Crypto-Test-Lab for Security Validation of ECC Co-processor Test Infrastructure

Emili Lupon, Rosa Rodríguez-Montañés and Salvador Manich
Universitat Politècnica de Catalunya-BarcelonaTech
Avda. Diagonal, 647Barcelona, Spain
{emili.lupon, rosa.rodriguez, salvador.manich}@upc.edu

***Abstract*—Elliptic Curve Cryptography (ECC) is a technology for public-key cryptography that is becoming increasingly popular because it provides greater speed and implementation compactness than other public-key technologies. Calculations, however, may not be executed by software, since it would be so time consuming, thus an ECC co-processor is commonly included to accelerate the speed.**

Test infrastructure in crypto co-processors is often avoided because it poses serious security holes against adversaries. However, ECC co-processors include complex modules for which only functional test methodologies are unsuitable, because they would take an unacceptably long time during the production test. Therefore, some internal test infrastructure is always included to permit the application of structural test techniques.

Designing a secure test infrastructure is quite a complex task that relies on the designer's experience and on trial & error iterations over a series of different types of attacks. Most of the severe attacks cannot be simulated because of the demanding computational effort and the lack of proper attack models. Therefore, prototypes are prepared using FPGAs. In this paper, a Crypto-Test-Lab is presented that includes an ECC co-processor with flexible test infrastructure. Its purpose is to facilitate the design and validation of secure strategies for testing in this type of co-processor.

I. Introduction

Since the publication of the public-key cryptography based on ECC in 1985 independently by V. S. Miller and N. Koblitz [1], its adoption has gradually grown. It presents advantages with respect to other technologies like RSA [2], requiring shorter key lengths for the same degree of security. For example, to achieve the same degree of security as a symmetric encryption such as AES (128 bits) [3], the key length for RSA is 3072 bits and for ECC is just 256-283 bits [4].

One of the most extended infrastructures for production testing are scan-paths, which have been the most extended standardized technology for many years now [5]. Nevertheless, from the point of view of security, scan-paths are a nightmare [6]. It has been recognized that test infrastructures like scan-paths [5] can be exploited by adversaries to retrieve sensitive information of security chips [7]. In [5] an attack was conceived to break the Data Encryption Standard (DES) [8] implementation. The attack first identified the positions of the intermediate registers in the scan-paths and then retrieved the DES first round key by applying only three chosen plain-texts. In [9] an attack on ECC cryptosystem was presented that first identified the position of the internal registers affected by the elliptic scalar multiplication, working in test mode, and later formed the attack by shifting-out the content of these registers through scan-paths.

978-1-5386-5108-7/17/$31.00

To avoid these problems, accesses are usually destroyed physically or logically by some means after production testing, so that they cannot be used anymore [10]. However, scan-paths are not only an efficient technology for production testing, but also for diagnosis in after-market maintenance and, as a consequence, locks are not considered an interesting solution to this problem.

In the last decade several strategies have been proposed to improve the security of scan-paths. The motivation is essentially to extend its operation in the field for diagnosis and maintenance without seriously compromising the security [11]. Amongst many of the ideas that are applied, most of them are not really secure but obscure, which means that once the trick is discovered, it can be bypassed .

How to find the degree of security achieved by a new trick is a very difficult question. Adversaries are continuously improving their techniques to attack security chips and, what is worse, is that they combine different techniques to improve the penetration efficiency. Side-channel techniques are commonly used whereby parameters such as the power consumption or electromagnetic radiation are monitored to trace the point of computation where the system is. Furthermore, once a point of interest is detected, failures can be induced and/or data can be extracted, e.g. using the scan-path.

One of the most dangerous points is time. Adversaries have plenty of time, in particular when the trade-off time/revenue is cost-effective. Over time, millions of attack iterations can be executed.

At the design phase, a particular technique is selected. The assessment of the degree of security, e.g. in scan-paths, cannot be supported by simulation because of the low speed, and the lack of adequate physical modeling of power or electromagnetic emissions. Consequently, the only way to conduct a comprehensive security analysis is to develop prototypes, usually on FPGAs, which provide enough speed and a close emulation of the electrical/electromagnetic behavior of a real device [12].

