# Hardware implementation of Elliptic Curve Digital Signature Algorithm (ECDSA) on Koblitz Curves

Ghanmy Nabil, Khlif Naziha
Electrical Engineering Department of National school of Engineers of Sfax
Electronic and Information Technology Laboratory (LETI)
Sfax, Tunisia
nabil_ghanmy@yahoo.fr

Fourati Lamia, Kamoun Lotfi
Electrical Engineering Department of National school of Engineers of Sfax
Electronic and Information Technology Laboratory (LETI)
Sfax, Tunisia
lamia.chaari@tunet.tn

*Abstract*— **This paper presents Elliptic Curve Digital Signature Algorithm (ECDSA) hardware implementation over Koblitz subfield curves with 163-bit key length. We designed ECDSA with the purpose to improve performance and security respectively by using elliptic curve point multiplication on Koblitz curves to compute the public key and a key stream generator "W7" to generate private key. Different blocs of ECDSA are implemented on a reconfigurable hardware platform (Xilinx xc6vlx760-2ff1760). We used the hardware description language VHDL (VHSIC Hardware Description Language) for compartmental validation. The design requires 0.2 ms, 0.8 ms and 0.4 ms with 7 %, 13 % and 5 % of the device resources on Slice LUT for respectively key generation, signature generation and signature verification. The proposed ECDSA implementation is suitable to the applications that need: low-bandwidth communication, low-storage and low-computation environments. In particular our implementation is suitable to smart cards and wireless devices.**

**Keywords:** ECDSA, Koblitz curves, PRNG W7, FPGA and Cryptography.

## I. INTRODUCTION

DSA [1] (Digital Signatures Algorithms) play a central role in modern cryptosystems. They are digital counterparts of handwritten. The basic objective of a digital signature is to guarantee authenticity, integrity of a signed message which is transmitted to the receiver and to prove the identity of the transmitter, including no repudiation. ECDSA is a variant of DSA which uses ECC (Elliptic Curve Cryptography) [2]. ECC offers shorter signatures compared to other methods as RSA (Rivest Shamir Adleman). This characteristic make ECC suited for applications on constrained devices. ECDSA is generated in three steps; key pair generation, signature generation and the signature verification. Basic blocks in the ECDSA are PRNG (Pseudo-Random Number Generators) for secret key, public key generator by elliptic curve point multiplication and SHA (Secure Hash Algorithm) to obtain the condensation of message. ECC is the basic operation in ECDSA cryptosystems. It is used three times by the transmitter and the receiver. Hence, it is important to implement it efficiently. Point multiplication is computed with a set of operations point addition and point doubling. Using special binary curves called Koblitz curves can reducing the cost of this operation [3]. All point doublings can be substituted by the Frobenius maps computation, that is squaring, an easy operation over a binary field.

Hardware implementations of complete ECDSA over binary fields are scarce. Jarvinen and Skytta [4] present a Nios II-based ECDSA system on an Altera Cyclone II FPGA over binary field performing key generation in 0.6 ms, signature generation in 0.94ms and verification in 1.61ms. Michael Hutter and all [5] presented the design of 192-bit ECDSA Processor on $F_p$ for RFID Authentication only to signs a message within 859 188 clock cycles (127ms at 6.78MHz). Glas and all [6] propose Prime Field ($F_p$) ECDSA Signature on Virtex V FPGA on $F_p^{256}$ with signature generation (only point multiplication) in 7,15 ms and verification (only double point multiplication) in 9,09 ms. This paper presents hardware implementation of complete Elliptic ECDSA over Koblitz subfield curves with 163-bit key length. We designed ECDSA by using elliptic curve point multiplication on Koblitz curves to compute the public key and a key stream generator W7 to generate private key respectively to improve performance and security.

The remainder of this paper is organized as follows as: In section 1, we give a brief description of the standard Digital Signature Algorithm. In section 2, we describe elliptic curve cryptography over a binary finite field. Section 3, presents ECDSA with emphasis on its different blocks. The simulations and results synthesis of the FPGA implementation for ECDSA architectures are given in the next section. Section 5, concludes this paper.

