A LEAK RESISTANT ARCHITECTURE AGAINST SIDE CHANNEL ATTACKS

Daniel MESQUITA[1], Benoît BADRIGNAN[1,2]S, Lionel TORRES[1], Gilles SASSATTELL[1], Michel ROBERT[1], Jean-Claude BAJARD[1], Fernando MORAES[3]*

[1]**Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)**
Université Montpellier II
161 rue Ada, 34392 – Montpellier – France
{*LASTNAME*}@lirmm.fr

[2]**NETHEOS**
Cap Omega – rond Point Benjamin Franklin
34.000 Montpellier – France
b.badrignans@netheos.net

[3]**Faculdade de Informática (FACIN)**
Pont. Universidade Católica do RS (PUCRS)
Av. Ipiranga 6681 Prédio 30 – 90.619-900
Porto Alegre – Brasil
moraes@if.pucrs.br

ABSTRACT

Hardware implementations of cryptographic algorithms may leak some information that can be used to recover cryptographic keys. This work combines reconfigurable techniques with the recently proposed Leak Resistant Arithmetic (LRA) to thwart some Side Channel Attacks (SCA). The introduced architecture outcomes the performance of classical implementation of modular multiplication, for key size exceeding 2048 bits, with a reasonable extra area overhead. Nevertheless, this is not a drawback, but a cost, since the main issue of the proposed architecture is the improved robustness in terms of security.

1. INTRODUCTION

Information leaked by cryptographic processors, as computing time, power consumption and electromagnetic emissions during an encryption or decryption process are known as side channel. Side channel attacks (SCA) are based on side channel information.

In the past, cryptographic attacks were performed by analyzing known cyphertext (cyphertext-only attacks) or by choosing the plaintext and then seeing the result of encryption (chosen plaintext attacks). SCA appeared more recently and do not rely on purely cryptanalytic considerations, but rather focus the VLSI implementation of the cryptographic algorithm.

For instance, the power consumed by a cryptographic engine has a strict relation with the data computed. *This* characteristic is due to the fact that power consumed by every single logic gat differs depending on the output states before and after switching (i.e. switching from logic 0 to logic 1 or from logic 1 to logic 0) [1]. An encryption process generates a power signature depending on the encrypted text and the cryptographic key used for encryption. If the cryptographic hardware does not have any protection measures, 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 gives a brief introduction concerning DPA attacks and countermeasures. Then, as the base of the proposed architecture, the Leak Resistant Arithmetic is explained. This technique can be considered as an efficient algorithmic countermeasure, because this algorithm is based on run-time changes of the number base, which leads to changes in the intermediate results (without modifying the final result). Afterwards, a reconfigurable architecture resistant against some SCA attacks is introduced. Finally some results, conclusions and further works are discussed.

2. DPA ATTACK AND COUNTERMEASURES

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 significant part 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 secure device, but experiments were performed also against FPGA implementations.

As an illustration, a DPA attack conducted 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 an internal register (R register) and produce four output bits.

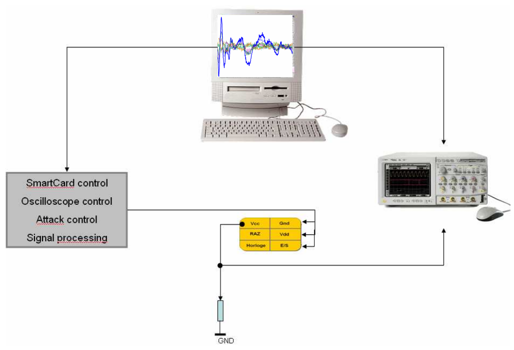

Figure 1 - A DPA attack platform

The attack consists in to guess the sub-keys at the input of the S-box, and predict the output. If the 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 hypothesis is correct, however, choice of the subsets will be correlated to the actual computation. In particular, the second-round bit would 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

But this paper focuses into algorithmic countermeasures. For the hardware ones, refer to [3].