In this paper, a platform Crypto-Test-Lab (CTL) is presented whose purpose is to offer flexibility for the prototyping and evaluation of security strategies applied in test infrastructures

oriented to ECC co-processors. CTL includes a fully functional ECC co-processor with scan-paths. An attack manager communicates with a PC that allows it to control and emulate attacks on the platform. The speed is significantly high, so millions of iterations can be carried out in a reasonable amount of time.

The rest of the paper is structured as follows. In section (II), a short description of the ECC cryptography is presented. In section (III), the functional description and the implementation of the ECC-core including an ECC co-processor is explained. In section (IV), the implementation and the Crypto-Test-Lab infrastructure is detailed. In section (V), the results of synthesis and performance of the FPGA platform are summarized. Finally, conclusions are discussed in section (VI).

II. Elliptic Curve Cryptography

As described in [4], an *elliptic curve* over a field $\mathcal{F}$ is a non-singular, plane curve of two variables x and y, with values belonging to field $\mathcal{F}$ that satisfies a cubic equation $f(x,y)=0$. Non-singular means that the curve has neither double or triple roots for $y=0$. The field $\mathcal{F}$ may be any field, as the real number field $\mathbb{R}$, any finite field $\mathbb{Z}_p$ or extended finite field $\mathrm{GF}(p^m)$, where $p \geq 2$ is a prime number and $m \geq 2$.

The set composed by all points $P=(x,y)$ of an *elliptic curve* over a field plus a point 0 in the infinity, together with an appropriate *addition operation* between points in the set, constitute an *Abelian cyclic group* $\mathcal{C}_n$ with order n (the order is the number of points in the group). Each point P in the group generates, by iterative addition, a cyclic subgroup $\mathcal{H}_P \subseteq \mathcal{C}_n$. P is named the *generator* of subgroup $\mathcal{H}_P$.

In ECC, elliptic curves over finite fields delivering finite *Abelian cyclic groups* $\mathcal{C}_n$ are selected. According to the recommendations of the National Institute of Standards and Technology [13], there are several finite fields, *elliptic curves* and *generators* of subgroups available for ECC cryptography. In CTL, the Klobitz *elliptic curve* $y^2+xy=x^3+x^2+1$ over $\mathrm{GF}(2^{163})$ with primitive polynomial $p(\alpha)=\alpha^{163}+\alpha^7+\alpha^6+\alpha^3+1$ is selected; any operation $\circ$ inside GF uses the regular modulus ($\circ \mod p(\alpha)$). Also, the cyclic subgroup $\mathcal{H}_P$ generated by point $P=(x,y)$ with $x = 2$ fe13c053 7bbc11ac aa07d793 de4e6d5e 5c94eee8 and $y = 2$ 89070fb0 5d38ff58 321f2e80 0536d538 ccdaa3d9 in hexadecimal notation is selected. Point $P=(0,1)$ does not belong to this subgroup, and its order is a prime number a bit higher than $2^{162} \simeq 0.5846\ 10^{49}$.

The most crucial operation for the encryption is the *addition* between points P in the *cyclic group* $\mathcal{C}_n$, which is defined as follows and identified with the symbol $\dot{+}$ (dot over plus sign) henceforth:

1) The point 0 in the infinity acts as the neutral or identity element. Therefore, it is the inverse of itself.
2) The inverse of any other point P is the alternative point having the same x. If there is no other point then, P is the inverse of itself.
3) The *addition* of three aligned points is the point 0 at the infinity. Therefore, the addition of two non-inverse points is the inverse of the third point aligned with them.

In CTL in particular, the point 0 in the infinity is not used in additions, since two inverse points will never be added. This simplifies calculations. The inverse of a point $P=(x,y)$ is calculated as $\dot{-}P=(x,x+y)$, and the *addition* $P_3=P_1\dot{+}P_2$ is a complex operation that needs to follow the steps described in the following paragraphs.

If $x_1 \neq x_2$, we have pure *addition*, while if $x_1 = x_2$ (necessarily $y_1=y_2$ and $P_2=P_1$), we will speak of *doubling*.

