MALAYSIAN JOURNAL OF COMPUTING AND APPLIED MATHEMATICS (2021) 4 (1) 27-34

DOI: *https://doi.org/10.37231/myjcam.2021.4.1.72*

Article: Received 23.11.2020; Accepted 18.03.2021; Published 31.03.2021

EFFICIENT HARDWARE IMPLEMENTATION OF ELLIPTIC CURVE DIFFIE-HELLMAN KEY EXCHANGE PROTOCOL

SAOUDI Mohamed, KERMICH Akram, ZEBDA Abdelfatah, ALLAILOU Boufeldja

Department of Electronics ESACH, Algiers, Algeria.
saoudimohamed26@gmail.com

Abstract: The aim of the present work is the hardware implementation of the elliptic curve Diffie-Hellman (ECDH) key exchange protocol on a reconfigurable circuit of type FPGA at the register-transfer level (RTL). Compared to the standard Diffie- Hellman (DH), based on exponentiation in a finite field, ECDH is known to provide equivalent level of security with lower number of bits used. Reduced bit usage implies less power and logic area are required to implement this cryptographic scheme. This is particularly important in secure embedded system, where a high level of security is required, but with low power consumption. The results show that ECDH can be implemented on FPGA with convincing performances in comparison with other published works.

Keywords: *ECDH, Diffie-Hellman, FPGA, Register-Transfer level, Elliptic Curve*.

1. INTRODUCTION

Cryptography has become nowadays a vital tool to ensure the security of the vertiginous growth in the number of connected device via internet. Secret key cryptography is the most popular approach to ensure the confidentiality over a computer network. In fact, the majority of tools provided for this purpose (e.g. Secure Sockets Layer (SSL), Secure Shell (SSH), Ipsec, etc.) rely on the use of symmetric ciphering algorithm like Advanced Encryption standard (AES). This is due to the high speed and the reduced time of ciphering of such algorithm in comparison with their asymmetric counterparts.

The main drawback of the symmetric cryptography is key sharing. For instance, exchanging the key over a public channel would compromise the whole security of the cryptosystem. Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical channel, such as paper key lists transported by a trusted courier. In 1976, Whitfield Diffie and Martin Hellman published a paper [1] where they presented their scheme, named after them Diffie-Hellman (DH), for securely exchanging cryptographic keys over a public channel. Elliptic-curve Diffie-Hellman (ECDH) is an anonymous key agreement protocol that allows two parties, each having an elliptic-curve public-private key pair, to establish a shared secret over an insecure channel. It is a variant of the Diffie-Hellman protocol using elliptic-curve cryptography (ECC). It was first introduced in 1986 by Victor S. Miller [2].

The use of Field Programmable Gates Arrays (FPGA) for cryptographic applications is highly attractive, especially for embedded secure systems where high performances are required at low power consumption. Therefore, several FPGA- based efficient ECC hardware architectures and elliptic curve cryptographic processors have been presented in the literature ([3], [4], [5] and [6]). The aim of the presented work is the implementation, in an efficient way, of a high performance version of ECDH in the Xilinx Virtex 6 FPGA over the finite binary Galois Field GF (2^{163}).

In order to give a clear presentation of our work, this paper is structured as following; After an introduction, a brief review of the mathematical background of ECC is given. Then, we present the cryptographic scheme for the ECDH key exchange protocol. Then, After that we present the proposed architecture of ECDH to be implemented in FPGA. We terminate this paper by giving the results of the implementation by comparing it with existing implementations in literature.

2. RELATED WORKS

Several FPGA-based efficient ECC hardware architectures and elliptic curve cryptographic processors have been presented in the literature. In [7], Ghanmy proposed ECC processor over GF (2^{163}) on an FPGA platform for wireless sensor networks (WSN). Reaz's design [5] can perform ECC over GF (2^{131}) and GF (2^{163}) on Altera FPGAs. Hasan and Benaissa [6] implemented their ECC processor using the μ- coding technique on Xilinx Spartan-3 FPGAs over GF (2^{131}), GF (2^{163}), GF (2^{283}) and GF (2^{571}). An ASIC implementa- tion of an elliptic curve crypto-processor over GF (2^{163}) is presented in [8], where they used an ASIC CMOS 45 nm technology as a hardware platform. Shieh [9], Park et al. [10] also proposed their ECC processor over a binary field using Xilinx FPGAs.

