. As inputs we have two RNS bases β_1 and β_2, such that M and M' can be computed as the product of the moduli that composes respectively β_1 and β_2. The inputs A, B and N are also represented in both β_1 and β_2 bases. Besides a redundant modulus m_r such that $\gcd(m_r, m_i)=1$ is needed. Also, N and M must be co-primes. The result is given by R in β_1.

$$1: T \leftarrow A \otimes_{RNS} B \ \ in \ \ \beta_1 \ \ and \ \ \beta_2$$
$$2: Q \leftarrow T \otimes_{RNS} (-N^{-1}) \ \ in \ \ \beta_1$$
$$3: Extend \ \ Q \ \ from \ \ \beta_1 \ \ to \ \ \beta_2$$
$$4: R \leftarrow (T \oplus_{RNS} Q \otimes_{RNS} N) \otimes_{RNS} M^{-1} \ in \ \beta_2$$
$$5: Extend \ \ R \ \ from \ \ \beta_2 \ \ to \ \ \beta_1$$

Algorithm 1 – Modular multiplication in RNS with base extension (LRA's core)

The points 1, 2 and 4 of the algorithm consist of full RNS operations that can be realized in parallel. Therefore, the most complex operations rely on the base extensions (points 3 and 5). There are some methods to compute a base extension, but here the Mixed Radix System, described in [21] that has the advantage that it requires only

a of k values for each base modification [22]. Due to space constraints, the algorithm for modular exponentiation was omitted in this paper, but details can be obtained in the same reference [22].

C. *The security provided by the LRA*

Besides performance due to the intrinsic parallelism, RNS algorithms provide also the possibility of randomize the basis: the algorithm's robustness relies on this concept. The LRA proposes two approaches of data randomization: one at the circuit level (spatial randomization) and the data level (arithmetic masking). They represent a good trade-off between security and implementation cost. The considered approaches are:

- **Random choice of initial bases**: Randomization of the input data is provided by randomly choosing the elements of β_1 and β_2 before each modular exponentiation.
- **Random change of bases before and during the exponentiation:** A generic algorithm is proposed in [22], offering many degrees of freedom in the implementation and at the security level.

The main goal of these approaches is to lead to a randomization of all intermediate data computed at the cryptographic circuit for the same input data and output. Based on the same principle of the DPA, if the data change during an operation, consequently the power consumption becomes non constant, thwarting DPA attacks. The security of this method was demonstrated in [22].

6. Results

A. *Area and Performance*

The Table I shows the size of each PE for different datapath sizes. Area is given in thousands of equivalent logic gates (elg).

TABLE I
Synthesis for PEs with different datapath sizes

Datapath (bits)	elg (k)	Clock (MHz)
16	4,3	60
32	8,5	40
64	15,8	30
128	32	20

Taking the 32bits datapath configuration, and considering the size of the controller being around 40k gates, a LR²A composed by 32 PEs (capable to perform 1024bits exponentiation), has an area about 352k gates. Comparing with state-of-art hardware accelerators for the same purpose, the LR²A takes five to eight times the commonly use area.

But in terms of performance, as shown in the Figure 4, for cryptographic keys larger than 1024 bits, the LR²A provides more interesting response times. The comparison was made taking account the square and multiply method to perform modular exponentiation. The classic implementation concerns a modular exponentiation calculated with a specific circuit implementing the Montgomery algorithm for modular multiplication, with a word size of 32 bits, and running at 41MHz. On the other hand, the LR²A implemented is a 32 bit datapath version, with 32 processors running at 40MHz.