## II. DIGITAL SIGNATURE ALGORITHM

The digital signature algorithm, DSA, was developed by the NSA and became the digital signature standard, DSS, of NIST (National Institute of Standards and Technology) in 1994 [1]. Digital signatures are fundamental and play a central role in modern cryptosystems. They can be viewed as digital counterparts for handwritten signatures and they are authentic, and non-reusable. Digital signatures used to provide the authentication, no-repudiation and integrity services.

Digital signature algorithms are public-key cryptographic algorithms and thus they involve two keys; one which is private and one which is public. A document is signed with the private key by the transmitter and the signature is verified with the public key by the receiver (Figure 1). Only the private

978-1-4577-1473-3/12/$26.00 ©2012 IEEE

key needs to be kept in secret in order to prevent other people from forging one's signature.

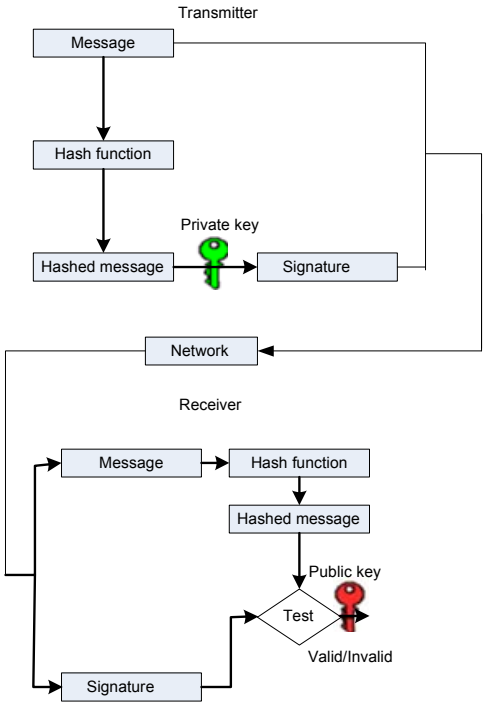


Figure 1. Digital signature algorithm

## III. ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

ECC [7] is a kind of public key cryptosystem, as RSA [8] that offer the same security level with shorter keys [9]. The use of shorter length keys gives less space for key storage, time saving when keys are transmitted. ECC has been proposed separately by N.Koblitz [10] and V.Miller [11]. ECC is defined as the tuple T = ($F_q$, a, b, G, n, h), where $F_q$ is a finite field, q is a prime number, a and b define the elliptic curve on $F_q$. G is a generator point of the elliptic curve, n is the order of G (identity point in the additive group), that is, the smaller integer such that n.G = 0. H is called the co-factor and it is equal to the total number of points in the curve divided by n. ECC's security is based on the discrete logarithm problem, called the Elliptic Curve Discrete Logarithm Problem (ECLDP). The ECLDP consists on given two points P, Q Є $F_q$, to find the positive integer k such as Q = K.P. This problem is of exponential complexity. On the contrary, knowing the scalar k and the point P, the operation K.P is relatively easy to compute. K.P is the point multiplication.

Security services are provided by ECC cryptographic schemes for key agreement, digital signatures. The point multiplication is the most time consuming operation. Elliptic curve operation can be performed according three layers, in a hierarchical way (Figure 2):

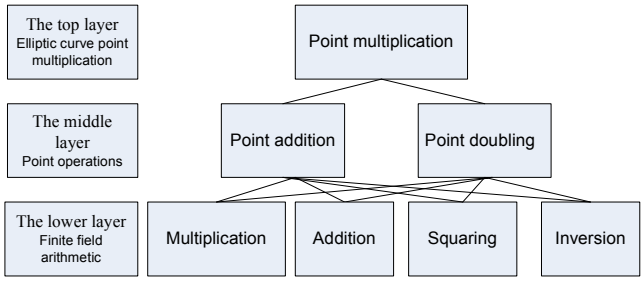


Figure 2. ECC arithmetic hierarchy

At the top layer, the point multiplication is independently of the selected finite field. This operation is the result of adding the point P to itself (n-1) times. That is K.P=P+P+P+…+P, K times. It can be performed using two kinds of sums for the middle layer: point addition, which consists of the sum of two different points (P + Q) and point doubling, which consist of the sum of the same point (P + P).