3. ELLIPTIC CURVE ALGEBRA

In a nutshell, an elliptic curve is a cubic bi-dimensional curve defined by the following relation between the x and y coordinates of any point on the curve:

$$y^2 + xy = x^3 + ax + b \qquad (1)$$

Where a and b are arbitrary parameters that define the specific curve used. For a chosen pair (a, b) we can define a group structure on it. To do so we define an internal composition rule which satisfies the following three proprieties: Associativity, Identity and Inverse [11]. Even more, if we define a second composition rule over the aforementioned group having the same proprieties as the first composition rule, we get an algebric structure called Field [11]. More precisely, elliptic curves are defined over a finite field called Galois Field. A Galois field denoted normally as GF ($q = p^m$) is said to be a binary field or characteristic-two finite field if $q = 2^m$.

A non-supersingular elliptic curve E over GF (2^m) in affine coordinates is the set of solutions to the equation 1 where x,y,a,b GF (2^m), b = 0. The coefficients a,b specifying an elliptic curve are typically defined by the NIST standard.

The two essential arithmetic procedures defined on the finite field of elliptic curves GF (2^m) are: the Point Addition (PA) and the Point Doubling (PD). For a given two points P = (x1, y1) and Q = (x2, y2) their PA R can be found by:

$$R(x3, y3) = P(x1, y1) + Q(x2, y2) \qquad (2)$$
$$x3 = \lambda^2 + \lambda + x1 + x2 + a \qquad (3)$$
$$y3 = \lambda(x1 + x3) + x3 + y1 \qquad (4)$$

where $\lambda = (y2 + y1)/(x2 + x1)$. For PD we use:

$$R(x3, y3) = 2P(x1, y1) \qquad (5)$$
$$x3 = \lambda^2 + \lambda + a = x1^2 + b/\, x1^2 \qquad (6)$$
$$y3 = x1^2 + \lambda x3 + x3 \qquad (7)$$

where $\lambda = (x1 + y1/x1)$.

4. ELLIPTIC CURVE DIFFIE-HELLMAN

The ECDH cryptographic scheme is shown in figure 1 and is given below:

1) Before starting a communication, Alice and Bob have toget agree, in the public channel, on the parameters of theelliptic curve EC and a point *P* on this curve, i.e: the coefficients *a* and *b* from equation 1, the characteristic polynomial of the field *GF* (2^m) and the coordinates (*x, y*) of the point *P* ;
2) Alice generates a secret random private secret k_A;
3) Then she computes $k_A{}^{pu} = k_A \; X \; P$, where denotes the scalar multiplication in *GF* (2^m) which could be achieved by using the two arithmetic procedures *PA* and*PD* described in the section III.
4) Bob executes the same actions 1 and 2 to get $K_B{}^{pu}$;

5) At this point, Alice and Bob exchange with each other K_A^{pu} and K_B^{pu};
6) Alice and Bob can now compute the Secret Symetric Key $k = k_B * k = k_B * k_A * P$.

