# Hardware Implementation of Elliptic Curve Digital Signature Algorithm over GF($2^{409}$) Using SHA-3

Vladimir Trujillo-Olaya and Jaime Velasco-Medina

*Abstract*—**This paper presents the hardware implementation of the Elliptic Curve Digital Signature Algorithm (ECDSA) by considering the NIST recommended elliptic curve in binary fields GF($2^{409}$) and the NIST recommended Secure Hash Algorithm-3 (SHA-3). In this case, four modules are implemented: modular arithmetic hardware, finite field arithmetic hardware, an elliptic curve cryptoprocessor over GF($2^{409}$), and SHA-3 module. The implementation is described in VHDL language, synthesized on the Cyclone V 5CSEMA4U23C6N using Intel-Quartus II V 19.1, and verified using Signal Tap II Logic Analyzer. The synthesis and performance results show a good area-throughput trade-off, and it is suitable for high-performance cryptographic applications for embedded systems.**

*Index Terms*—**ECDSA, finite fields, elliptic curve cryptography, hash functions, FPGAS.**

## I. Introduction

Cryptography plays an important role in the security of the information and specially to protect or exchange confidential information among constrained devices such as wireless sensor network (WSN), radio frequency identifier (RFID), and devices for the Internet of Things (IoT).

The Internet of Things represents a network of devices that are interconnected, and the amount of information must be securely stored and processed. Securing this information is a challenge in IoT. Many IoT devices are inexpensive with restrains in memory and computational resources. Some devices are unable to support the implementation of asymmetric cryptography. Therefore, designers of such devices must create their own protocols.

Elliptic Curve Cryptography (ECC) can be used to guarantee confidentiality and can be used in applications for ensure data integrity and authenticity by using Elliptic Curve Digital Signature Algorithm (ECDSA) [1]

In this context, ECDSA is based on elliptic curve cryptography and was proposed by Victor Miller [2], and Neal Koblitz [3] in 1985, where ECC is a good alternative to RSA, the conventional public-key system used on the Internet today. The main advantage is that ECC offers equivalent security with smaller key sizes over other public-key systems, that means ECC provides the same level of security as RSA and uses fewer computational resources.

Due to its computational advantages, ECC is particularly well suited for mobile and wireless applications where the computational platforms are constrained in the amount of available computational capacity, memory capacity, and battery power.

In cryptography, a digital signature scheme is a number, which has been created using the signer's secret key and the contents of the message that is being signed. In this case, ECDSA enables the sender to attach a signature to a message or document, which is processed by a hash function and then is signed using the sender's private key. The signature must be verified for a third-party without the knowledge of the signer's secret key. Additionally, the signature must be linked with the message so that a forger cannot copy it to another message. The receiver decrypts the signature and checks for its validity.

ECDSA is a variant of the Digital Signature Algorithm (DSA) [4] and it is based on Elliptic curve cryptography. It was accepted as an ISO standard (ISO 14888-3 in 1998), as an ANSI standard (ANSI X9.62 in 1999) [5], and as an IEEE standard (IEEE P1363-2000) and a FIPS standard (FIPS 186-2[4], FIPS 186-3[6], and FIPS 186-4 [7]).

Due to the performance of cryptographic algorithms is crucial for real world applications, the hardware technologies can be used for implementing with high performance and low cost. It is accepted that cryptographic algorithms implemented in hardware are physically more secure and cannot be easily read or modified by an external agent.

In this paper, it is presented a modified implementation of the Elliptic Curve Digital Signature Algorithm using a binary extension field recommended by NIST and the NIST recommended secure hash algorithm 3 (SHA-3)[8], which specifies a family of functions based on Keccak, the winning algorithm selected from NIST's SHA-3 Competition.

## II. Background

### A. Elliptic Curve Cryptography

Elliptic curves are described by the set of solutions to certain equations in two variables. In particular, elliptic curves defined over a finite field are of central importance in public-key cryptography.

For the binary field GF($2^m$), the standard or (Weierstrass) equation for a non-supersingular elliptic curve is as shown

$$y^2+xy=x^3+ax^2+b \tag{1}$$

where $a$ and $b \in$ GF($2^m$), $b \neq 0$. It is well known that the set of points $G = (x, y)$, where $x, y \in$ GF($2^m$), that satisfy the

Manuscript received November 5, 2020; revised February 11, 2021. This work was supported in part by Universidad del Valle under the Grant CI 21068, and the Colombian Ministry of Science, Technology, and Innovation (Minciencias).