Step 1 The slope λ of the line passing through P_1 and P_2 is calculated. If points are different, $\lambda=(y_1+y_2)/(x_1+x_2)$ with $x_1+x_2\neq 0$. If points are the same, $\lambda=x_1+y_1/x_1$ with $x_1\neq 0$. Notice that, here and in the next steps, the addition and division are the regular operations in the finite field GF.

Step 2 The square λ^2 is calculated.

Step 3 The x-coordinate of P_3 is calculated: $x_3=\lambda^2+\lambda+x_1+x_2+1$. For *doubling* $x_3=\lambda^2+\lambda+1$.

Step 4 The y-coordinate of P_3 is calculated: $y_3=x_3+\lambda(x_3+x_1)+y_1$.

A. Public and private keys

The central operation for the public key cryptography is the scalar product ($\dot{\times}$, dot over times sign) that is defined as,

$$Q = k \dot{\times} P = [P \dot{+} P \dot{+} \ldots \dot{+} P]_{k \text{ times}} \tag{1}$$

where the subgroup $\mathcal{H}_P$ is of very high order. Operation (1) is still quite easy to be computed, while the reverse operation becomes extremely difficult. P is the *generator point* of the subgroup and is obtained from a table of precomputed numbers specified in the standard, k is the private key that is usually generated at random, and Q is the public key.

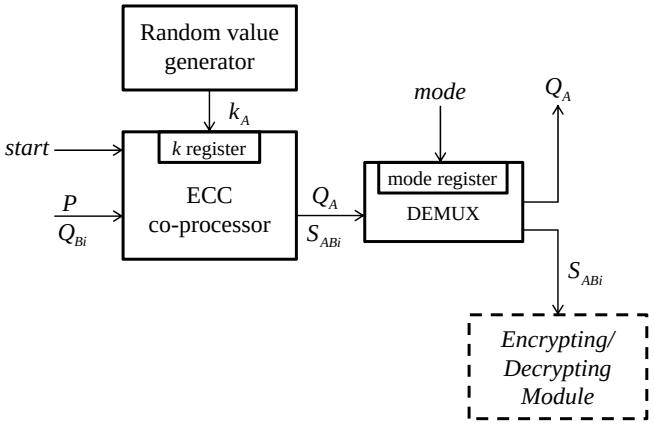
B. Public key cryptography

Consider two people, **A** and **B**, passing encrypted information using a common secret key S_{AB} but without transmitting it. Through an insecure channel, they agree on using a subgroup $\mathcal{H}_P$ and a *generator point* P. **A** selects a private key k_A and generates the public key $Q_A=k_A\dot{\times}P$. A sends Q_A to **B** through the same channel. **B** does the same with k_B and $Q_B=k_B\dot{\times}P$. Finally, **A** calculates the common secret key from his private key and the public key of **B**, $S_{AB}=k_A\dot{\times}Q_B=(k_A\times k_B)\dot{\times}P$, and similarly does **B**, producing the same common secret key $S_{AB}=k_B\dot{\times}Q_A=(k_B\times k_A)\dot{\times}P$. Then, using symmetric encryption, they can transmit information through a secure channel. The crucial point is that knowing P, Q_A and Q_B is extremely difficult to discover the private keys k_A and k_B and, therefore, to discover S_{AB}.

III. ECC Core

The ECC core is a module based on an ECC co-processor whose purpose is to perform all the actions related with ECC data encryption and decryption. The full diagram of the ECC core module is presented in Fig. 1. It includes the following blocks:

Figure 1: Block diagram of the ECC core hardware.



A random value generator. It is based on an array of 8 independent self-clock LFSRs of 22 bits each producing a 161 bit word.

An ECC co-processor. It performs the scalar product of a private key k_A by a point P or by a public key Q_{Bi} after input $start$ becomes active. It produces a public key Q_A or a common secret key S_{ABi}, respectively.

Mode selector DEMUX. Under the control of signal $mode$, it returns the output of the ECC co-processor (when the public key is generated) or keeps the common secret key internally for the encrypting/decrypting module.

