CRT RSA HARDWARE ARCHITECTURE WITH FAULT AND SIMPLE POWER ATTACK COUNTERMEASURES

Apostolos P. Fournaris and Odysseas Koufopavlou

VLSI Design Lab,
Department of Electrical and Computer Engineering
University of Patras, Rio Campus, Greece
email: apofour@ieee.org

ABSTRACT

RSA cryptographic algorithm has long achieved cryptographic and market maturity. However, RSA implementations, after the discovery of Side Channel Attacks (SCA), are susceptible to a variety of different attacks that target the hardware structure rather than the algorithm itself. There are a wide range of countermeasures that can be applied on the RSA structure in order to protect the algorithm from SCAs, however few of them are efficient in hardware since they add extensive performance cost to an SCA resistant RSA implementation. In this paper, a hardware architecture is proposed based on a Fault attack (FA) and Simple Power attack (SPA) resistant algorithm for Chinese Remainder Theorem (CRT) RSA that through the principles of parallelism and component reusability can guarantee hardware efficiency. We describe an implementation approach based on Montgomery modular multiplication and also propose a testing hardware architecture to simulate the security chip environment that our FA-SPA resistant CRT RSA can be integrated in. The designed architecture is implemented in FPGA technology and results on its time and space complexity are extracted and evaluated.

I. INTRODUCTION

Public key cryptographic algorithms due to their high computational complexity are considered difficult to implement, have significant computational time and consume considerable number of hardware resources (chip covered area, power dissipation). So, in order to design an efficient Public key cryptography hardware architecture capable of processing information fast and be able to fit in a relatively resource constrained environment, special attention needs to be paid in the utilized design approach. Furthermore, since nowadays security enhanced devices usually operate in a hostile environment where they can be stolen or manipulated by untrusted users, security attacks on their hardware structure can not be excluded as a potential thread. Such attacks called side channel attacks, exploit an architecture's hardware characteristics leakage (power dissipation, computation time, electromagnetic emission e.t.c) to extract information about the processed data and use them to deduce cryptographic keys and messages [1]. While RSA is considered very secure against traditional cryptanalysis, side channel attacks (SCA) have been successful in determining RSA keys using information leaking from a straightforward implementation of the algorithm. For the above reasons, side channel attack resistance is a highly desirable feature of an RSA encryption/decryption unit architecture but it usually constitutes a performance bottleneck.

SCAs can be mounted very easily using a PC, a simple oscilloscope and some probes and therefore can be used by a wide range of attackers [2], [3], [4]. This ease of use makes SCA very potent. Among the most effective SCA launched on RSA are Fault attacks (FA) and simple power attacks (SPA). The fault attack (FA) goal is to disturb a hardware device during cryptographic operation execution, analyze the faulty behavior of the disturbed device and as a result deduce sensitive information. Combining such attack with a simple power attack (SPA), where a hardware device's power trace is measured and exploited for secret information leakage [1], a crypto-system attacker can relatively easily deduce a cryptographic key of a secure hardware device [5]. Protection of a hardware device against SCAs aims either at the disaccociation of the leaked information with the computed secret data or at minimizing the information leakage itself. To achieve these goals, the existing proposed SCA countermeasures follow two main directives, changes on the cryptographic algorithm and related computer arithmetics or changes on the hardware architecture at hand.

Many researchers have proposed solutions on protecting RSA from FA and SPA with relative success [6], [7], [8], [9]. However, those solutions are focused on FA-SPA resistance on algorithmic level without taking into account the implementation cost for one FA SPA secure RSA encryption/decryption operation. This cost is associated with the arithmetic operations required for RSA and can

978-0-7695-4798-5/12 $26.00 © 2012 IEEE
DOI 10.1109/DSD.2012.38

be very high - restrictive when applying the existing FA SPA resistant RSA solutions in real security devices.

The Chinese Remainder Theorem approach is widely used in the design of an RSA crypto core since it can speed up RSA calculations (modular exponentiation) up to four times compared to the original algorithm. Following this approach, the RSA modulus is divided in two independent sub-modulus that are used for encryption-decryption of the message. The Gauss's combination algorithm is employed in the computation's end in order to reconstruct the correct RSA outcome. The modular exponentiation unit in general, is the main target of FAs and SPAs. The use of CRT, increases RSA vulnerability to such attacks, so strong countermeasures are needed for modular exponentiation protection. SPA resistance is achieved by making the arithmetic operations during the exponentiation algorithm execution undiscriminated from an external observer [8]. Fault attack countermeasures are based on techniques of detecting single fault injection and blocking further processing thus prohibiting the release of secret information. Giraud in [6] proposed a FA- SPA resistant modular exponentiation algorithm and later Kim and Quisquater [9] proposed an attack on the algorithm along with a way to thwart it. Recently, Fournaris [10] proposed a modification of Giraud's and Kim's RSA algorithm [9] that works using Montgomery modular multiplication algorithm and results in very optimistic performance characteristics. The RSA modified algorithms of [10] [6] and [9] guaranty FA resistance by introducing two values, S_0 and S_1 and checking if a known equation between them is always true. If this connection between S_0 and S_1 is disturbed then a fault attack is detected and the cryptographic process is canceled.