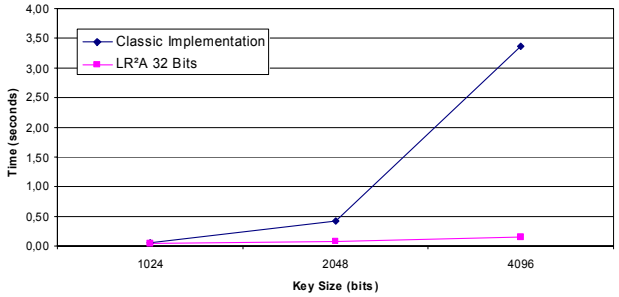

Figure 4 – Comparison between LR²A and a classic implementation of the "a^e mod N"

Meanwhile, comparing the performance of the LR²A with other state-of-art architectures to compute modular exponentiation, the Figure 5 shows clearly that the improvements in security achieved by the LR²A do not compromise the performance. LR²A is capable to compute one 1024 bits modular exponentiation in the same time that the most performing architecture reviewed by us.

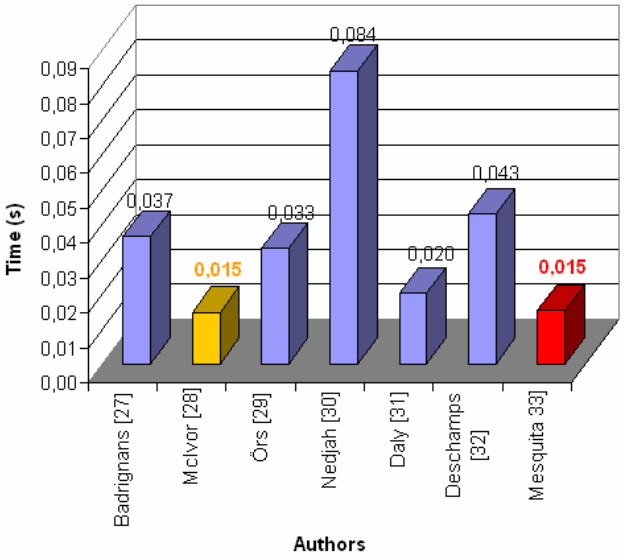

Figure 5 - LR²A performance vs other state-of-art implementations of "a^e mod N"

Concerning the area, the report is 5 times for 2048 bits and 10 times for 4096. This is the cost of the security. The classic implementation does not contain any countermeasure against DPA, while the LR²A incorporate the robustness needed to counteract this attack.

B. *Robustness*

Regarding the robustness aspect, by changing intermediate results via base extension within the LRA method, our architecture consumes differently for the same data set. It means that we compute an exponentiation (A^e mod N) within different basis. In the first part we can see the consumption for the first base β_1, while in the second part, the consumption is for the base β_2. The arrows highlight a visible different consumption.

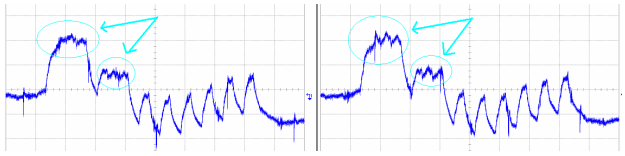

Figure 6 - Different traces for each base, with the same computed A x B mod M

By making a difference of the two curves, it is possible to see that the computed data is no longer related with the power consumption. In the Figure 7 we have the difference of two computations of the same decimal data (I.e. the same values for 'A' and for the exponent 'e'). The Figure 7-(a) shows the difference between two computations of (A^e mod N) in a classical representation: it means that for the same data set we always have the same consumption.

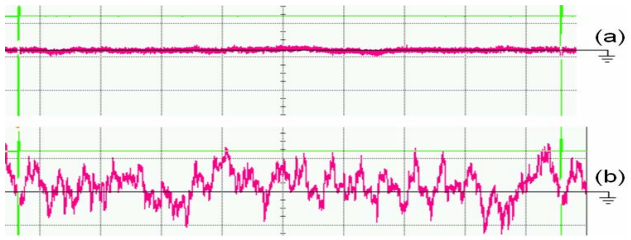

Figure 7 – The robustness analysis of the LR²A:

On the other hand the Figure 7-(b) highlights the difference of two modular exponentiation with the same 'A' and 'e', but expressed in the LRA representation. As the first exponentiation is made in the base β_1, and the second one in base β_2, the resulting consumption is different, avoiding a DPA analysis.