To begin a new scalar product, the ECC co-processor has two control inputs, $mode$ (registered) and $start$. For $mode = 0$, a new random value is loaded in k-register $= k_A$ when $start$ is asserted and a scalar product is executed taking input point P (shared with the communicator counterpart) and the public key Q_A is returned. After receiving the public key of communicator **B** (Q_{Bi}), the common private key is generated by selecting $mode = 1$. In this mode, the same private k_A stored in the k-register is used. When $start$ is asserted, the scalar product is executed and the common secret key S_{ABi} is obtained. Now, the product is not output, but sent internally to the encrypting/decrypting module running a symmetric encryption algorithm. In this second mode, the common secret key is hidden.

A. ECC co-processor

The ECC co-processor performs two sets of operations. On the one side, the regular arithmetic addition, product, squaring and division in the finite field $\mathrm{GF}(2^{163})$ with primitive polynomial $p(\alpha)$ and, on the other side, the elliptic curve arithmetic *addition* ($\dot{+}$) and scalar multiplication ($\dot{\times}$) in the cyclic subgroup $\mathcal{H}_P$ over the points of the Koblitz elliptic curve.

B. Regular arithmetic

Regular arithmetic is done in the finite field $\mathrm{GF}(2^{163})$. Addition and squaring (λ^2) are performed just using XOR gates.

Figure 2: Euclidean division algorithm.

```
1: function DIVISION(a(α), b(α))
2:     R_A ← a(α)
3:     R_B ← b(α)
4:     M_A ← 0
5:     M_B ← p(α)
6:     while R_B > 1 do
7:         if lsb(R_B) = 0 then
8:             R_A ← R_A/α
9:             R_B ← R_B/α
10:        else
11:            R_A ← (R_A + M_A)/α
12:            R_B ← (R_B + M_B)/α
13:        end if
14:        if (lsb(R_B) = 1)&&(deg(R_B) < deg(M_B)) then
15:            M_A ← R_A
16:            M_B ← R_B
17:        end if
18:    end while
19:    return R_A
20: end function
```

The product $\lambda(x_3+x_1)$ is implemented using a shift-and-add algorithm proceeding from the msb to the lsb. After the left shifting of the accumulated value, the primitive polynomial $p(\alpha)$ is used for the modulus operation to obtain the new partial product. The operation can be completed in 164 clock cycles.

The division $(y_1 + y_2)/(x_1 + x_2)$ or y_1/x_1 is implemented using the Euclidean division algorithm , which considers that the quotient of a polynomial $d(\alpha)$ by α is a right-shift of $d(\alpha)$ if the lsb is 0 or a right-shift of $d(\alpha) + p(\alpha)$ if the lsb is 1.

In the Euclidean division algorithm, the quotient $q(\alpha)$ of $a(\alpha)/b(\alpha)$ is carried out using four registers that are cyclically updated. These, namely R_A, R_B, M_A and M_B, are initialized with $a(\alpha)$, $b(\alpha)$, 0 and $p(\alpha)$ respectively. After each cycle in the algorithm, the two following conditions are satisfied: 1) the lsb of M_B is 1 and 2)

$$q(\alpha) = \frac{R_A}{R_B} = \frac{M_A}{M_B} = \frac{R_A + M_A}{R_B + M_B}$$

The pseudo-code of the algorithm is shown in Fig. 2 and it needs a maximum of 327 clock cycles to complete. After this, $R_B = 1$ and, therefore, $q(\alpha) = R_A$.

C. Elliptic curve arithmetic

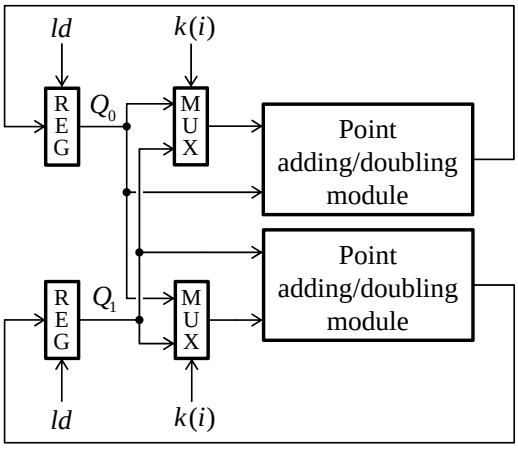
Addition ($\dot{+}$) is implemented following the steps described in Section II and the scalar product ($\dot{\times}$) is explained henceforth.