In this paper, we proposed a Fault and simple power attack resistant CRT RSA Hardware architecture that uses the CRT RSA scheme proposed by Giraud [6] and further optimized in [10] by adopting Montgomery modular multiplication and exponentiation as its structural element. The proposed scheme uses Montgomery Ladder technique in combination with modulus blinding adapted for Montgomery exponentiation-multiplication so as to offer SPA-FA protection with small compromises in performance and hardware resources. We also propose a testing system-on-chip like architecture that can simulate the security chip environment where the proposed architecture can be integrated and can be used as a verification tool for RSA related protocols as well as a platform for mounting side channel attacks. The efficiency of the proposed architecture is evaluated by realizing it in FPGA technology using VHDL on the testing platform. The resulting implementation provides very interesting results in terms of space and time complexity that are not easily matched even by non SCA resistant designs not bearing the SCA countermeasures extra cost.

The paper is organized as follows. In section II, the algorithms of the proposed architecture are analyzed in detail and their SCA resistance is discussed. In section III, the hardware architecture deriving from the algorithms of section II is proposed and analyzed. The proposed testing platform is discussed in section IV and performance results and comparisons are made in section V. Section VI concludes the paper and presents future goals.

II. EMPLOYED ALGORITHM

Our approach on an SCA resistant RSA module is an extension of the methodology described in [10] and is based on the Montgomery multiplication algorithm as is adapted for Giraud's modular exponentiation methodology resulting in a CRT FA-SPA resistant RSA algorithm. However, the work of [10] only focus on how FA-SPA modular exponentiation can be done and not how a fully functional CRT based FA-SPA unit can operate. In this paper, we adopt the FA countermeasures of [9] as they are enhanced and made more hardware oriented in [10]. Both [10], [9] and [6] are based on the Montgomery Ladder technique [8] for SPA resistance. However, this technique by itself provides protection against the simplest form of SPA and is still vulnerable in more sophisticated attacks like relative doubling attack (RDA) [11] and (N-1) based attack [12]. So, in the proposed scheme, the Montgomery ladder technique SPA resistance is enhanced by multiplying the public modulus with a 32 bit random number r. This enhancement renders RDA and (N-1) based attacks useless against the proposed scheme. The randomness introduced by r is removed in the CRT reconstruction following a modified Garner recombination method [6].

The proposed algorithm employs Montgomery modular multiplication as a structural element, chosen among other approaches due to its efficient realization in hardware. Following the above directive, we adopt the optimized version of the Montgomery modular multiplication algorithm (CSMMM) described in [13], [10], that employs carry-save logic in all its inputs, outputs and intermediate values along with precomputation. The proposed FSPAME algorithm is described below. Note, that a,r are small random numbers (32 bits), e is the exponent, m is the message to be encrypted/decrypted and G is the modulus (p or q in CRT RSA)

FA-SPA Montgomery Modular Exponentiation (FSPAME) algorithm

```
Input: $m, a, r, e = (1, e_{t-2}, ...e_0), G$ where $a$, $r$ : random number
Output: $(a + m^{e-1}) mod(r \cdot G), (a + m^e) mod(r \cdot G)$
Initialization: T = $R^2 mod(r \cdot G)$, $a_R = a \cdot R mod(r \cdot G)$
where $R = 2^{\tilde{n}+2}$, $\tilde{n}$ is the bit length of $(r \cdot G)$.
  1) $S_0 = T \cdot m \cdot R^{-1} mod(r \cdot G)$
  2) $S_1 = S_0 \cdot S_0 \cdot R^{-1} mod(r \cdot G)$
```

```
3) For $i = t - 2$ to $1$
    a) If $e_i = 1$ then
        i) $S_0 = S_0 \cdot S_1 \cdot R^{-1} mod(r \cdot G)$
        ii) $S_1 = S_1^2 \cdot R^{-1} mod(r \cdot G)$
    b) else
        i) $S_1 = S_0 \cdot S_1 \cdot R^{-1} mod(r \cdot G)$
        ii) $S_0 = S_0^2 \cdot R^{-1} mod(r \cdot G)$
4) $S_1 = (S_0 \cdot S_1) \cdot R^{-1} mod(r \cdot G)$
5) $S_0 = (S_0^2) \cdot R^{-1} mod(r \cdot G)$
6) $S_1 = (S_1 \cdot 1 + a_R) R^{-1} mod(r \cdot G)$
7) $S_0 = (S_0 \cdot 1 + a_R) R^{-1} mod(r \cdot G)$
8) If (Loop Counter $i$, exponent $e$ are not modified then
   return $(S_0, S_1)$ else return error
```