In practice, to discover a 64bits key of a DES algorithm (without any countermeasure), around 100 plaintext are needed, with 1000 power acquisitions per text. Considering a fully automated process, in the worst case it takes less than 8 minutes to discover the secret key.

3.1. Algorithmic Countermeasures

There are several algorithmic (or software) countermeasures to thwart DPA attacks. Some of the first ones were proposed in [4], and the three proposed countermeasures are efficient against SPA and classical DPA attacks. For RSA cryptosystems [5] the first method described by Coron is applicable, and the second one is just an adaptation of the Chaum's blind signature [6]. The third method is only suitable for ECC (Elliptic Curve Cryptosystems). But the recently proposed Refined Power Analysis (RPA) [7] 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 [8]. The message blinding proposed by P. Kocher [9] seems to be an efficient countermeasure against the MRED [10], an attack targeting CRT (Chinese Remainder Theorem) 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. It is not rare that the defense 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 [13], another proposes a full RNS (Residue Number System) representation to compute RSA [14], [15]. Besides the acceleration, a full RNS implementation of RSA can intelligently be used to counteract DPA and DFA attacks, as shown below.

4. LEAK RESISTANT ARITHMETIC (LRA)

The Leak Resistant Arithmetic (henceforth called LRA) is based on the RNS representation and the RNS Montgomery's modular multiplication proposed in [12].

It uses the Residue Number System, which relies on the Chinese Remainder Theorem. The CRT indicates that is possible to represent a large integer using a set of smaller integers, so that computation may be performed more efficiently.

The version of the Montgomery's modular multiplication presented below was proposed in [14]. In the RNS representation the value M is taken from (1):

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

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 $\beta_1=\{m_1, m_2, m_3, \ldots m_k\}$. 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, [14] points 3 and 5) M_1^{-1} a base extension from β_1 to β_2 is performed.

So, the Figure 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.

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 [11] that has the advan that it requires only a of k values for each base modification [12]. Due to space constraints, the algorithm for modular exponentiation was omitted in this paper, but details can be obtained in the same reference [12].

4.1. 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 [12], 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 became random, thwarting DPA attacks. The security of this method was demonstrated in [12].

5. RECONFIGURABLE ARCHITECTURE

As the RNS implementation of RSA leads to parallel operations, we conceived a parallel reconfigurable architecture to run the Leak Resistant Arithmetic called "*Leak Resistant Reconfigurable Architecture*" (LR²A). Besides, the LR²A intends to be a flexible solution for cryptographic algorithms, so 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 processing, but the discussion about the flexibility exceeds the scope of the paper.

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 responsible of data injection and configuration control to/of the PEs. There are k PE, where k is given by the number of bases used for the LRA. The LR²A memory schema is non-uniform memory access architecture, i.e. the memory organization 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.

5.1. The controller

The main difference from the LR²A and some common reconfigurable architectures is that the LR²A is not only a loop-core (*asic, hardware, rever*) 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.

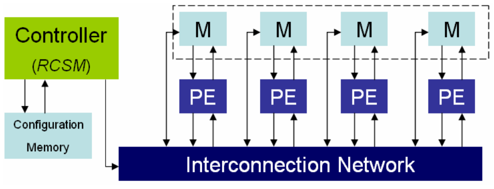

Figure 2 - The LR²A overview

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 an 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.

When the system starts, the controller reads the configuration for each PE from the configuration memory, sending it to the target PE. Also, the controller stores internally the status of each node into a data structure. After, the initial data supposed to be computed is loaded into the local memories. Since configuration and data are in place, the controller starts the computation. Finally, is the controller the element charged to recuperate the computed data and to store it into the main memory.

For the next changes (i.e., data arriving, or other configuration needed), the controller verifies the current status of the PEs and their respective memories before send the new information. In fact, controlling the LR²A consists in running the configuration framework explained in the reference [16].