At the lower layer is the finite field arithmetic. The operators are multiplication, inversion, squaring and adding. The performance of the arithmetic units influences the overall performance of the point multiplication and hence the performance of the ECC cryptographic schemes. We can still increase this performance by using binary finite field which is denoted as $F_2^m$ and given by Eq.(1). This is because the arithmetic with polynomials in $F_2^m$ is like binary arithmetic well suited to be implemented in hardware. The sum of two elements is a simple XOR operation and the elements in $F_2^m$ are binary words that are well stored in m-bit registers.

$$E(F_2^m): y^2 + xy = x^3 + ax^2 + b \quad (1)$$

## IV. ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

ECDSA is a variant of DSA, which uses Elliptic curve cryptography. It was accepted in 1998 as an ISO standard [2], in 1999 as an ANSI standard, and in 2000 as IEEE and NIST standards. Both DSA, ECDSA is generated in three steps; the first and the second are generated by the transmitter of the message which called key pair generation and signature generation and the third step is done by the recipient which is called signature verification.

The transmitter will transmit a message to the receiver. Therefore, he must firstly generate a pair key; the first is secret used for generating the signature. The second is public sent to the receiver. The transmitter generates the signature with her private key and H the hashing function after he send the set message, signature and public key to receiver. This entity will verify the signature with the public key to authenticate the message.

Later, we gives a description for each step of the ECDSA; key pair generation, signature generation and signature verification.

- **Key Pair Generation** (Figure 3):

To generate key pair, the sender use key pair generation step. It consists of:

- Select an elliptic curve E ($F_2^m$).
- Select a point P from the curve with order n.
- Choose an integer d from [1, n-1].
- Compute Q = dP.

P is the point generator of the curve, d is called the private key and Q is the public one. For more security, this step needs one pseudo random number generator to choose d and one point multiplication to compute Q.

- **Signature generation** (Figure 4):

To sign the message, the transmitter uses signature generation step. It consists of:

- Choose an integer k from [1, n-1]
- Compute kP = (x1, y1)
- Compute r = x1 mod n
- Compute e = h (m) with m the message, e the message digest and h the hash function
- Compute $s = k^{-1} (e + dr) \bmod n$

The signature is the set (r, s). This step needs a pseudo random number generator to choose k, one point multiplication, one modular reduction, one hash function, addition and modular division.

- **Signature verification**(Figure 5):

To verify the signature, the receiver use Signature verification step. It consists of:

- Compute e = h (m)
- Compute $u_1 = es^{-1} \bmod n$
- Compute $u_2 = rs^{-1} \bmod n$
- Compute $u_1P + u_2Q = (x_2, y_2)$
- Compute $v = x_2 \bmod n$

The receiver must compare v and r. If v = r then the signature is valid else it is invalid.

The demonstration is as presented down:

$$u_1 P + u_2 Q = es^{-1}P + rs^{-1}Q$$
$$= es^{-1}P + rs^{-1}dP$$
$$= (e + dr)\, s^{-1}P$$

Since $s = k^{-1} (e + dr)$ so $k = s^{-1}(e + dr)$

As a result $u_1 P + u_2 Q = kP$

Consequently $(x_2, y_2) = (x_1, y_1)$

Therefore, $x2 \bmod n = x_1 \bmod n$

Which gives that the signature is well verified: v = r.

This step needs one hash function, two modular divisions, two point multiplications, one point addition and one modular reduction.