The above algorithm constitute the core of the CRT RSA module. Assuming that $N = p \cdot q$ is the RSA modulus described as the product of two large secret prime numbers p, q and that s be the public RSA exponent while d be the private RSA exponent satisfying the equation $s \cdot d \equiv (1) mod(p-1) \cdot (q-1)$, we can denote as d_p the value $d_p = d mod(p-1)$ and as d_q the value $d_q = mod(q-1)$ acting as partial private keys of CRT mode. Using the above definitions and the FSPAME algorithm, we can describe the CRT FA-SPA RSA algorithm as follows:

FA-SPA CRT RSA algorithm

```
Input: $m, a, p, q, d_p, d_q, i_q = q^{-1} mod p, N$
Output: $m^d mod N$
1) Generate random numbers $r$ and $a$
2) $(s_0^p, s_1^p) = FSPAME(m, a, d_p, p, r)$
3) $(s_0^q, s_1^q) = FSPAME(m, a, d_q, q, r)$
4) $\acute{S} = s_0^q + q \cdot ((s_0^p - s_0^q) mod(r \cdot p) \cdot i_q mod(r \cdot p)$
5) $S = s_1^q + q \cdot ((s_1^p - s_1^q) mod(r \cdot p) \cdot i_q mod(r \cdot p))$
6) $\acute{S} = (m \cdot \acute{S} + a) mod N$
7) $S = (S + a \cdot m) mod N$
8) If $(S = \acute{S})$ and p, q not modified then return
   $(S - a - a \cdot m) mod N$ else return error
```

In the CRT RSA crypto process two executions of the FSPAME algorithm are performed. The algorithm is run using d_p and d_q exponents and p, q modulus respectively. We can assume without loss of generality that p, q are of similar bit length and therefore about half the bit length of N. Also, we can assume that in each execution of the FSPAME algorithm, $R = R_p = 2^{\widetilde{n}/2+2}$ or $R_q = 2^{\widetilde{n}/2+2}$ since $\widetilde{n}/2$ is the bit length of $r \cdot p$, $r \cdot q$. Thus, using CRT, the tedious n bit modular exponentiation operation is broken into two $\widetilde{n}/2$ ($(n+32)/2$ at most) bit modular exponentiations that can provide results faster and require less hardware resources. The price for this action, is a final CRT operation (steps 4 and 5) for both fault secure streams of data. Note, that the use of the random number a prohibits the discovery of the RSA result before the fault attack check is performed (step 8).

III. PROPOSED HARDWARE ARCHITECTURES

Designing an FA-SPA resistant RSA encryption/decryption poses several challenges especially if the system at hand is resource constrained. Our primary goal is to minimize the chip covered area without compromises in security. For this reason, we employ one FSPAME unit responsible for modular exponentiation using either p or q. The FSPAME unit is structured around two Montgomery Modular Multiplier units using Carry-Save logic input (Carry Save Montgomery Multiplier Unit, CSMMU) and output signals as described in [13], [10]. The FSPAME unit's CSMMUs are connected through a series of input-output storage elements (Register) that are specially designed to feed those units with appropriate input data. Each Montgomery multiplier is a 5 parallel ($\widetilde{n}/2 + 1$) bit and 2 serial (1) bit input architecture that generates a Montgomery modular multiplication product in Carry-Save format every $\widetilde{n}/2 + 2$ clock cycles. The functionality and full architecture of the CSMMU is described in [13], [10]. The architecture takes advantage of the inherent parallelism in the FSPAME algorithm so as to minimize the needed hardware components for the CRT RSA operations and maximize the reuse of the existing components (mainly the CSMMUs). More specifically, in FSPAME algorithm, steps 3ai and 3aii, steps 3bi and 3bii, steps 4 and 5 as well as steps 6 and 7 can be run in parallel. So, the two CSMMUs apart from the initialization FSPAME steps (step 1 and 2) operate in parallel during one exponentiation and have distinct roles. The first Montgomery Multiplier unit (CSMMU1) always perform Montgomery multiplication between the previous FSPAME round's two multiplication results while the second Montgomery Multiplier unit (CSMMU2) performs always a squaring operation of its previous round's squaring result or the multiplication results of the first Montgomery multiplier. The proposed FSPAME based CRT RSA architecture is presented in Figure 1.