V. Trujillo-Olaya is with Universidad de San Buenaventura, Cali, Colombia (e-mail: vtrujillo1@ usbcali.edu.co).

J. Velasco-Medina is with Universidad del Valle, Cali, Colombia (e-mail: jaime.velasco@correounivalle.edu.co).

equation (1), together with the point *O*, call the point at infinity, form an additive abelian group $E_{a,b}$ with *O* serving as its identity.

Point addition and doubling can be computed in terms of the basic finite field operations (addition, multiplication, squaring and inversion), which define the overall efficiency of elliptic curve calculations. Namely, point addition (doubling) requires the following $GF(2^m)$ operations: (two additions, one squaring (two for doubling), one inversion, and nine (five for doubling) additions. For situations where inversion in $GF(2^m)$ is relative expensive to multiplication, it may be advantageous to represent elliptic points using projective coordinates [9].

In a cryptographic application, the elliptic curve will be selected so that $E_{a,b}$ will contain a large subgroup of prime order. NIST has recommended elliptic curves over the finite fields $GF(2^m)$, where m = {163, 233, 283, 409, 571}.

The fundamental and most expensive operation underlying the elliptic curve cryptographic schemes is the point multiplication or scalar multiplication *kG* where *k* is an integer and *G* is an elliptic point, and the point multiplication *kG* can be decomposed into point additions and point doublings.

The basic technique for point multiplication is the double-and-add method, also known as the binary method, which is the additive version of the repeated-square-multiply method for exponentiation. This algorithm requires, in average, $t/2$ point additions and $t$ point doublings, where $t = \lfloor \log_2 K \rfloor + 1$. In [9], the authors describe several algorithms to efficiently compute *kG*. Our design uses López and Dahab method [10], which does not have any extra storage requirements and the same operations (an addition and doubling) are performed in each iteration of the main loop, thereby potentially increasing resistance to timing attacks. In terms of finite field multiplication, the approximate cost of computing *kG* using López and Dahab method is $6m + 20$.

## B. Elliptic Curve Digital Signature

The Elliptic Curve Digital Signature is a variant of the Digital Signature Algorithm, which provides same security level as DSA for smaller key size and it provides authentication, integrity, and non-repudiation of messages. The ECDSA consists of key generation, signature generation and signature validation that are presented in Algorithm 1, Algorithm 2 and Algorithm 3, respectively.

| **Algorithm 1. ECDSA key pair generation [7]** |
|---|
| **Input:** A valid set of elliptic curve domain parameters |
| **Output:** A key pair $(Q,d)$ |
| 1. Select a random integer $d$ in the interval **[1, n-1]** |
| 2. Compute the point $Q=(x_Q, y_Q)=dG$ |
| 3. The key pair is $(Q, d)$, where $Q$ is the public key and $d$ is the private key. |

| **Algorithm 2. ECDSA Signature generation [7]** |
|---|
| **Input:** message $M, n, d$ |
| **Output:** $(r, s)$ |
| 1. Compute the hash value $e = H(M)$ *(in this paper SHA-3)* |
| 2. Select a random integer $k$ in the interval **[1, n-1]** |
| 3. Compute the point $(x_1, y_1)=kG$ |
| 4. Convert the field element $x_1$ to an integer $\overline{x_1}$, |
| 5. Set $r = \overline{x_1}$ mod $n$ |
| 6. If $r = 0$, then go to step **2** |
| 7. Compute $s = k^{-1}(e + dr)$ **mod** $n$ |
| 8. If $s = 0$, then go to step **2** |

| **Algorithm 3. ECDSA Signature verification [7]** |
|---|
| **Input:** message $M', (r', s'), Q$ |
| **Output:** Signature valid |
| 1. Compute $e' = H(M')$ |
| 2. Check **r', s'** are integers in the interval **[1, n-1]** |
| 3. Compute $c = (s')^{-1}$ **mod** $n$ |
| 4. Compute $u_1 = e'c$ **mod** $n$ and $u_2 = r'c$ **mod** $n$ |
| 5. Compute the EC point $(x_1, y_1) = u_1G + u_2Q$. (If $u_1G + u_2Q$ is the point at infinity, then reject the signature.) |
| 6. Convert the field element $x_1$ to an integer $\overline{x_1}$ |
| 7. Compute $v = \overline{x_1}$ **mod** $n$ |
| 8. If $r' = v$, then the **signature is verified** Else **signature invalid** |

## III. Procedure Proposed ECDSA Hardware Architecture

This section presents the hardware architecture for the ECDSA processor. As it is mentioned before, the implementation of ECDSA can be divided into 4 modules: modular arithmetic GF(p), Finite field arithmetic $GF(2^m)$, Elliptic Curve arithmetic over $GF(2^m)$, and the hash function H(m). Therefore, the hardware implementation of ECDSA requires special attention.