The core of ECDH is the scalar multiplication, which computes $\alpha * P$ using only the arithmetic procedures *PA* and *PD*, for example:

$$7 * P = (2P((2P) + P)) + P \tag{8}$$

For a given $Q = \alpha * P$, the problem of calculating α from the points *P* and *Q* is called the discrete logarithm problem over the elliptic curve (ECDLP) which is the hard problem underpinning elliptic curve cryptography. Despite almost three decades of research, mathematicians still haven't found an algorithm to solve this problem that improves upon

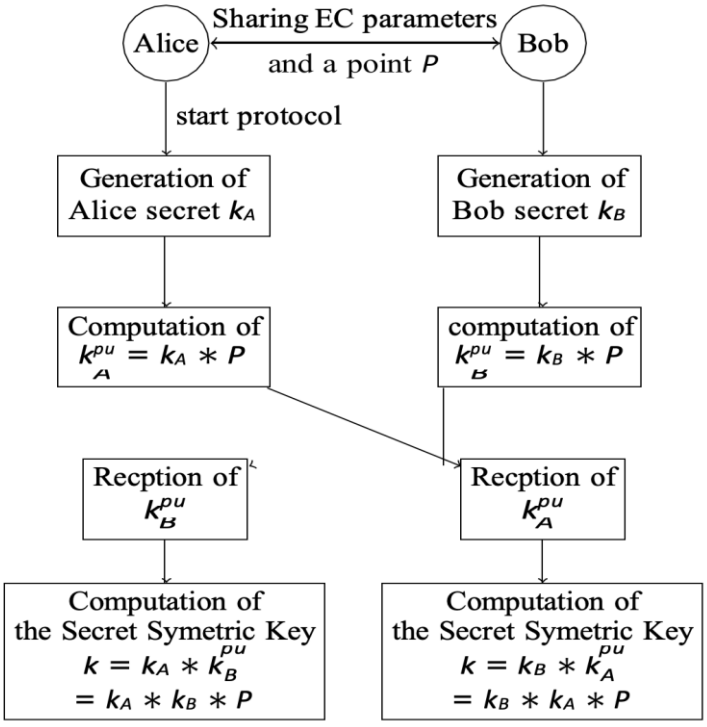

Fig. 1. ECDH key exchange protocol

the naive approach. In other words, unlike with factoring (Classical DH), based in currently understood mathematics, there doesn't appear to be a shortcut that will help to find α in a reduced time. This means that for numbers of the same size, solving ECDLP is significantly harder than factoring. Since a more computationally intensive hard problem means a stronger cryptographic system, it follows that elliptic curve cryptograms are harder to break than the ones based on modular exponentiation like RSA and DH [12].

In 2000, FIPS-2 was recommended with 10 finite fields: 5 prime fields, and 5 binary fields. The binary fields are GF (2^{163}), GF (2^{233}), GF (2^{283}), GF (2^{409}) and GF (2^{571}) [13]. Prime fields GF (p) and binary fields GF (2^m) of similar size are considered to provide almost the same level of security [14]. Table I compares symmetric cipher key length, and key lengths for PKC such as RSA, Diffie-Hellman (DH), and ECC (both prime and binary fields). It demonstrates that smaller field sizes can be used in ECC than in RSA and DH systems at a given security level. ECC is many times more efficient than RSA and DH for either private-key operations (such as signature generation and decryption) or public-key operations (such as signature verification and encryption).

5. PROPOSED ARCHITECTURE

As mentioned before, The core of the ECDH is the scalar multiplication (figure 2), therefore, a high importance is given to the proposed FPGA architecture implementing this module. The scalar multiplication module contains two main components. Each one of them ensure either the PA or the PD procedures. From equations 3, 4, 6 and 7, we can see.

TABLE I COMPARISON OF KEY LENGTH FOR EQUIVALENT SECURITY OF SYMMETRIC-KEY AND PUBLIC-KEY CRYPTOGRAPHY [15]

Summytric Key	Example Alg	RSA and DH	GF(p)	ECC in GF(2m)
112	Triple-DES	2048	224	233
128	AES Small	3072	256	283
192	AES Medium	8192	384	409
256	AES Large	15360	521	571

that the operations needed to implement PA and PD are : addition, multiplication, squaring and division. It is known that the addition in GF (2m) is equivalent to a simple xor in either hardware or software. For the remaining operations, this section gives in details the algorithms and methods used to implement them. It is useful to mention that all of the operations are executed using the polynomial representation of elements in GF (2m).