We have successfully managed to reuse the two CSMMUs in the CRT reconstruction functions of the CRT RSA algorithm (steps 4 and 5) utilizing the existing Montgomery multipliers for all necessary modulo p operations. The parallelism and reusability principle is retained in CRT reconstruction since Steps 4 and 5 inputs are propagated through CS subtractor units back into the CSMMUs for CRT modulo $(r \cdot p)$ calculations in parallel. Parallelism is also used for the multiplication of the modulo p results with q. As these results, for steps 4 and 5, reach the output at the same time, two parallel Carry-Save serial multipliers are used for q multiplication followed by Carry Save Full adders in order to perform addition with s_0^q, s_1^q in Carry Save format accordingly.

The FSPAME unit needs to be used twice for one CRT RSA session since it implements steps 1 and 2 of the

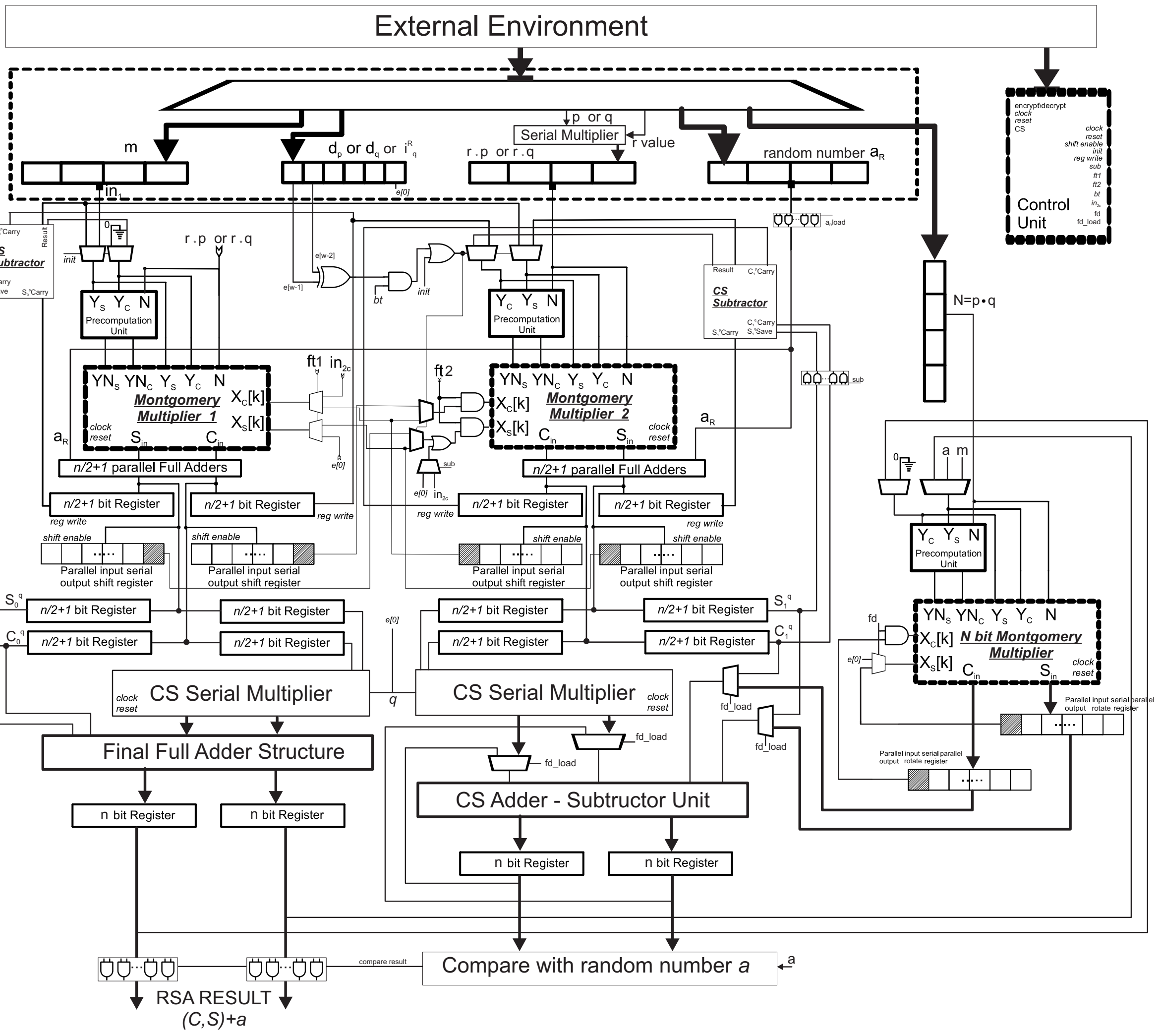

Fig. 1. The CRT FA-SPA resistant RSA architecture

CRT RSA in a serial fashion. During the first use of the FSPAME unit the outcome of $m^{d_q} mod(r{\cdot}q)$ is calculated in carry - save format and stored in a intermediate register set while on the second use of the FSPAME unit the outcome $m^{d_p} mod(r \cdot p)$ is calculated and stored in Montgomery Multiplier 1 output registers. At this point, through the sub control signal, the subtractor is put into effect (before CRT reconstruction, it performs subtraction with zero) and value $S_0^p - S_0^q$ is inserted in Montgomery Multiplier 1 while the value $S_1^p - S_1^q$ is inserted in Montgomery multiplier 2. After one Montgomery multiplication in each CSMMU with $i_q^{R_p} = q^{-1} R_p modp$ where $R_p = 2^{n/2+2}$, the values $(S_0^p - S_0^q) i_q modp$ and $(S_1^p - S_1^q) i_q modp$ are stored in a register set and are ready to be inserted in the CS serial multipliers for multiplication with q. The output of the CRT transformation unit is an n-bit value in Carry Save format stored in two n-bit registers.

After the completion of CRT transformation, the 2 outcomes are examined for fault attack, as described in section II and the CRT RSA algorithm steps 5 and 6. The outcome of this action is either an error code provided by an all zero value or the RSA ciphertext message in Carry - Save (CS) format added with the random number a. To achieve this we use an additional n bit Montgomery Multiplier unit that is set up to interact with the existing CS Adder unit. To further this action we enhance the CS adder unit of Montgomery Multiplier 2 with subtraction capabilities (it thus becomes a CS adder-Subtractor Unit) so as to realize the full Fault Detection functionality. More specifically, the extra Montgomery multiplier unit (n bit Montgomery Multiplier) and CS adder-subtractor unit performs serially the following operations, where $R = 2^{n+2}$:

1) $m_R = m \cdot T \cdot R^{-1} modN$
2) $a_m = (a \cdot m_R) \cdot R^{-1} modN = a \cdot m \cdot modN$

3) $S = S - a_m$
4) $\bar{S} = \acute{S} \cdot m_R \cdot R^{-1} modN = m \cdot \acute{S} modN$
5) $comp = \bar{S} - S$

The value $comp$ is inserted into a comparator unit that tests if it is equal to the random number a. A true answer releases the correct result that is stored in the $n - bit$ registers following the data path of Montgomery Multiplier 1. This result is equal to $\acute{S} + a$.

All modern processing systems have certain restrictions in the information that can be process within their respected data core. These limitations are posed by their bus length, memory size and depth e.t.c. No modern processor can handle data with bit length closely relevant to that of the RSA core (at least 1024 bit). All modern systems work with at most 64 bit numbers. For this reason, we develop a mechanism for bus data insertion into the RSA architecture so that the arbitrary input RSA algorithm where the bit length of the processed values is higher than the one provided by an external system's data bus, can fully work. The data bus inputs are multiplexed using the input reconstruction mechanism. There exist 3 digit input parallel output shift registers (for the message m, the random number a_R, a and the $r \cdot p$ or $r \cdot q$ modulus), one digit input serial output shift register (for the d_p or d_q exponent). The data bus is accessed in digits analogous to a connected control processor word length ($wbits$) and information are stored to the appropriate registers through a multiplexer structure. During the initialization of the RSA values, a $w \times 32$ digit serial multiplier is used in order to calculate the $r \cdot p$ or $r \cdot q$ values in digits (assuming that the bit length of random number b is 32).

IV. CRT RSA TESTING ARCHITECTURE

In order to evaluate the correct functionality of the proposed architecture in pragmatic conditions, it is of value the development of a testing architecture that can accurately simulate the system where the CRT RSA Design can be placed. Modern security chips, like smart cards, TPM or sensors as well as special purpose token chips [14] can be a potential integration target of the proposed design. Such chips, consist of a security processor, input/output interface as well as non-volatile memory for security data storage [15]. To simulate such system, we designed a testing architecture for FPGA technology, described in Figure 2, that is based on the Picoblaze soft core microprocessor provided by Xilinx. The proposed testing architecture includes UART serial I/O communication to the external world, a non volatile memory module (NAND Strata Flash memory), the developed CRT RSA crypto core and a control Picoblaze processor.