5.2. The processing element

The LR²A's processing element is a load-store architecture similar to Plasma: the logic and arithmetic instructions are executed using internal registers only, while the memory access instructions execute either the reading from (load) or the writing to (store) one memory position. In fact, the processor's organization is similar to the Plasma, but the PEs are optimized for cryptographic operations, and consequently, simplest and smaller.

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 1 memory word each. The instruction contains the operation code and the operands specification, in case they exist. The processor datapath includes specific cryptographic operators. Because the LRA's basic operations are the modular ones, the following operations are hardwired and incorporated to the ALU.

6. RESULTS

The architecture was synthesized in AMS 0.35 technology, for different scenarios of key and datapath sizes. Taking the 32bits datapath configuration, and considering the size of the controller being around 40k gates, a LR²A composed by 32 PEs (capable of performing 1024bits exponentiation), occupies 352k gates on silicon. 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 3, for cryptographic keys larger than 1024 bits, the LR²A outcomes classical implementations. The comparison was made taking into account the square and multiply method to perform modular exponentiation. The reference implementation concerns a modular exponentiation calculated with a dedicated circuit implementing the Montgomery algorithm for modular multiplication, with a word size of 32 bits, and running at 40MHz. On the other hand, the LR²A implementation features are a 32 bit datapath version, with 32 processors running at 40MHz. For both versions the area corresponds the exponentiation control was not considered.

As shown in Figure 3, in terms of performance, the LR²A is equivalent to classic implementations of modular exponentiation for key sizes until 2048 bits. Beyond 4096 bits, the LR²A is clearly more interesting in performance, but with an area penalty.

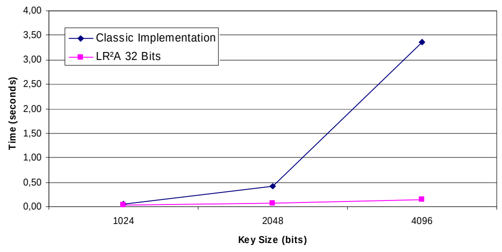

Figure 3 - LR²A versus a classic implementation of the modular exponentiation

The area 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.

7. CONCLUSIONS

The Leak Resistant Arithmetic improves the robustness of cryptographic applications counteracting the principle of some hardware attacks, like DPA. Due to its nature, the LRA leads to a parallel architecture, requiring important hardware resources. Even with area penalties, the LR²A is competitive in terms of performance, equivalent to state-of-art cryptographic accelerators.

8. REFERENCES

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

[2] – . " Data Encryption Standard (DES)". Federal Information Processing Standards Publications (FIPS PUBS) N° 46-3. EUA.October 25, 1999.

[3] D. Mesquita, J-D. Techer, et al. "Current Mask Generation: A New Hardware Countermeasure for Masking Signatures of Cryptographic Cores". *International Conference on Very Large Scale Integration: proceedings of IFIP VLSI SoC '05*. Perth, Australia, 2005.

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

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

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

[7] L. Goubin. "A refined power-analysis attack on elliptic curve cryptosystems". *Publick Key Cryptography: Proceedings of PKC '03*. Pp 199-210. 2003.

[8] M. Hideyo, M. Atsuko. "Efficient Countermeasures against RPA, DPA, and SPA". *Cryptographic Hardware and Embedded Systems, Proceedings of CHES 2004*. Pp 343-356, 2004.

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

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

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

[12] J-C. Bajard, L. Imbert, et al. "Leak Resistant Arithmetic". *Cryptographic Hardware and Embedded Systems, Proceedings of CHES 2004*. Pp 62-75, 2004.

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

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

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

[16] E-L. Carvalho, N. Calazans, et al. "Reconfiguration Control for Dynamically Reconfigurable Systems". *19th Conference on Design of Circuits and Integrated Systems*: *Proceedings of DCIS '04*. Bordeaux, France, 2004.