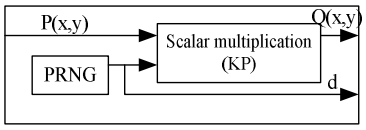


Figure 3. Key Pair Generation

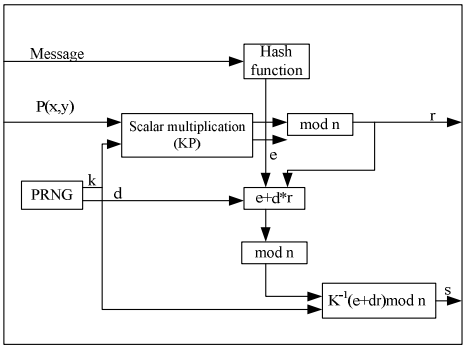


Figure 4. Signature generation scheme

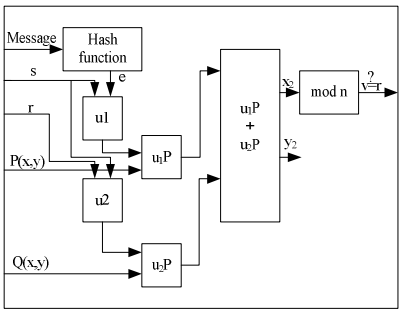


Figure 5. Signature verification scheme

### *A. PRNG W7*

A pseudo-random number generator (PRNG) is used by the transmitter to generate random integers in key pair generation and signing for more security. In this context there's many algorithms as A5/1 [12], W7 [13], E0 [14], RC4 [15] and Helix [16]. A comparison between these generators is given by [17] in terms of time, number of clock cycles, occupation, frequency and Throughput. W7 generator has many advantages in terms of area and performance. W7 has been proposed by S. Thomas, D. Anthony, T. Berson, and G. Gong [18]. It is a symmetric key algorithm supporting key lengths of 128-bit. It has more trustworthy solution for random number generator. W7 cipher contains eight similar models, C1, C2,…, C8. Each model consists of three LFSR's and one majority function. W7 architecture consists of a control and a function unit. The function unit is responsible for the key-stream generation. This unit contains eight similar cells. Each cell has two inputs and one output. The proposed architecture for the hardware implementation of one cell is presented in Figure 5. The one input is the key and it's the same for all the cells. The other input consists of control signals. Finally the output is one bit long. The outputs of each cell complete the key stream byte.

Each cell consists of a majority function and three LFSRs which are respectively 38, 43 and 47-bit long. Their initial state which is the same for all is the symmetric encryption key. The 128-bit of the key are mapped to the LFSRs initial state as the following:

$LFSR_a$ (38-bit) : $LFSR_0 = K_0$, $LFSR_1 = K_1$, ... , $LFSR_{36} = K_{36}$, $LFSR_{37} = K_{37}$

$LFSR_b$ (43-bit) : $LFSR_0 = K_{38}$, $LFSR_1 = K_{39}$, ... , $LFSR_{41} = K_{79}$, $LFSR_{42} = K_{80}$

$LFSR_c$ (47-bit) : $LFSR_0 = K_{81}$, $LFSR_1 = K_{82}$, … , $LFSR_{45} = K_{126}$, $LFSR_{46} = K_{127}$

The three LFSRs together determine when each shift register is clocked. One bit in each register is designated as the clock tap for that register as it's shown in Fig. 6. At each clock cycle the majority value for these taps determines which LFSRs advance. Only the LFSRs whose clock taps agree with the majority advance. The output bit arises after a non-linear function in the register which is a combination of several bits in the LFSR. The non-linear function is a combination of logical-AND functions. The actual key-stream output is taken as the exclusive-OR of the three LFSRs. The key-stream byte is the aggregation of each cell output.

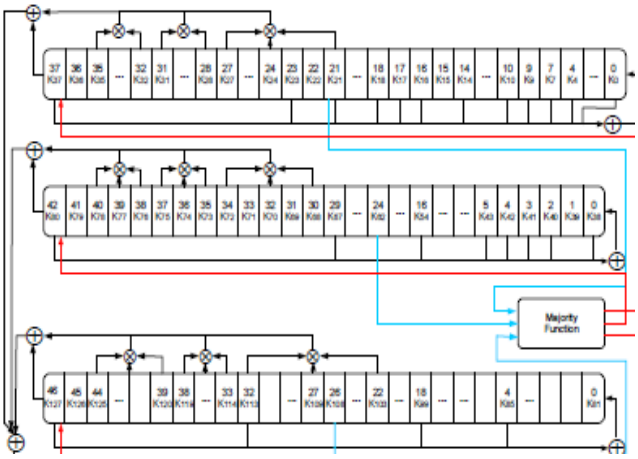


Figure 6. Cell C3 Architecture of W7

## B. *Public key generator*

In ECDSA, public key generator is elliptic curve point multiplication (K.P) which is used by the transmitter and the receiver respectively to generate Q and the quantity ($u_1$G, $u_2$Q). Point multiplication is the basic operation of ECDSA, is computed with standard algorithm called binary algorithm by successive point additions and point doublings so that, when:

$$K = \sum_{i=0}^{l-1} k_i . 2^i \text{ , where } ki \in \{0,1\} \quad (2)$$

Point doublings are performed for all $k_i$ and point additions when $k_i$=1.