As it can be observed from Figure 2 the Picoblaze processor is connected to all the system's component and is responsible for storing data that are read from the UART interface into memory as well as sending data through the UART interface to the external processing environment. The processor is also responsible for initiating the RSA cryptographic operations through appropriate control signals (crypto control). When CRT RSA crypto core is enabled, then the processor receives control-status messages from the core but it plays no direct role in the cryptographic operations since the data, address bus to the flash memory at this point is controlled by the CRT RSA crypto core. The Picoblaze processor regains control of the system when ciphertext computations in the crypto-core are concluded. The processor is responsible for storing those results in the Flash memory for further use.

The goal of the above system's functionality is to maximize the simplicity and minimize the interference of the processor in the proposed CRT RSA unit's cryptographic operation. As a result, the testing system has small chip covered area and does not downgrade the overall performance of the RSA crypto core that is our testing target.

V. PERFORMANCE

Assuming that n bit RSA encryption/decryption needs to be performed, data insertion to the RSA unit's input registers (modulus, random number, exponent and message registers) can be done with small speed-throughput overhead to the whole system. Thus, ignoring the cost of data insertion, that is closely relevant to the input external device speed (memory write speed, bus read speed), the proposed CRT RSA architecture requires $C_{CRT_{RSA}} = C_{FSPAME} + C_{CRT} + C_{FA}$ clock cycles for one encryption/decryption operation where C_{FSPAME} is the clock cycle number for Modular exponentiation using the FSPAME algorithm, C_{CRT} is the clock cycle number for CRT transformation and C_{FA} is the clock cycle number for the Fault detection. The $C_{CRT_{RSA}}$ is dominated by the FSPAME unit since C_{FSPAME} has $O(n\dot{t})$ time complexity (as is described below) where t is the Hamming weight of the exponent. In CRT RSA decryption, we assume that d_p and d_q are of similar bit length, i.e. $n/2$, thus $t = n/2$ (usual case for CRT RSA) and C_{FSPAME} has $O(\frac{n^2}{2})$ time complexity. The C_{CRT} and C_{FA} numbers are of $O(n/2)$ and $O(n)$ complexity, respectively.

The Montgomery multiplication unit needs $\frac{\widetilde{n}}{2} + 2$ clock cycles to come up with a result. The FSPAME algorithm is concluded after $t - 2$ algorithmic rounds where the CS Montgomery Multiplication algorithm is executed in parallel 2 times in each round. Also, in FSPAME, 2 serial executions of Montgomery multiplication are needed for initialization (FSPAME steps 1, 2) and 2 parallel Montgomery executions for post processing calculations. As a result, the total number of clock cycles for one modular exponentiation using FSPAME is $(t+2) \cdot (\frac{\widetilde{n}}{2} + 2)$. Since through the CRT RSA algorithm (steps 1 and 2) two FSPAME executions are performed, all modular exponentiations are concluded after $(t+2) \cdot (\widetilde{n} + 4)$ clock cycles.

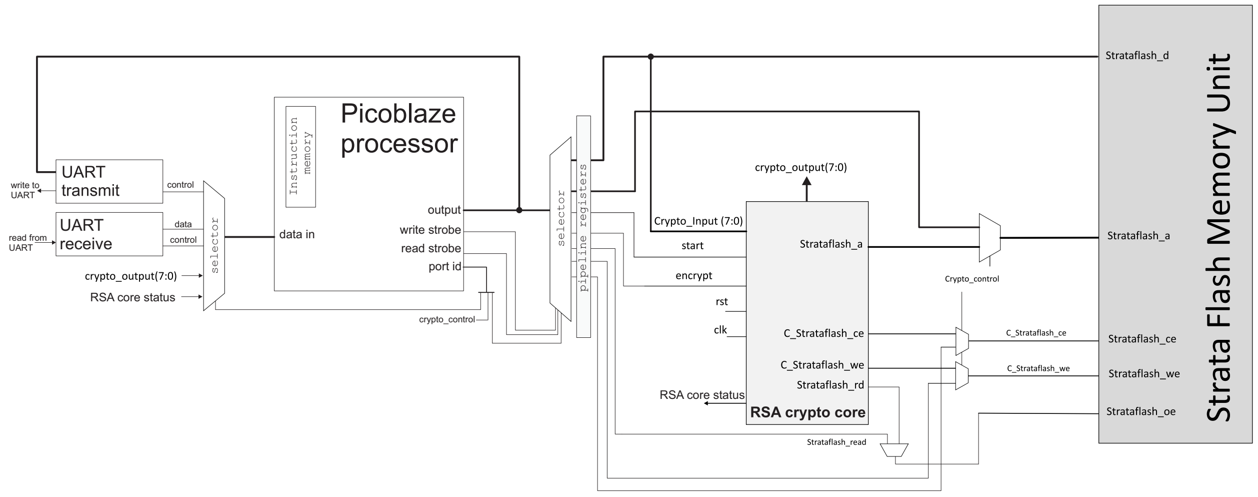

Fig. 2. Proposed Security chip environment for testing the CRT FA-SPA resistant RSA architecture

If $t = \frac{n}{2}$ then $C_{FSPAME} = (\frac{n}{2} + 2)(n + r_b + 4)$ where r_b is the bit length of the random number r.