In security applications, the Montgomery ladder algorithm [14] is usually used because of the low amount of information leaked due to power consumption or electromagnetic radiation. Despite the ones or zeros processed in the private key, in each cycle it performs the same operations and, therefore, a power

Figure 3: The parallel Montgomery ladder algorithm for the scalar product. (*) $Q_1 \dot{+} Q_0$ is always calculated, but Q_0 is not updated for $i = 161$.

function SCALARPRODUCT(k, P)
  $Q_0 \leftarrow P$; $Q_1 \leftarrow P$
  for $i = 161$ **downto** 0 **do**
    if $k_i = 1$ **then**
      $Q_0 \leftarrow Q_1 \dot{+} Q_0$; $Q_1 \leftarrow Q_1 \dot{+} Q_1$ (*)
    else
      $Q_1 \leftarrow Q_1 \dot{+} Q_0$; $Q_0 \leftarrow Q_0 \dot{+} Q_0$
    end if
  end for
  return Q_0
end function

Figure 4: simplified scheme of the implementation of the scalar product algorithm (Montgomery ladder).



monitoring will not reveal the state of the private key internal bits. The algorithm is based on the following decomposition,

$$\begin{aligned} Q = k \dot{\times} P = [([(k_{w-1} \times 2 + k_{w-2}) \times 2 + \ldots] \\ \times 2 + k_1) \times 2 + k_0] \dot{\times} P = \\ ([([k_{w-1} \dot{\times} P] \dot{\times} 2 \dot{+} k_{w-2} \dot{\times} P) \dot{\times} 2 \dot{+} \ldots] \\ \dot{\times} 2 \dot{+} k_1 \dot{\times} P) \dot{\times} 2 \dot{+} k_0 \dot{\times} P \end{aligned} \quad (2)$$

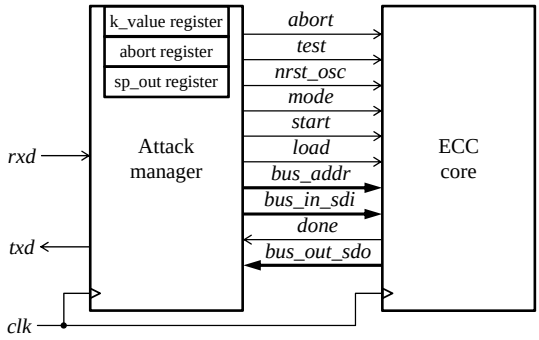
for a k of w bits and in which ($\dot{\times}2$) is a *doubling* operation.

In CTL, a simplification is used by restricting the scalar k to the range $[2^{161}, 2^{162} - 1]$. This keeps the msb of the 162 bits of k to 1 and avoids adding two inverse points. The parallel pseudo-code of the implemented algorithm is presented in Fig. 3.

D. Hardware implementation

Point *adding* and *doubling* modules are the basis of the hardware implementation running the Montgomery ladder algorithm described in Fig. 3. Since the point *doubling* can be easily generated with slight modifications of the *adding* hardware, a versatile *point adding/doubling module* has been conceived, both operations requiring exactly the same amount

Figure 5: Schematic of the crypto-attack analysis bank.



of clock cycles to complete. The operation is selected on the fly after point analysis. As illustrated in Fig. 4, a full parallelized implementation has been selected for the synthesis in the FPGA. While a module performs an *addition*, the other performs a *doubling*. Multiplexers are controlled by the bits of k. The controller of this processing unit is not shown. With this implementation, the scalar product requires 492 clock cycles per bit of k plus an initialization clock cycle that gives a total of 79,705 clock cycles. Running at $100\,\mathrm{MHz}$, this gives a time lower than $800\,\mu\mathrm{s}$.