### A. Modular Arithmetic GF(p)

The modular arithmetic refers to operations that require the *mod n* operation presented in ECDSA algorithm. In this case, modular addition is the operation wherein if the addition result is greater or equal to the number *n*, then it is performed the subtraction by the number *n*. The modular multiplication uses the Montgomery multiplication algorithm, and the inversion is performed by using the modified Montgomery inversion algorithm presented in [11].

### B. Finite Field Arithmetic over $GF(2^m)$

Finite field multiplier plays an important role and determines the performance of the cryptoprocessors due to the elliptic curve point multiplication involves intensive field multiplications operations.

In this work it is presented a modified version of the multiplier presented in [11], which has a systolic architecture where every Processing Element (PE) computes a partial result of the multiplication. Fig. 1 illustrates the internal architecture for a PE, which includes:

- The Unity-degree Reduction Cell that corresponds to the modular reduction.
- NAND cell consists of *m* NAND gates in parallel to perform the NAND operations of one bit of operand B and all the bits of the reduced operand A corresponds to the bit-multiplication node.
- XOR cell consists of m XOR gates to perform the addition of two elements.

Due to a design of a pure systolic array presents more hardware consumption. Fig. 2 shows the design of a sequential multiplication using the PEs presented in [11]. Where, the intermediate partial multiplication results are computed by using the same PE.

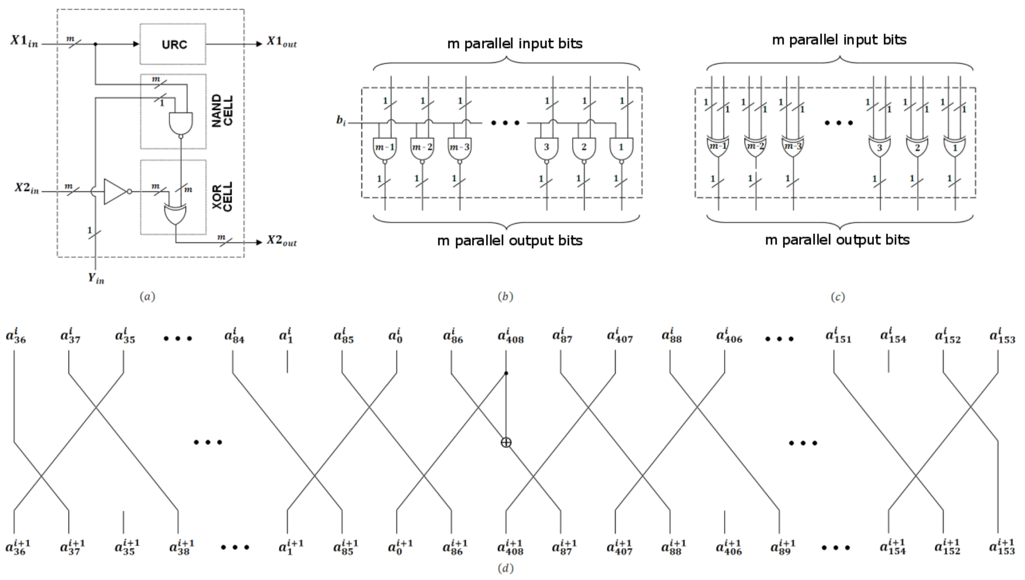


Fig. 1. Hardware architecture of a regular PE. a) Internal architecture of PE. b) NAND cell architecture. c) XOR cell. d) URC block architecture.