The delay introduced by the CRT recombination and Fault detection operations is considerably smaller than the FSPAME delay . More specifically, the parallel bit serial CS multipliers employed for CRT recombination require $n/2$ clock cycles to come up with a result and they are followed by CS Final Full Adder units performing addition with s_0^q, s_1^q values in 1 clock cycle. Thus the total delay for CRT recombination is $C_{CRT} = n/2 + 1$. Similarly, fault detection operations, as described in section III, are performed in parallel to FSPAME and CRT recombination (steps 1, 2, 3). The delay overhead of Fault detection comes from section's III step 4 and 5 where a n-bit Montgomery multiplication is performed (requires $n + 2$ clock cycles) followed by a serial comparator unit operation (requires n clock cycles). Thus the total delay of Fault detection us $C_{FA} = 2 \cdot n + 2$ clock cycles. Taking into account the delay cost of a final result transformation from C-S format to non C-S format (using a serial Adder requiring n clock cycles) the total clock cycle delay for one CRT RSA operation is $C_{CRT_{RSA}} = C_{FSPAME} + C_{CRT} + C_{FA} + n =$ $(\frac{n}{2} + 2)(n + r_b + 4) + \frac{n}{2} + 1 + 2 \cdot n + 2 + n =$ $\frac{n^2}{2} + (r_b + 15) \cdot \frac{n}{2} + 2 \cdot r_b + 11$.

To further evaluate our proposed system we realized the CRT RSA architecture in FPGA technology (Xilinx Virtex 5) using xilinx ISE 13.2 tools, for $n = 1024$ RSA operations, integrated it to the proposed testing platform and performed real verification tests. Assuming that p and q are approximately $n/2$ bit length, a 512 bit FSPAME architecture, needed for 1024 bit CRT RSA, was synthesized, placed and rooted on the FPGA board while measurements in chip covered Area (FPGA slices), maximum clock frequency (MHz), decryption delay (msec) and Throughput (bps) were taken. In this implementation, the CRT reconstruction unit is also integrated along with the Fault Detection n-bit Montgomery Multiplier, that are described in Figure 1. We have also included into our design the full functionality of the control unit that handles both RSA control and input data reconstruction. The synthesis results of the proposed design in comparison with similar works are presented in Table I. For compatibility reasons, the proposed architecture was also implemented on the outdated (not supported anymore by the latest Xilinx ISE tools) Xilinx Virtex 2 FPGA technology (XC2V6000) in order to provide more fair comparison results with existing RSA implementations.

Table I. RSA Implementation Results for $n = 1024$

Arch.	Techn.	Area	Freq.	FA-SPA	CRT	delay	Throughput
prop.	XC5VLX110T	9242	219	Yes	Yes	2.6 ms	392 Kbps
prop.	XC2V6000	26523	107	Yes	Yes	5.35 ms	179 Kbps
[16]	XC2V3000	12537	152.5	No	No	13.8 ms	74.3 Kbps
[17]	XC2V6000	24767	100.5	No	Yes	2.63 ms	388.7 Kbps
[18]	XCV1000E	22712	14,55	Yes	No	27,79 ms	18.43 Kbps
[19]	XC2V6000	25193	84.33	Yes	No	18.7 ms	54.72 Kbps

As can be observed from Table I, the proposed architecture in Xilinx virtex 2 technology has similar Area and Frequency results with the CRT design of [17] but is slower. Similarly, when compared with the non CRT implementation of [16], our work requires more area resources and has smaller operating frequency but achieves considerably higher throughput and small delay. Note, however, that both [17], [16] do not have FA-SPA countermeasures and thus do not bare any hardware protection overhead. When comparing the proposed work with the SPA (but not FA) resistant designs of [18] and [19], we observe that the proposed implementation is considerably faster using approximately the same number of FPGA slices. When using the superior xilinx virtex 5 (instead of virtex 2) FPGA technology then the proposed implementation, as expected, has better results that all other compared works. From the comparisons, it can be concluded that the proposed architecture along with its FA-SPA countermeasures constitutes a very efficient RSA hardware protection approach that leads to small overheads when compared to other existing protected RSA architectures thus offering performance and utilized hardware resources characteristics that are close to unprotected designs.