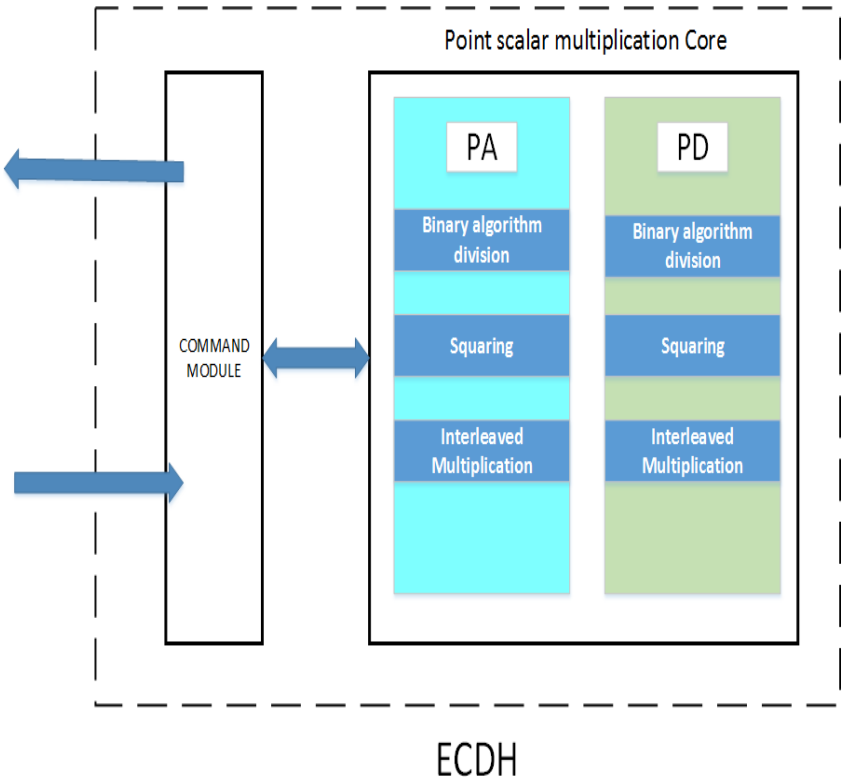

Fig. 2. proposed hardware architecture for ECDH

A. Multiplication in GF (2m)

Multiplication in GF (2m) with the interleaved modular reduction algorithm is a well-known algorithm for hardware implementation [16]. It computes the product of two poly- nomials then applies modular reduction, and its operation is different from simple integer multiplication. The algorithm 1 describes in details the interleaved modular reduction.

B. Squaring in GF (2m)

Squaring in GF (2m) has less computation complexity than polynomial multiplication because it can be achieved by setting a 0 bit between consecutive bits of the operand, as shown in figure 3.

C. Division in GF (2m)

Division in GF (2m) is the most expensive operation for implementing ECC over a binary field. The quotient of two polynomials in GF (2m) can be computed using the binary version of the binary algorithm that is used for calculation of

Algorithm 1 Multiplication in $GF(2^m)$ with interleaved mod- ular reduction

Input : $P(x)$, $Q(x)$ and $f(x)$ $GF(2^m)$ $\in$
Output : $R(x) = P(x)Q(x) \bmod f(x)$ **Initialization :** $R_v = 0$; $P_v = '0' \& P(x)$ **for** $i = (m\ 1)$ **to** 0 **do**
  if $Q(i) = '1'$ **then** $-$
    $R_v = R_v x \quad Q_v$; $\oplus$
  else
    $R_v = R_v x$;
  end if
  if $R_v(m) = '1'$ **then**
    $R_v = R_v \quad f(x)$; $\oplus$
  end ifend for
Return $R(x)$;

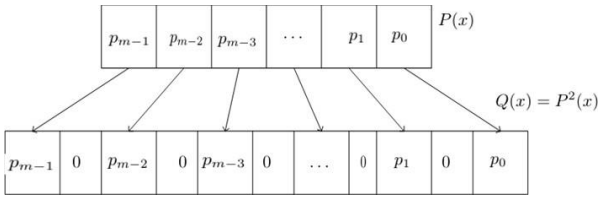

Fig. 3. Squaring a binary polynomial P (x)

the great common divider (GCD) for polynomials. The binary algorithm for computing R(x) = P (x)Q−1(x) mod f (x) is described in algorithm 2.