C. *Reconfiguration issue*

The main security point of the LR²A is the base extension. This is realized through a dynamic reconfiguration. As can be viewed in the Algorithm 1, in a first moment the architecture executes RNS operations (points 1, 2 and 4). But when a base extension is required, due to its algorithmic nature, the architecture must be reconfigured to execute new instructions. So, the RSCM intervenes to manage the reconfiguration procedure.

The Reconfiguration Monitor determines the moment when the base extension module is needed and notifies the Central Configuration Control. The CC calls the Configuration Scheduler, which verifies if the base extension module is present in the Reconfiguration Memory. Meanwhile, the necessary data is stored in the local memories. So, the Reconfiguration Dispatcher brings the new configuration to the PEs.

After the base extension procedure finished, the RNS operations must be restored. This is accomplished by the RSCM, repeating the steps described in the previous paragraph.

Acting this way the system is dynamically reconfigured, and the context is never lost due to the traces of each RSCM operation stored in the Table of Dependencies and Descriptors and in the Table of Resources Allocation.

7. Summary and Conclusions

The Leak Resistant Arithmetic improves the robustness of cryptographic applications counteracting the principle of some hardware attacks, like DPA. Due its nature, the LRA leads to a coarse grain reconfigurable architecture, requiring important hardware resources.

Even with area penalties, the LR²A is competitive in terms of performance, equivalent to state-of-art of cryptographic accelerators. But in cryptographic applications, security always has a cost. The LR²A show that is possible to improve security remaining flexible and performing.

By slightly modifying the PEs it is possible to run other cryptographic or compression algorithms, but the main further work is to make the PEs still more flexible, by adding a fine grain reconfigurable area in each PE. In this way, with a reconfigurable datapath, the LR²A could implement a wide spectre of applications.

References

[1] P. Kocher, al. "Differential Power Analysis : Leaking Secrets". *Advances in Cryptology: CRYPTO'99*, pp. 388-397. 1999.

[2] – . " Data Encryption Standard (DES)". Federal Information Processing Standards Publications (FIPS PUBS) Nº 46-3. EUA.October 25, 1999.

[3] C. Walter. "Sliding Windows Succumbs to Big Mac Attack". *Cryptographic Hardware and Embedded System: CHES'01,* pp286-299. 2001.

[4] L. Benini, et al. "Energy-aware design techniques for differential power analysis protection". *Design Automation Conference: DAC '03*. Anaheim, USA. June, 2003.

[5] A. Razafindraibe, et al. "Asynchronous Dual rail Cells to Secure Cryptosystem Against SCA". *Sophia-Antipolis Forum on MicroElectronics*. Nice, France, 2005.

[6] H. Saputra, et al. "Masking behavior of DES encryption". *Design, Automation and Test Europe – DATE '03*. Munich, Germany, 2003.

[7] M. Simon, et al. "Balanced Self-Checking Asynchronous Logic for Smart Card Applications", *Microprocessors and Microsystems Journal*, 27. Elsevier, pp 421-430, October 2003.

[8] C. Clavier, et al. "Differential Power Analysis in the presence of hardware countermeasures". *Cryptographic Hardware and Embedded Systems – CHES '00*. Pp 252-263, 2000.

[9] J. Irwin, et al. "Instruction stream mutation for non-deterministic processors". *International Conference on Application Specific Systems, Architectures and Processors – ASAP 2002*. IEEE press. Pp 286-295. 2002.

[10] D. May, et al. "Non-deterministic processors". *Information security and privacy – ACISP'01*. Sydney, Australia. July 2001.

[11] S. Mangard, "Hardware countermeasures against DPA – a statistical analysis of their effectiveness". *Topics in Cryptology – CT-RSA'04*. pp. 222 – 235. San Francisco, USA. 2004.

[12] D. Mesquita, et al. "Current Mask Generation: A New Hardware Countermeasure for Masking Signatures of Cryptographic Cores". *International Conference on VLSI: IFIP VLSI SoC '05*. Perth, Australia, 2005.