E. ECC core test infrastructure

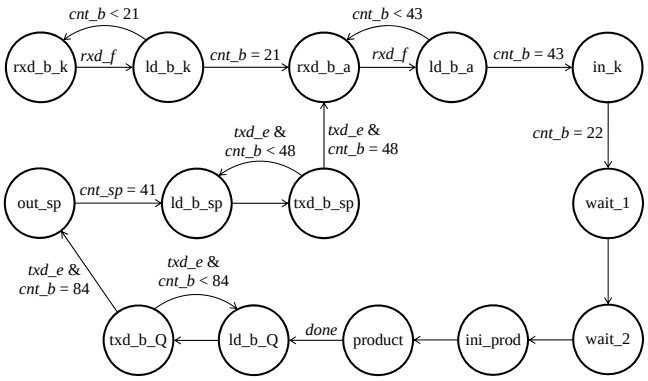
The production test is made easier by a test controller that manages internal scan-paths. Two external signals, $test$ and $nrst\_osc$, activate the test mode. Eight parallel scan-paths provide access to most of the internal flip-flops and serial-data-in/out is made through the same 8-bit data-in and data-out ports. These parallel scan-paths give access to the Q_0 and Q_1 registers. Any time the $test$ signal is active, the content of these registers can be output through the serial-data-out ports. A total of 41 bytes (the plain content of the registers) are output using the test countermeasures as described in paper [11]. The FPGA allows for modification of the included countermeasures if required.

For $test = 1$ and $nrst\_osc = 0$ the test mode is selected. The content of the random generators can be initialized using the serial-data-inputs. The value that is initialized can be used only once for executing a scalar product, since when the ECC core is switched to normal mode, the contents of the random generators are randomized immediately.

IV. CRYPTO-TEST-LAB

ECC core is designed as the core for the CTL platform validating test strategies for crypto-cores. In this section the wrapper organized around the ECC core is presented, whose objective is to facilitate the mounting of attacks. CTL aims to validate the security of scan-path modifications and auxiliary units meant for this objective. The final goal is to avoid the need of blowing up the scan-path ports and, therefore, to be able of using them for diagnostics in after-marked failures.

Figure 6: State machine controller of the crypto-attack analysis bank.



A. The crypto-attack analysis bank

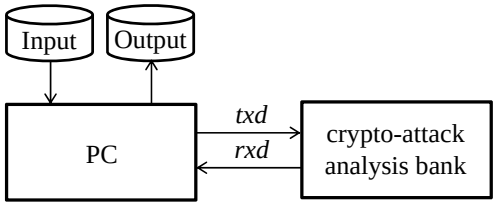
In Fig. 5 the crypto-attack analysis bank is presented. It includes an ECC core and an *attack manager* that takes control of the ECC core, always forcing $mode = 0$. Externally, the *attack manager* is controlled by a serial communication link using an UART interface. It manages the attack sessions as described below.

When the attack session initializes, the *attack manager* waits for a k_A value and stores it in its internal k_value register. Then, the attack begins taking the following actions:

1) Waits for an elliptic curve point P and an aborting step of the Montgomery ladder algorithm. After this, it initializes registers Q_0 and Q_1 of ECC co-processor as described in Fig. 3 and stores the aborting step value in the abort register of the manager.
2) Initializes the content of random generators with the value stored in the k_value register using the test infrastructure.
3) Starts the scalar product $k_A \dot{\times} P$.
4) Aborts the scalar product when the number of iterations in the Montgomery ladder algorithm reaches the value of the abort register. The value of k_A in the ECC co-processor is lost.
5) Reads and sends to the serial channel txd the content of registers Q_0 and Q_1 (the address bus is expanded to 7 bits in order to include Q_1).
6) Reads again the content of Q_0 and Q_1 through scan-paths using the test infrastructure with countermeasures [11] and the content is stored in sp_out register for further reading through serial communication channel txd.