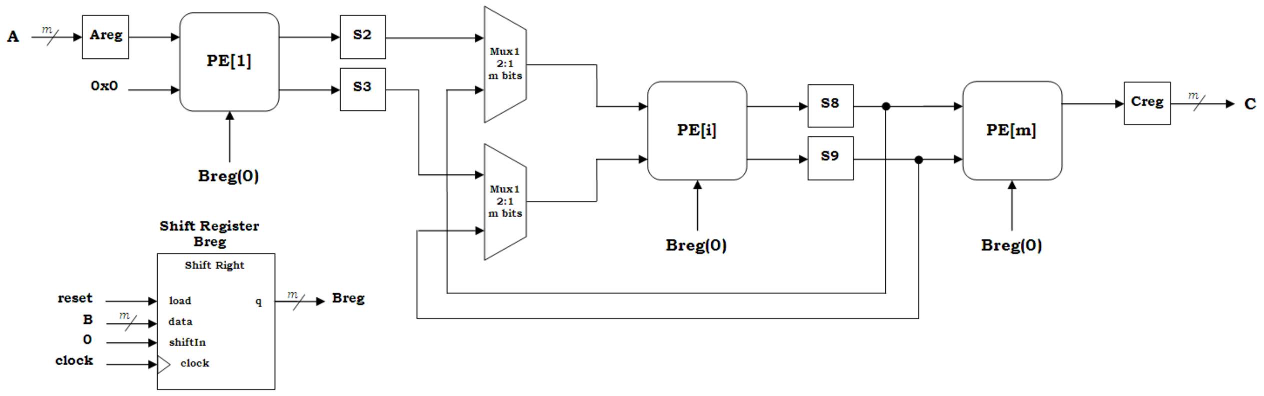


Fig. 2. Hardware architecture for the polynomial basis multiplication.

The processing element PE[i] performs the m-2 partial products, while PE [1] and PE[m] have irregular forms.

The operand B is stored in a shift register and its LSB is an input of the block PE[i]. In this case, the execution time of the polynomial basis multiplier is m clock cycles. The inversion operation over GF($2^m$) is performed by using the Itoh-Tsujii algorithm [12].

### *C. Elliptic Curve Cryptoprocessor over GF($2^m$)*

The cryptoprocessor architecture using polynomial basis has two register files and several functional blocks, which allow to calculate the addition, squaring, multiplication and inversion arithmetic over GF($2^{409}$). In order to carry out the point multiplication *kG* [9], one FSM is used to realize the point addition and doubling operations. Additionally, a main controller is used to control the I/O registers to generate the control sequences for the scalar multiplication to process the key and to initialize the cryptoprocessor. The architecture is oriented to carry out parallel processing by considering the duplication of functional blocks. The processor architecture is shown in Fig. 3, shows the multiplexor blocks that are designed to select the stored data from the register files. The register files store the elliptic curve parameters and the results of the GF($2^m$) operations.

### *D. Secured Hash Algorithm (SHA – 3)*

Algorithm-3 Keccak was selected because in 2004 SHA-1 was found to be weak, and the threat was carried to SHA-2 also. In this case, Federal Information Processing Standard (FIPS) 202[8], SHA-3 Standard, specifies the Secure Hash Algorithm-3 (SHA-3) family of functions on binary data. The SHA-3 functions are based on the KECCAK algorithm that NIST selected as the winner of the SHA-3 Cryptographic Hash Algorithm Competition. Hash functions are used in many important information security applications, including the generation and verification of digital signatures, key derivation, and pseudorandom bit generation. In this work, it is used the SHA3-512, where the state for the Keccak-f[1600] permutation is comprised of 1600 bits and uses 24 rounds. In this case, Federal Information Processing Standard (FIPS) 202 [8], SHA-3 Standard, specifies the Secure Hash Algorithm-3 (SHA-3) family of functions on binary data.

The SHA-3 functions are based on the KECCAK algorithm that NIST selected as the winner of the SHA-3 Cryptographic Hash Algorithm Competition. Hash functions are used in many important information security applications, including the generation and verification of digital signatures, key derivation, and pseudorandom bit generation. In this work, it is used the SHA3-512, 1600 bits and uses 24 rounds.

This Standard also specifies the KECCAK-p family of mathematical permutations, including the permutation that underlies KECCAK, to facilitate the development of additional permutation-based cryptographic functions.

Fig. 4 shows the structure of the Keccak function, and Fig. 5. shows the algorithm for the Keccak function.