6. FPGA IMPLEMENTATION RESULTS AND PERFORMANCE ANALYSIS

This section presents the hardware implementation results of the proposed architecture. We have implemented and tested our design on a modern Xilinx Virtex-6 (XC6VLX240T) FPGA. All VHDL modules are extensively simulated using both Isim and ModelSim, and synthesized using Xilinx ISE 14.7 synthesis technologies. The parameters for the elliptic curve used are taken from the NIST standard and are given in table II. We choose to work with the irreducible polynomial f (x) = $x^{163} + x^7 + x^6 + x^3 + 1$ over the field GF (2^{163}).

TABLE II NIST-RECOMMENDED ELLIPTIC CURVES OVER GF (2^{163}) [13]

K-163: m = 163, f (x) = x163 + x7 + x6 + x3 + 1, a = b = 1, h = 2					
n=0x 4	00000000	00000000	00020108	A2E0CC0D	99F8A5EF
x=0x 2	FE13C053	7BBC11AC	AA07D793	DE4E6D5E	5C94EEE8
y=0x 2	89070FB0	5D38FF58	321F2E80	0536D538	CCDAA3D9

A. Implementation of PA and PD

To implement the PA core we needed to infer a module for multiplication, a module for squaring and another one for division, in addition to a Finite State Machine (FSM)

Algorithm 2 Binary algorithm for polynomials division in GF (2^m)

Input : P (x), Q(x) and f (x) GF (2^m)
Output : R(x) = P (x)Q−1(x) mod f (x)
Initialization : a = f , b = Qv, c = 0, d = Pv, α = m
and β = m 1
while β > 0 do
if b(0) = 0 then
b = b << 1;
d = d/x mod (f);
β = β 1;
else
old β = β;
b = a b << 1;
d = (c d)/x mod (f);
if α > β then
β = α 1; α = old β; c = d;
else
β = β 1;
end if end if
end while
R(x) = c;
Return R(x);

for the control of the whole operation. The same modules were inferred to implement the PD core but with a different FSM. Result of implementation are given in table III. Useful to notice that the time row in the table correspond to the time needed to complete the PA or PD procedure at a frequency of 200 Mhz.

TABLE III RESOURCES UTILIZATION AND PERFORMANCES OF PA AND PD IMPLEMENTATION

	PA	PD
Slice registers	4855 (1%)	4367(1%)
Slice LUT	2817 (1%)	2661 (1%)
Max frequency (Mhz)	325	325
Time (µs)	2.54	3.38

B. Implementation of the scalar multiplication core

Using PA and PD module, we have succeeded to implement the scalar multiplication core. The proposed design contains a PA module and PD module which are controlled by an FSM. Table IV summarizes the results of the scalar multiplication core implementation in comparison with a recent work [11], where the authors implemented another architecture for the scalar multiplication module over Binary Field GF (2^{163}).

Worth to mention that the FPGA used in [11] is more modern and possibly offers a higher work frequency. Nevertheless, as shown in Table IV our design takes less time to compute the scalar multiplication

TABLE IV SCALAR MULTIPLICATION RESOURCES UTILIZATION AND PERFORMANCES : COMPARAISON BETWEEN OUR WORK AND [11]

	Our work	Hossain et al. [11]
FPGA	Virtex-6	Kintex-7
Slice registers	11217 (3%)	6620(1%)
Slice LUT	6560 (4%)	7963 (3%)
Frequency (Mhz)	200	306.48
Time (µs)	766	1060

C. Implementation of ECDH

With scalar multiplication core ready to use, the implementation of ECDH only needed an appropriate FSM. Therefore the resources utilization is approximately equal to the one shown in table IV.