Fig. 6 presents the state diagram of the *attack manager* controller. After power-on reset, it starts the attack session in state **rxd_b_k**. From this state to state **in_k**, data reading from the UART allows to initialize the ECC core. States **wait_1** and **wait_2** allow synchronization. Signal $start$ is asserted in state **ini_prod**. The scalar product is calculated in state **product**. In the following states, all the information is sent finally to the UART serial channel txd as formerly described.

Figure 7: Block diagram of the Crypto-Test-Lab.



B. The CTL platform

In Fig. 7 the block diagram of the Crypto-Test-Lab is presented. It includes a desktop computer (PC) and the *crypto-attack analysis bank*. They are connected through an asynchronous serial link at $921.6\,\mathrm{kBd}$.

The CTL executes attack sessions, one per run. An attack session is defined in a binary input file and the results generated after the attack are recorded in another output binary file. The attack session starts with an initialization, in which the value of k_A is preset, and follows with a number of attacks that use the same k_A. For each attack, the input file also includes the point P and the aborting step of the Montgomery ladder algorithm. All this information is copied to the output file together with the results obtained from the Q_0 and Q_1 registers as described before. This file is later processed to evaluate the validity of the secure test countermeasures.

V. Results

A. Performance of the platform

The ECC core has been synthesized in an FPGA EP3C16F484C6N device of the Cyclone III family from Altera included in a DE0 board from Terasic. The complete synthesis requires 7053 logic elements (46% of the device capability), including 2712 logic elements for registers. It can run up to $116.47\,\mathrm{MHz}$.

Afterwards, the crypto-attack analysis bank including an ECC core has been synthesized in the same device. It requires 8795 logic elements (57% of the resources), from which 3284 are used for registers. It can operate at a maximum speed of $118.76\,\mathrm{MHz}$, but in the DE0 board the system clock runs at $50\,\mathrm{MHz}$.

A single attack takes a total time between $2.1\,\mathrm{ms}$ to $3.7\,\mathrm{ms}$ depending on the aborting step. A session including 1 million attacks is executed in 48 minutes on average.

B. Performance of the scan-path countermeasure

CTL platform has been used to evaluate the [11] countermeasure. It is meant for scan-path protection and thus the attack has been mounted to discover k using the [15] methodology but applied in the ECC core, as described by A. Das in [16].

In this attack it is assumed that the adversary has a limited access to the system. He doesn't have any direct access to the registers containing k since they are not connected to the scan-chain. He can input data to the ECC core but restricted to elliptic curve points, either P or Q_{Bi} as described in Figure

1. Once the unit starts he can interrupt it at any iteration of the Montgomery algorithm, see Figure 3, switch to test mode and scan-out the content of Q_0 and Q_1. He also assumes that scan-paths are compressed, as described in [15], and therefore uses the parity evaluation of the output bits as a feedback for carrying out the attack.

To prepare the attack the adversary builds a model in his computer of the ECC core and calculates the parity of Q_0 and Q_1 at the point of interruption. For each bit of k, he searches for a pair of points Q_{B0} and Q_{B1} that, when applied to the model, produce different parity bits when a particular bit of the key, k_i, is 0 or 1. Once the pair is found he applies it to the real system and validates the hypotheses estimating the real value of k_i. This process is repeated for the 163 bits of the key. In the attack methodology, points Q_{Bi} are generated randomly inside the elliptic curve and pairs may become valid for estimating many bits of the key.

In our experiments we have repeated the attack for two different configurations. In the first (SP), scan-paths with standard compressing are used. In the second (DSP), scan-paths in differential mode as described in [11] are used before the compressing circuit. In this second case the differential mode act as pre-compressor of (1:2) ratio that does not preserve the parity property.

In Table I results are presented. Six different keys are used (col-1). For each, the attack is repeated 50 times and averages are presented in each row of the table. Columns (col-2&4) show the average number of points selected successfully for the attack, pairs are organized from these sets. Columns (col-3&5) present the degree of success achieved by the adversary, it counts the percentage of bits correctly estimated. It is remarkable to see that in the DSP only half of the bits are correctly estimated, which is the most secure scenario. Finally, in (col-6) the difference in the number of points between the attacks SP and DSP is evaluated. It is noticeable to see that from the point of view of the adversary there is no significant difference and therefore the presence of the DSP is disguised.