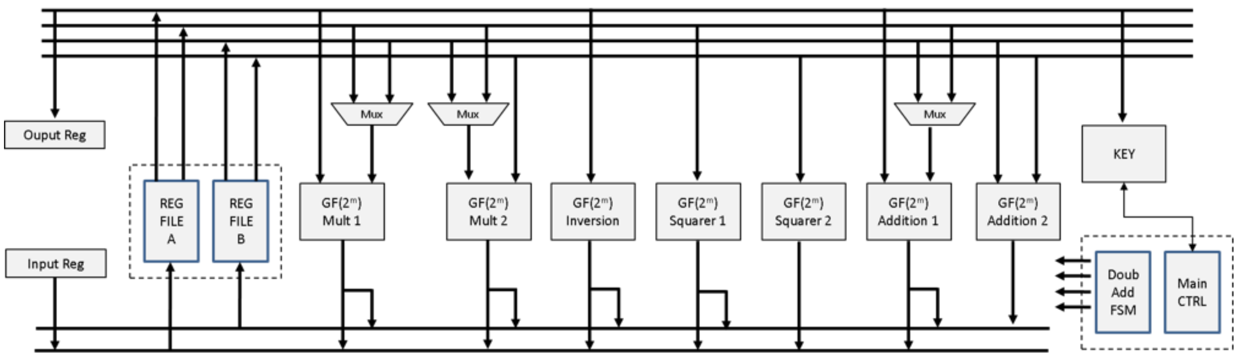


Fig. 3. Elliptic curve cryptoprocessor using polynomial basis.

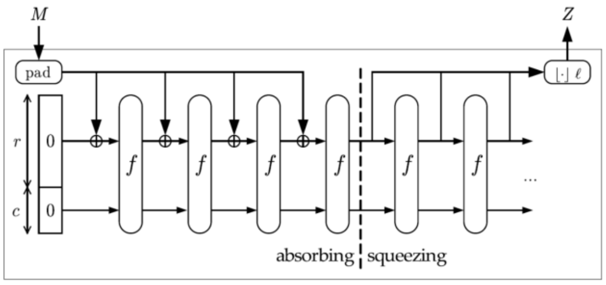


Fig. 4. Structure of the Keccak function.

Where the block $f$ is an iterative structure, and it consists of a sequence of rounds. The algorithm for the rounds is shown in Table I.

TABLE I: ALGORITHM FOR SPONGE FUNCTION KECCAK

**Algorithm** for sponge function *Keccak* [r,c](M)

```
1. Padding
      $P=M||pad[r](|M|)$
2. Initialization
      $s=0^b$
3. Absorbing Phase
      for $i=0$ to $|P|_r-1$ do
            $s=s\oplus(P_i \,||\, 0^{b-r})$
            $s=Keccak\text{-}f[b](s)$
      end for
4. Squeezing phase
      $Z=\lfloor s \rfloor_r$
      While $|Z|_r r<l$ do
            $s=Keccak\text{-}f[b](s)$
            $Z=Z||\lfloor s \rfloor_r$
      end while
   Return $\lfloor Z \rfloor_l$
```

## E. Hardware Architecture for ECDSA over GF($2^m$)

The hardware implementation of the ECDSA is based on Algorithms 1-3. The datapath is mainly conformed by elliptic curve processor, the modular arithmetic processor, and the SHA-3 block, as it is shown in Fig. 6.

In addition, there are other blocks such as 16 × 409-bits RAM memory, which stores parameters and operation results from other blocks, also the block comp which is used to verify some data are in the range stablished by the algorithms.

The size of input signal parameters is 64-bits and the block parameter buffer stores and shift the data to obtain 409-bit size for each data that is stored into RAM memory. The block control generates all control signals to store data, to make an elliptic curve operation or modular arithmetic operation and has signals to know the state of each block. The output data are stored in the signature block which is built by a shift register with 409-bits input data and 64-bits output data. During the execution, the results are verified to validate the signature and to generate error codes that verify if data r or s are equal to zero.

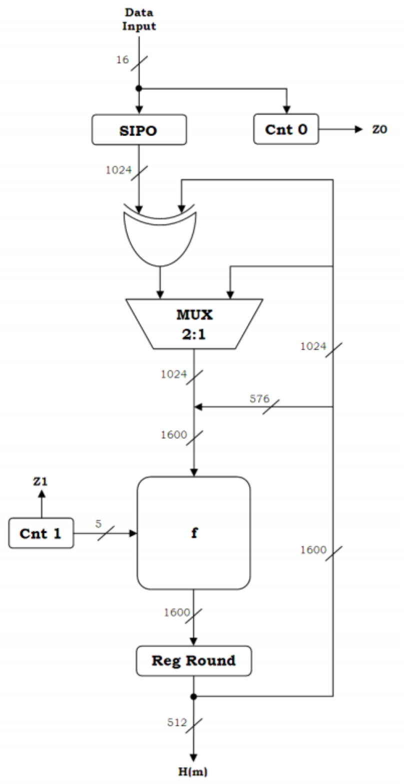


Fig. 5. Hardware architecture for the SHA-3 function.