[13] A. Shamir. "Protecting smart cards from passive power analysis with detached power supplies". *Cryptographic Hardware and Embedded Systems, CHES'00.*Pp 71-77, 2000.

[14] J-S Coron. "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems". *Cryptographic Hardware and Embedded Systems, CHES'99*. Pp 292-302, 1999.

[15] R. Rivest, et al. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". *ACM Communications*, vol 21. pp 120-126. 1978.

[16] D. Chaum. "Security without identification: transaction systems to make Big Brother obsolete". *Communication of the ACM*. Vol. 8., n° 10, pp 1030-144. 1985.

[17] L. Goubin. "A refined power-analysis attack on ECC". *Publick Key Cryptography: PKC '03*. pp 199-210. 2003.

[18] M. Hideyo, et al. "Efficient Countermeasures against RPA, DPA, and SPA". *Cryptographic Hardware and Embedded Systems, CHES'04*. Pp 343-356, 2004.

[19] P. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". *16th Workshop in Cryptology: Crypto '96.* pp 104-113. Santa Barbara, USA. 1996.

[20] B. Boer. "A DPA Attack against the Modular Reduction within a CRT Implementation of RSA**".** *Cryptographic Hardware and Embedded Systems, CHES'02*. pp 228-243, 2002.

[21] H. Garner. "The Residue Number System". *IRE Transactions in electronic Computers*. Vol 8, pp. 140-147, 1959.

[22] J-C. Bajard, et al. "Leak Resistant Arithmetic". *Cryptographic Hardware and Embedded Systems CHES'04*. Pp 62-75, 2004.

[23] C. Kim, et al. "A CRT-Based RSA Countermeasure against Physical Cryptanalysis". *Conference on High Performance Computing and Communications: HPCC '05*. Pp 549-554, Naples, Italy, 2005.

[24] J-C. Bajard, et al. "A Full RNS Implementation of RSA".*IEEE Transactions on Computers.* Vol. 53, n° 6, pp. 769-774. 2004.

[25] M. Ciet, et al. "Parallel FPGA implementation of RSA with residue number systems – can side-channel threats be avoided?". *46th. International Midwest Symposium on Circuits and Systems: MWSCAS '03*. Cairo, Egypt, December 2003.

[26] E. Carvalho, et al. "Reconfiguration Control for Dynamically Reconfigurable Systems". *Conference on Design of Circuits and Integrated Systems*: *DCIS '04*. Bordeaux, France, 2004.

[27] B. Badrignans, D. Mesquita, J-C. Bajard, L. Torres, G. Sassatelli, and M. Robert. A parallel and secure architecture for asymetric cryptography. In Proceedings of ReCoSoC, page in press, Montpellier, France, 2006.

[28] C. McIvor, M. McLoone, J. McCanny, and W. Marnane. Fast montgomery modular multiplication and RSA cryptographic processor architectures. In Proceedings of the Asilomar Conference, Pacific Groove, USA, 2003. IEEE Computer Society.

[29] S. Örs, L. Batina, B. Preneel, and J. Vandewalle. Hardware implementation of a montgomery modular multiplier in a systolic array. In IPDPS, page 184, Nice, France, 2003. IEEE Computer Society.

[30] N. Nedjah and L. Mourelle. A review of modular multiplication methods and respective hardware implementations. Informatica, 30 :111–130, 2006.

[31] A. Daly and W. Marnane. Efficient architectures for implementing montgomery modular multiplication and RSA modular exponentiation on reconfigurable logic. In Proceedings of the FPGA '02, pages 40–49, 2002.

[32] J-P. Deschamps and G. Sutter. Fpga implementation of modular multipliers. In Proceedings of the DCIS '02, pages 107–112, 2002.

[33] D. Mesquita, B. Badrignans, L. Torres, G. Sassatelli, M. Robert, and F. Moraes. A leak resistant soc against side channel attacks. In Proceedings of ISSoC, page in press, Tampere, Finlande, 2006.