Point multiplication acts as the bottleneck in ECDSA for its optimization. There are many algorithms used for computing elliptic curve point multiplication [19]. The selection of the efficient computational technique of K.P provides a more reliable circuit of ECDSA for resource-constrained devices like wireless sensor nodes, smart cards and radio frequency identification (RFID) tags and avoids many problems such as high energy consumption and long delay [20]. In this context Koblitz curves ($E_k$) from a family of curves defined by Eq. (1) with a ∈ {0, 1} and b = 1 [21], allows to achieve this goal.

For this subfield curves point multiplication can be enhanced by avoidance the most used operation that is the point doublings implemented with 2 multiplications, 2 squarings, 1 inversion and 5 additions. Frobenius maps $\tau$operation defined over $E_k$ can replace the point doubling with only 2 squaring.

It is defined by:

$$\tau(\infty) = \infty;\ \tau(x, y) = \tau(x^2, y^2) \quad (3)$$

$$2.P = -\tau^2 P + \mu\tau P \text{ , where } \mu = 1 \text{ or } -1 \quad (4)$$

The scalar K for point multiplication is represented with τ-adic expansion (Eq.4). It can be obtained by repeatedly dividing k by τ, the digits $k_i$ are the remainders of the division steps. This procedure is analogous to the derivation of the binary representation of K by repeated division by 2.

$$K = \sum_{i=0}^{t-1\tau} k_i \tau^i \text{ , where } k_i \in \{-1,0,1\} \quad (5)$$

The point multiplication is computed by algorithm 1, with three computation primitives: adding (when $k_i$ = 1), subtracting (when $k_i$ = -1) and Frobenius map $\tau$.

**Algorithm1:**

**Input:** Point P, τ-adic expansion of K ($k_{t-1}$,….$k_0$).
**Output:** K.P
For i in 0 to ( t-1) loop
    if k(i) =1  then Q := Q + P
    elsif k(i) = -1  then Q := Q - P
    end if
    P := frobenius(P)
End loop

## C. *Hash function*

Designed by the National Security Agency and published by the NIST [22]. Hash functions are important security primitives used for authentication and data integrity. SHA allows obtaining the condensation of message in the sender and the receiver. They are used in conjunction with public-key algorithms for both encryption and digital signatures to check integrity and authentication. The three SHA algorithms are structured differently and are distinguished as SHA-0, SHA-1, and SHA-2. SHA-1 is very similar to SHA-0, but corrects an error in the original SHA hash specification that led to significant weaknesses. on the other hand SHA-2 significantly differs from the SHA-1 hash function. SHA-1 is better suited for implementations on constrained devices.

SHA-1 handles messages in blocks of 512 bits each of which requires computation of 80 steps. One step handles five 32-bit variables by computing four 32-bit modular additions (a + b mod 232) and certain 32-bit logical functions which depend on the step index. When all blocks have been processed, the hash of the message is in the five variables and thus SHA-1 outputs a 160-bit hash. The implementation consists of four main components; namely, step function, message schedule, constants block and control logic. The step function block determines the maximum clock cycle of the implementation and it was carefully optimized.

The four 32-bit additions form the critical path and all other operations (logic operations and rotations) are computed in parallel with these additions. The message schedule stores 512-bits message bits and derives a 32-bit word for each step from these 512 bits by using three 32-bit bitwise XORs and a rotation.

## V. HARDWARE IMPLEMENTATION RESULTS

This section gives the obtained results of our ECDSA hardware implementation and compares it to others to others related works [4][5][6]. The described circuits have been implemented in VHDL using ModelSim simulator SE 6.2a and synthesized by target device of Xilinx (Xilinx xc6vlx760-2ff1760 FPGA). The architecture was simulated for verification of the correct functionality, by using know-answer test vector that provided by NIST for Elliptic Curve Digital Signature Algorithm [2].