## IV. HARDWARE VERIFICATION AND SYNTHESIS RESULTS

The ECDSA processor is described in structural VHDL, synthesized on the Cyclone V 5CSEMA4U23C6N using Intel-Quartus II V 19.1, and verified using Signal Tap II Logic Analyzer. Besides, the hardware implementation of the ECDSA was verified by using the Modelsim-Intel to performs functional simulations and the results were verified by the Matlab software. Then, it is used Signal Tap II for in-system hardware verification. In this case, the respective digital signature generation and verification using Signal Tap II are presented in Fig. 7. Where, some stored data in memory are r and s signals, respectively. The signature generation requires 2.176 ms and the Signature verification requires 4.032 ms. Table II presents the synthesis results for the ECDSA hardware implementation.

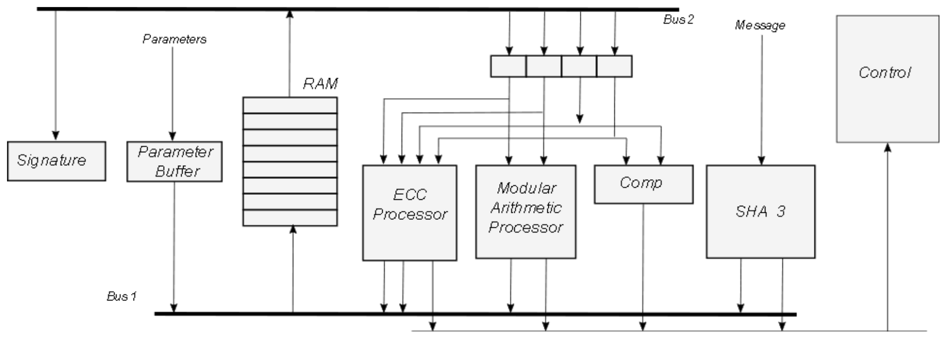
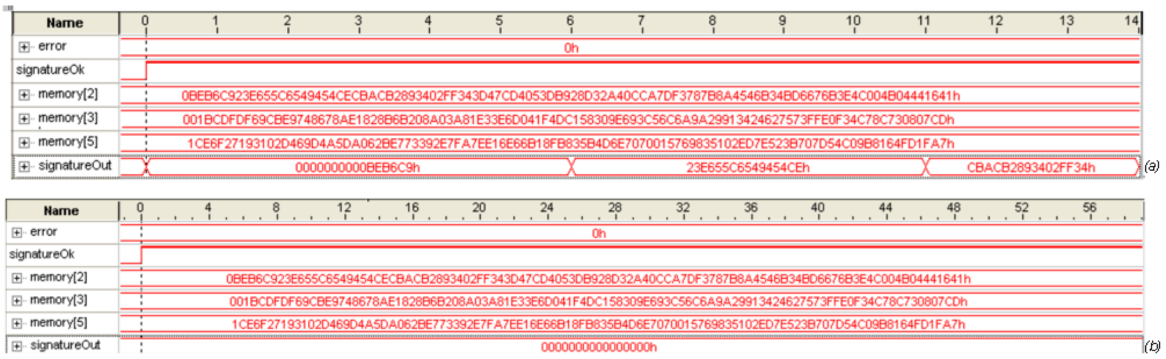


Fig. 6. ECDSA architecture.



```
Name          0   1   2   3   4   5   6   7   8   9   10   11   12   13   14
error         0h
signatureOk
memory[2]     0BEB6C923E655C6549454CECBACB2893402FF343D47CD4053DB928D32A40CCA7DF3787B8A4546B34BD6676B3E4C004B04441641h
memory[3]     001BCDFDF69CBE9748678AE1828B6B208A03A81E33E6D041F4DC158309E693C56C6A9A29913424627573FFE0F34C78C730807CDh
memory[5]     1CE6F27193102D469D4A5DA062BE773392E7FA7EE16E66B18FB835B4D6E707001576983510 2ED7E523B707D54C09B8164FD1FA7h
signatureOut  0000000000BEB6C9h   23E655C6549454CEh   CBACB2893402FF34h                                    (a)

Name          0   4   8   12   16   20   24   28   32   36   40   44   48   52   56
error         0h
signatureOk
memory[2]     0BEB6C923E655C6549454CECBACB2893402FF343D47CD4053DB928D32A40CCA7DF3787B8A4546B34BD6676B3E4C004B04441641h
memory[3]     001BCDFDF69CBE9748678AE1828B6B208A03A81E33E6D041F4DC158309E693C56C6A9A29913424627573FFE0F34C78C730807CDh
memory[5]     1CE6F27193102D469D4A5DA062BE773392E7FA7EE16E66B18FB835B4D6E7070015769835102ED7E523B707D54C09B8164FD1FA7h
signatureOut  0000000000000000h                                                                     (b)
```

Fig. 7. ECDSA results in SignalTap II. a) ECDSA generation. b) ECDSA verification.