7. CONCLUSIONS

An efficient implementation of ECDH in FPGA has been presented in this work. A high-performance cores for computing the PA and PD procedures over GF (2^{163}) were designed. The implementation results have shown that our proposed architecture present two main advantages : a low resource utilization which makes it ideal for embedded system; and reduced time of calculation which makes this solution a good candidate for hardware acceleration of various internet security.

References

[1] W. Diffie, M. Hellman, “New Directions in Cryptography,” IEEE trans. on Information Theory, pp. 644–654, November 1976.
[2] V. S. Miller, “Use of Elliptic Curves in Cryptography,” Advances in Cryptology – CRYPTO’85, pp. 417–426, 1986.
[3] G. Sutter, J. Deschamps, J. Imana, “ Efficient Elliptic Curve Point Multiplication Using Digit-Serial Binary Field Operations,” IEEE Trans. on Industrial Electronics, pp. 217–225, 2013.
[4] W. Chelton, M. Benaissa, “Fast Elliptic Curve Cryptography on FPGA,” IEEE Trans. on Very Large Scale Integration (VLSI) Systems, pp. 198– 205, 2008.
[5] M. B. I. Reaz, J. Jalil, H. Husain, F.H. Hasim, “FPGA implementation of elliptic curve cryptography engine for personal communication systems,” Tran. on Circuits and Systems, pp. 82–91, 2012.
[6] M. Hassan, M. Benaissa, “Efficient time-area scalable ECC processor using µ-coding technique,”Third International Workshop, WAIFI, Arith- metic of Finite Fields, pp. 250—268, 2010.
[7] N. Ghanmy, L. C. Fourati, L. Kamoun, “Elliptic curve cryptography for WSN and SPA attacks method for energy evaluation,” Journal of Networks, pp. 2943–2950, 2014.
[8] M. Machhout, Z. Guitouni, K. Torki, L. Khriji, R. Tourki, “Coupled FPGA/ASIC implementation of elliptic curve crypto-processor,” Interna- tional Journal of Network Security its Applications (IJNSA), pp. 100— 112, 2010.
[9] M.D. Shieh, J.H. Chen, W.C. Lin, C.M. Wu, “An efficient multi- plier/divider design for elliptic curve cryptosystem over GF (2m),” Journal of Information Science and Engineering, pp. 1555—1553, 2009.
[10] J. Park, J.T. Hwang, “FPGA and ASIC implementation of ECC processor for security on medical embedded system,” The Third International Conference on Information Technology and Applications (ICITA’05), pp. 547—551, Washington, DC, USA, 2005. Computer Society
[11] M. S. Hossain, E. Saeedi, Y. Kong, ” High-performance FPGA Imple- mentation of Elliptic Curve Cryptography Processor over Binary Field GF (2163),” In Proceedings of the 2nd International Conference On Information Systems Security and Privacy, pp 415–422, 2016.

[12] M. Amara, A. Siad, “Elliptic Curve Cryptography And Its Applications,” 7th International Workshop on Systems, Signal Processing and their Applications (WOSSPA), Tipaza, Algeria, 2011.
[13] NIST-National Institute of Standards and Technology, Digital Signature Standard, FIPS Publication 186-2, 2000.
[14] N. Koblitz, A. Menezes, S. Vanstone, “The state of elliptic curve cryptography,” Des. Codes Cryptography, pp. 173–193, 1987.
[15] D. Hankerson, A. Menezes, S. Vanstone, “ Guide to Elliptic Curve Cryptography,” Springer-Verlag New York, Inc., Secarus, NJ, USA, 2003.
[16] J. Wolkerstorfer, “ Dual-field arithmetic unit for GF (p) and GF (2m),” CHES, Lecture Notes in Computer Science, pp. 500–514, 2002.