Fig. 7 shows the output results from signature generation in the transmitter. The signature is the set (rg, sg). Fig.8 shows the output results from signature verification, which gives that the signature is well verified (v = r).

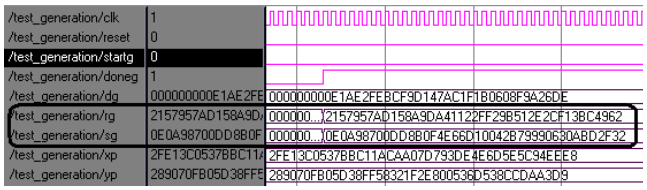


Figure 7. Final result of signature generation

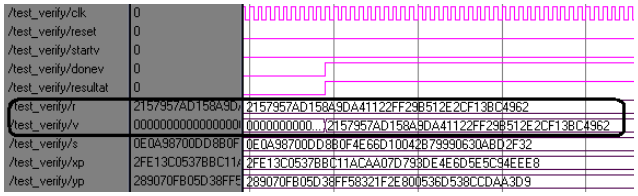


Figure 8. Final result of signature verification (v=r)

In table 1 we show the timing results and occupation of main blocks of ECDSA which are: W7, ECC and SHA-1.

Table 1. Implementations results of ECDSA modules.

| | Occupation (LUTs) | Fréqency (MHz) | Clock cycles | Time(µs) |
|---|---|---|---|---|
| W7 | 31705 | 400 | 96 | 0,2 |
| ECC | 2502 | 213 | 54138 | 254 |
| SHA-1 | 16084 | 253 | 482 | 1,9 |

W7 block takes 0.2 µs and 400 MHz Frequency with area occupation of 31705 slices LUTs. The frequency of SHA-1 is 253 MHz. It takes 1.819 µs with area occupation of 16084 slices LUTs. ECC is the most greedy operation time; this block takes 254 µs, 213 MHz frequency and occupation of 2502 slices LUTs.

Table 2, indicates the implementation results and the comparison of the proposed ECDSA against other works with different parameters; field type (binary (B, K), primary (P)) size key and method to compute ECC. ECDSA over Koblitz subfield curves with Frobenius maps to compute ECC is faster. For our version of ECDSA, the key generation block (key _gen) takes 0.2 ms and area occupation of 34215 slices LUTs (7%). The signature generation block (sign_gen) requires 0,8ms time and 64870 slices LUTs (13%). Time and occupation of signature verification (sign_verf) are respectively 0.4 ms, 45450 slices LUTs (5 %).

Table 2. Performance comparison for ECDSA FPGA implementations

| | Field Type-key size/ ECC method | Operations | Time(ms) | Occupation (Slices LUTs) | Devices |
|---|---|---|---|---|---|
| **[4]** | B-163/ Montgomery | key _gen<br>sign_gen<br>sign_verf | 0.6<br>0.94<br>1.61 | 85% | EP2C20F484C7 |
| **[5]** | P-192/ Montgomery | sign_gen | 127 | 19 115 gates | 0.35 µm CMOS technology (c35b4 AMS) |
| **[6]** | P-256/ Montgomery | sign_gen (KP)<br>sign_verf ($u_1$ P + $u_2$Q) | 7.15<br>9.09 | 14256 | Xilinx Virtex 5 |
| **This work** | K-163/Frobenius map | key _gen<br>sign_gen<br>sign_verf | 0,2<br>0,8<br>0.4 | 34215 (7%)<br>64870 (13%)<br>25450 ( 5%) | xc6vlx760-2ff1760 |

## VI. CONCLUSION

We have presented a complete, fast and secure version of ECDSA based on Koblitz curves and W7 to compute respectively the public and private key. We attained better performance compared with version based on general curves over binary and primary fields; the design requires 0.2 ms, 0.8 ms and 0.4 ms with 7 %, 13 % and 5 % of the device resources on Slice LUT for respectively key generation, signature generation and signature verification. The proposed ECDSA scheme is suitable to the application are needed to low-bandwidth communication, low-storage and low-computation environments, and particularly applicable to smart cards and wireless devices.