VI. Conclusion

In this paper we introduced a CRT RSA hardware architecture based on Montgomery Modular exponentiation and Multiplication that using the principles of parallelism and component reusability can offer high efficiency in space and time complexity as well as Fault attack and simple power attack protection. To test the architecture, a testing platform was also designed in order to simulate the security chip environment where our proposed architecture can be integrated. To further proof the validity of our case, we implemented this architecture in FPGA technology and came up with very interesting results. The proposed approach towards a FA-SPA resistant RSA cryptosystem is a step towards the realization of a fully protected hardware structure against most FA-SCAs.

VII. References

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology Proceedings of Crypto 99*. Springer-Verlag, 1999, pp. 388–397.

[2] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[3] N. Sklavos and X. Zhang, *Wireless Security and Cryptography: Specifications and Implementations*. Boca Raton, FL, USA: CRC Press, Inc., 2007.

[4] A. Bechtsoudis and N. Sklavos, "Side channel attacks cryptanalysis against block ciphers based on fpga devices," *VLSI, IEEE Computer Society Annual Symposium on*, vol. 0, pp. 460–461, 2010.

[5] F. Amiel, K. Villegas, B. Feix, and L. Marcel, "Passive and active combined attacks: Combining fault attacks and side channel analysis," in *Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography*, ser. FDTC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 92–102.

[6] C. Giraud, "An rsa implementation resistant to fault attacks and to simple power analysis," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1116–1120, 2006.

[7] D. Vigilant, "Rsa with crt: A new cost-effective solution to thwart fault attacks," in *CHES*, ser. Lecture Notes in Computer Science, E. Oswald and P. Rohatgi, Eds., vol. 5154. Springer, 2008, pp. 130–145.

[8] M. Joye and S.-M. Yen, "The montgomery powering ladder," in *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*. London, UK: Springer-Verlag, 2003, pp. 291–302.

[9] C. H. Kim and J.-J. Quisquater, "Fault attacks for crt based rsa: New attacks, new results, and new countermeasures," in *WISTP*, ser. Lecture Notes in Computer Science, D. Sauveron, C. Markantonakis, A. Bilas, and J.-J. Quisquater, Eds., vol. 4462. Springer, 2007, pp. 215–228.

[10] A. P. Fournaris, "Fault and simple power attack resistant rsa using montgomery modular multiplication," in *Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS 2010)*. IEEE, 30 May-02 June 2010.

[11] S. Yen, L. Ko, S. Moon, and J. Ha, "Relative doubling attack against Montgomery Ladder," *Information Security and Cryptology-ICISC 2005*, pp. 117–128, 2006.

[12] S.-M. Yen, W.-C. Lien, S.-J. Moon, and J. Ha, "Power analysis by exploiting chosen message and internal collisions - vulnerability of checking mechanism for rsa-decryption," in *Mycrypt*, ser. Lecture Notes in Computer Science, E. Dawson and S. Vaudenay, Eds., vol. 3715. Springer, 2005, pp. 183–195.

[13] A. P. Fournaris and O. G. Koufopavlou, "A new rsa encryption architecture and hardware implementation based on optimized montgomery multiplication," in *ISCAS (5)*. IEEE, 2005, pp. 4645–4648.

[14] D. Hein and R. Toegl, "An autonomous attestation token to secure mobile agents in disaster response," in *The First International ICST Conference on Security and Privacy in Mobile Information and Communication Systems (MobiSec 2009)*. Torino, 2009.

[15] A. P. Fournaris, "Trust ensuring crisis management hardware module." *Information Security Journal: A Global Perspective*.

[16] M.-D. Shieh, J.-H. Chen, H.-H. Wu, and W.-C. Lin, "A new modular exponentiation architecture for efficient design of rsa cryptosystem," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 9, pp. 1151–1161, 2008.

[17] C. McIvor, M. McLoone, and J. McCanny, "Fast montgomery modular multiplication and rsa cryptographic processor architectures," in *Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on*, vol. 1, nov. 2003, pp. 379 – 384 Vol.1.

[18] K. A. Bayam and B. Ors, "Differential Power Analysis resistant hardware implementation of the RSA cryptosystem," *2008 IEEE International Symposium on Circuits and Systems*, vol. 18, no. 1, pp. 3314–3317, 2008.

[19] ——, "Differential power analysis resistant hardware implementation of the RSA cryptosystem," *Turkish Journal of Electrical Engineering Computer Sciences*, vol. 18, no. 1, pp. 129–140, 2010.