Table I: Results of the attacks performed in the scan-path and differential scan-path implementation of the ECC core.

	SP		DSP		
Key k	# points	Bits estim. OK (%)	# points	Bits estim. OK (%)	# points difference SP - DSP (%)
1 (avg. 50)	9.38	100	9.46	**50.70**	0.85
2 (avg. 50)	9.68	100	9.82	**50.01**	1.45
3 (avg. 50)	9.80	100	9.72	**50.31**	-0.82
4 (avg. 50)	9.90	100	9.66	**49.22**	-2.42
5 (avg. 50)	9.68	100	9.62	**50.68**	-0.62
6 (avg. 50)	9.78	100	9.80	**48.85**	0.20
Average	9.70	100	9.68	**49.97**	-0.24

VI. Conclusions

In this paper a Crypto-Test-Lab is presented. It is meant for the design, prototyping and validation of solutions improving the security of test infrastructures for ECC cryptographic applications. The platform includes an Elliptic Curve Cryptography core using a test infrastructure of eight scan-paths. A number of countermeasures to improve the testing security are included. The platform is synthesized in an FPGA and is controlled from a desktop computer that can organize attack sessions and evaluate the effectiveness and validity of the countermeasures applied. With this platform, 1 million attacks can be run in 48 minutes.

Acknowledgment

Authors wish to thank Néstor Tuneu for his support in the elaboration of this paper. This work has been partially supported by the Spanish Ministry of Economics and Competitiveness, project reference TEC2013-41209-P.

References

[1] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[2] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 26(1):96–99, 1983.

[3] FIPS Pub. 197: Advanced encryption standard (aes). federal information processing standards publication, us department of commerce. *Information Technology Laboratory (ITL), National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA*, 2001.

[4] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[5] Colin M Maunder and Rodham E Tulloss. *The test access port and boundary scan architecture*. IEEE Computer Society Press Los Alamitos/Washington, DC, 1990.

[6] Aijiao Cui, Yanhui Luo, Huawei Li, and Gang Qu. Why current secure scan designs fail and how to fix them? *Integration, the VLSI Journal*, 56:105–114, 2017.

[7] Junfeng Fan, Hua Xie, and Yiwei Zhang. On the use of scan chain to improve physical attacks. In *Test Symposium (ATS), 2014 IEEE 23rd Asian*, pages 354–357. IEEE, 2014.

[8] Data Encryption Standard et al. Federal information processing standards publication 46. *National Bureau of Standards, US Department of Commerce*, 1977.

[9] Sk Subidh Ali and Ozgur Sinanoglu. Scan attack on elliptic curve cryptosystem. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), 2015 IEEE International Symposium on*, pages 115–118. IEEE, 2015.

[10] Laurent Sourgen. Security locks for integrated circuit, March 31 1992. US Patent 5,101,121.

[11] Salvador Manich, Markus S Wamser, Oscar M Guillen, and Georg Sigl. Differential scan-path: A novel solution for secure design-for-testability. In *Test Conference (ITC), 2013 IEEE International*, pages 1–9. IEEE, 2013.

[12] Alberto Fuentes Rodríguez, Luis Hernández Encinas, Agustín Martín Muñoz, and Bernardo Alarcos Alcázar. A toolbox for dpa attacks to smart cards. In *International Joint Conference SOCO13-CISIS13-ICEUTE13*, pages 399–408. Springer, 2014.

[13] National Institute of Standards and SP Technology (NIST). 800-56a: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revision 2), 2013.

[14] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 291–302. Springer, 2002.

[15] Jean Da Rolt, Giorgio Di Natale, Marie-Lise Flottes, and Bruno Rouzeyre. New security threats against chips containing scan chain structures. In *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*, pages 110–110. IEEE, 2011.

[16] Amitabh Das. *Differential Scan-Based Side-Channel Attacks and Countermeasures (Differentiële scan-gebaseerde nevenkanaalaanvallen en tegenmaatregelen)*. PhD thesis, KU Leuven, Arenberg Doctoral School, October 2013.