TABLE II: SYNTHESIS RESULTS FOR THE ECDSA HARDWARE IMPLEMENTATION ON CYCLONE V 5CSEMA4U23C6N

| Modular Arithmetic | | | Finite Field and Elliptic Curve Arithmetic | | |
|---|---|---|---|---|---|
| Module | ALUTs | FFs | Module | ALUTs | FFs |
| ModuleP | 985 | 821 | Adder | 1.227 | 1.227 |
| AdderP | 1.394 | 820 | Squarer | 818 | 818 |
| MultiplierP | 6.579 | 5.339 | Multiplier | 3.332 | 2.872 |
| InverterP | 12.749 | 5.749 | ASMI | 7.933 | 6.233 |
| IntOp | 21.947 | 13.151 | ECC | 13.798 | 1.3351 |
| HASH Function | | | ECDSA Implementation | | |
| Keccak | 4784 | 2230 | ECDSA | 43749 | 38869 |

Table III presents the comparison for hardware implementations of ECDSA.

TABLE III: COMPARISON OF ECDSA IMPLEMENTATIONS

| | m | ALUTs | Time(ms) | Device |
|---|---|---|---|---|
| [13] | 163 | 64,870 sig-gen | 0.8 | XC6VLX760-2FF1760 |
| | | 25,50 sig-ver | 0.4 | |
| [14] | 163 | 23,675 sig-gen | 0.615 | XC6VLX240T-1FF1156 |
| | | 27,791 sig-ver | 0.672 | |
| [15] | 233 | 88,031 sig-gen | 1.24 | XC6VLX760-2FFL760 |
| | | 26,590 sig-ver | 2.33 | |
| [16] | 163 | 18,504 sig-gen | 0.782 | VIRTEX5-ML50 |
| | | sig-ver | 1.5 | |
| This | 409 | 43.749 sig -gen | 2.176 | 5CSEMA4U23C6N |
| | | sig-ver | 4.032 | |

From Table III, the number of ALUTS for different implementations are discriminated according to the ECDSA operation, however in this work, generation and verification uses the same hardware. The execution time is not taken into account due to this hardware was implemented to 409 bits, while others works were implemented for 163 and 233 bits. And other similar works does not present their results.

## V. CONCLUSIONS

This work presents the hardware implementation of the Elliptic Curve Digital Signature Algorithm over GF($2^{409}$), using polynomial basis representation, were a systolic architecture is implemented which less resource consumption, the Montgomery point multiplication version for the ECC uses GF($2^m$) inversion operation in the las part and it is implemented in and the SHA-3 function, which make this design different from others. This approach gives a trade-off security, performance, and area.

The ECDSA implementation was described in generic structural VHDL, synthesized on the Cyclone V 5CSEMA4U23C6N using Quartus II V 19.1, and verified using modelsim-Intel and SignalTap II Logic Analyzer.

Despite of the m size of the finite field, the synthesis results and comparisons show that the design uses few area resources, and it can be suitable for cryptographic applications on embedded systems.

## CONFLICT OF INTEREST

The authors declare no conflict of interest.

## AUTHOR CONTRIBUTIONS

Jaime Velasco-Medina conducted the research, did 40% of analysis and design, and 40% of writing and did the proof reading. Vladimir Trujillo-Olaya, did the 60% of analysis and design, 100% of implementation and simulation, and wrote 60% of the paper; all authors has approved the final version.

## ACKNOWLEDGMENT

The authors wish to thank the Colombian Ministry of Science, Technology, and Innovation (Minciencias) and Universidad del Valle for the project financial support CI 21068. Also, special thanks to Julian F. Bravo-Parra for assistance with previous designs.

## REFERENCES

[1] N. Koblitz, S. Vastone, and A. Menezes, “The state of elliptic curve cryptography,” *Designs, Codes and Cryptography*, pp. 173–193, March 2000.

[2] V. S. Miller, “Uses of elliptic curves in cryptography, advances in cryptology,” *Lecture Notes in Computer Science* (*LNCS*), vol. 218, pp. 417-426, springer-verlag, New York, USA, 1986.

[3] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, pp. 203-209, 1987.