Our ECDSA design can be still more competitive in performance and security while introducing respectively more optimization at the level of operators for the point multiplication and using True Random Number Generator (TRNG) to generate private key, which is the objective of our future works.

## REFERENCES


[1] National Institute of Standards and Technology (NIST): Digital signature standard (DSS). Federal Information Processing Standard, FIPS PUB 186-2 (2000)

[2] NIST, "Recommended elliptic curves for federal government use," Tech. Rep., National Institute of Standards and Technology, U.S. Department of Commerce, 1999.

[3] N. Koblitz, “CM-Curves with Good Cryptographic Properties,” Advances in cryptology—Proc. 11th Ann. Int’l Cryptology Conf.,pp. 279-287, 1992.

[4] K. J¨arvinen and J. Skytt¨a, “Cryptoprocessor for Elliptic Curve Digital Signature algorithm (ECDSA),” Tech. Rep., Helsinki University of Technology, Signal Processing Laboratory, 2007.

[5] Michael Hutter, Martin Feldhofer, and Thomas Plos“An ECDSA Processor for RFID Authentication”RFIDSec 2010, LNCS 6370, pp. 189–202, 2010.

[6] Benjamin Glas, Oliver Sander, Vitali Stuckert, Klaus D.Muller-Glaser, and Jurgen Becker ” Prime Field ECDSA Signature Processing for Reconfigurable Embedded Systems” International Journal of Reconfigurable Computing Volume 2011

[7] Certicom Corp., SEC 1: Elliptic Curve Cryptography, published September 20, 2000

[8] M. Adleman, R. L. Rivest, and A. Shamir. “A Method for Obtaining digital Signatures and Public-key Cryptosystems.” Communications of the ACM, 21, pp. 120–126, 1978.

[9] Dr.R.Shanmugalakshmi and M.Prabu “Research Issues on Elliptic Curve Cryptography and Its applications” IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.6, June 2009.

[10] Koblitz, N. “Elliptic curve cryptosystems” Mathematics of Computation 48, 203–209 (1987).

[11] Miller, V.“Use of elliptic curves in cryptography”. In:Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)

[12] European Telecommunications Standards Institute (ETSI), Recommendation GSM 02’09, Security Aspects.

[13] G. Kostopoulos1, N. Sklavos2, M.D. Galanis3, and O. Koufopavlou “VLSI implementation of GSM Security: A5/1 and W7 Ciphers”.

[14] Kitsos P, Sklavos N, Papadomanolakis K, and Koufopavlou O, “Hardware Implementation of Bluetooth Security,” IEEE Pervasive Computing, vol. 2, no.1, pp. 21-29, January-March 2003.

[15] Dworkin M. “Recommendation for Block Cipher Modes of Operation, Methods and Techniques,” National Institute of Standards and Technology (NIST), Technology Administration, U.S. Department of Commerce, Special Publication, 2004.

[16] Ferguson N, Whiting D, Schneier B, Kelsey J, Lucks S, and Kohno T, “Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive,” Lecture Notes in Computer Science (LNCS), Springer-Verlag, Berlin, Germany, vol. 2887, pp. 330-346, 2003.

[17] Michalis Galanis, Paris Kitsos, Giorgos Kostopoulos, Nicolas Sklavos, Costas Goutis, “Comparison of the Hardware Implementation of Stream Ciphers,” The International Arab Journal of Information Technology, Vol.2, No.4, October 2005.

[18] S. Thomas, D. Anthony, T. Berson and G. Gong, “The W7 Stream Cipher Algorithm”, Internet Draft, April 2002.

[19] Gordon, D.M., “A survey of fast exponentiation methods," Journal of Algorithms, vol. 27, no. 1, Apr. 1998, pp. 129-146.

[20] L. Chen and G. Gong, Appendix B. “Design of Stream Ciphers,” Communication Systems Security, 2008.

[21] J. Solinas, “Efficient Arithmetic on Koblitz Curves,” Designs, Codes, and Cryptography, vol. 19, nos. 2-3, pp. 195-249, 2000.

NIST (2009), “Secure Hash Standard”, FIPS PUB180-3.