[4] National Institute of Standards and Technology, Digital Signature Standard. (February 2000). *FIPS Publication 186-2*. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/fips/186/2/archive/2000-01-27/documents/fips186-2.pdf

[5] ANSI, “Public key cryptography for the financial services industry,” The Elliptic Curve Digital Signature Standard (ECDSA), ANSI X9.62, American National Standard Institute, 1999

[6] National Institute of Standards and Technology. Digital Signature Standard. (June 2009). *FIPS Publication 186-3*. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf

[7] National Institute of Standards and Technology. Digital Signature Standard (DSS). (July 2013). *FIPS Publication 186-4*. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[8] National Institute of Standards and Technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. *FIPS Publication 202*. (August 2015). [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

[9] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 2004.

[10] J. Lopez and R. Dahab, “Fast multiplication on elliptic curves over GF($2^n$) without precomputation,” *Cryptographic Hardware and Embedded Systems*, pp. 316-327, 1999.

[11] P. K. Meher, “Systolic and super-systolic multiplier for finite field GF($2^m$) based on irreducible polynomials,” *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 55, no. 4, May 2008.

[12] T. Itoh and S. Tsujii, “A fast algorithm for computing multiplicative inverses in GF($2^m$) using normal bases,” *Information and Computation*, vol. 78, pp. 171-177, 1988.

[13] G. Nabil, K. Naziha, F. Lamia, and K. Lotfi, “Hardware implementation of Elliptic Curve Digital Signature Algorithm (ECDSA) on Koblitz Curves,” in *Proc. 2012 8th International Symposium on Communication Systems, Networks & Digital Signal Processing*, July 2012 , pp. 1-6.

[14] B. Panjwani and D. C. Mehta, “Hardware-software co-design of elliptic curve digital signature algorithm over binary fields,” in *Proc. 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Aug. 2015, pp. 1101-1106.

[15] N. Ghanmy, L. C. Fourati, and L. Kamoun, “Enhancement security level and hardware implementation of ECDSA,” in *Proc. 2013 IEEE Symposium on Computers and Communications (ISCC)*, July 2013, pp.000423-000429.

[16] A. Sghaier, M. Zeghid, C. Massoud, and M. Machout, “Design and implementation of low area/power elliptic curve digital signature hardware core,” *Electronics*, vol. 6. p. 46, June 2017.

**Vladimir Trujillo-Oaya** received his B.S, M.Sc and PhD in Electronics Engineering from Universidad del Valle, School of Electrical and Electronics Engineering in 2004, 2009 and 2014 respectively. He is currently an associate professor at Universidad de San Buenaventura, Cali, Colombia. In 2007, he received one grant from the European Union through the Alfa/Nicron project at the TIMA-INPG, Grenoble, France, working on fault injections on complex digital circuits using VHDL.

Dr. Trujillo-Olaya is currently a professor at Universidad de San Buenaventura, Cali-Colombia. His research interests include hardware implementation of finite fields and cryptosystems, fault tolerant design and complex digital systems design. He has been a reviewer of *IEEE-CAS, IEEE-LASCAS*, *IBERCHIP*, the *IEEE Transactions on VLSI System*, the *IEEE Transactions on Circuits and Systems I, IEEE Transactions on Circuits and Systems II*, and some national publications.

**Jaime Velasco-Medina** received the BS degree in electrical engineering from the Universidad del Valle, Cali, Colombia, in 1985, and the M.Sc. and Ph.D. degrees in microelectronics from the Institute National Polytechnic of Grenoble, Joseph Fourier University, Grenoble, France, in 1995 and 1999, respectively. He held an internship position, for six months, as a technical staff member at AT\&T BellLabs in Allentown, PA, USA, in 1988.

He was the pioneer of the current-based testing for analog and mixed signal circuits, and on-line testing of operational amplifiers. He is currently a faculty professor with the School of Electrical and Electronics Engineering, Universidad del Valle, where he is the director with the Bionanoelectronics Research Group. He has authored more than 40 IEEE papers and 50 peer-reviewed papers in other scientific events and journals. His current research interests include digital systems design for computer arithmetic and digital signal processing, test of analog circuits, and hardware architectures for cryptography, quantum computing, wireless communications, citocomputation, modeling of biological systems, design of graphene-based digital circuits, and computational design of bionanosensors and bionanomachines for nano drug delivery systems.

Dr. Velasco Medina is a reviewer for the *JETTA*, the *IEEE LATW*, the *IEEE SPL*, the IBERCHIP, the IEEE LASCAS, the *IEEE Transactions on VLSI System*, the IEEE Transactions on Signal Processing, the *IEEE Transactions on Circuits and Systems I*, and many other international publications and conferences.