Elliptic Curve Diffie-Hellman Protocol Implementation Using Picoblaze

Makhamisa Senekane†, Sehlabaka Qhobosheane††, and B.M. Taele†††

†Department of Electronics and Information Technology, iThemba LABS, Faure 7129, South Africa
††Department of Electrical and Electronic Engineering, Stellenbosch University, Matieland 7602, South Africa
†††Department of Physics and Electronics, National University of Lesotho, Roma 180, Lesotho

Summary
Compared to other public key cryptography counterparts like Diffie-Hellman (DH) and Rivest Shamir Adleman (RSA), Elliptic Curve Cryptography (ECC) 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 wireless networks, where a high level of security is required, but with low power consumption. This paper presents the implementation of Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol over GF (2^{163}). The implementation is targeted to Spartan 3AN Field Programmable Gate Array (FPGA) from Xilinx. The results show that ECDH scalar multiplication can be computed in 1.34 milliseconds, using 4725 of 5888 FPGA slices available in Spartan 3AN. These results confirm the utility of Picoblaze in Elliptic Curve Cryptography.

Key words:
Diffie-Hellman, Elliptic Curve Cryptography, Field Programmable Gate Array, Galois Field, Picoblaze

1. Introduction

In this information age, the need to securely transmit information over an insecure channel is more pronounced than ever. Cryptography is the science of securely transmitting and retrieving information using an insecure channel [1], [2]. It involves two processes, namely encryption and decryption. Encryption is the process in which the sender (normally called Alice) transforms information into an unintelligible string of characters (known as ciphertext) for transmission over the transmission channel, so that an eavesdropper (normally called Eve) could not know the information. Decryption is a reverse of encryption; where the receiver (normally called Bob) transforms Alice's ciphertext into an intelligible message (known as plaintext).

In modern cryptography, the mode of distributing a key among communicating parties plays a critical role. Key distribution can be achieved either through private key cryptography or through public key cryptography [3], [4]. In private key cryptography (also known as symmetric key cryptography), a single key is used for both encryption and decryption of the message, while on the other hand, public key cryptography (also called asymmetric key cryptography) uses a pair of keys, one for encryption, and the other for decryption. Additionally, public key cryptography relies on the existence of one-way function [1], [4].

Diffie-Helmann (DH) key exchange protocol is the first public key cryptography scheme, and it was proposed by Witfield Diffie and Martin Hellman in 1976 [5]. This protocol uses a pair of keys (secret and private keys), since it is a public key cryptographic scheme. If Alice wants to communicate with Bob, she encrypts her message with her private key and Bob's public key. On the receiving end, Bob decrypts the message using his private key and Alice's public key [1]. DH key exchange protocol is based on the difficulty of computing logarithmic functions of prime exponents, and this is known as Discrete Logarithm Problem (DLP) [6].

Elliptic Curve Cryptography (ECC) was independently proposed by koblitz [7] and Miller [8] in the late 1980s. ECC is a public key cryptographic scheme that uses the properties of Elliptic Curves in mathematics to develop cryptographic algorithms. Security of ECC is based on the intractability of Elliptic Curve Discrete Logarithm Problem (ECDLP) [9]. Elliptic Curve Cryptography is defined by Elliptic Curve domain parameters given by:

$$T = (q, FR, a, b, c, G, n, h) \quad (1)$$

Where; q: the prime p or 2^m that defines the curve's form, FR: the field representation, a, b: curve coefficients, G: the base point (G_x, G_y), n: the order of G, which must be a large prime, and h: the cofactor co-efficient.

A soft-core is a flexible Hardware Description Language (HDL) architecture of a specific processor that can be customized for a given application and be synthesized for either Application Specific Integrated Circuit (ASIC) or Field Programmable Gate Array (FPGA) target [10], [11]. Picoblaze is a compact 8-bit microcontroller with Harvard Reduced Instruction Set Computer (RISC) architecture. It is a Xilinx product and is optimized for the Spartan-3, Virtex II and Virtex II Pro families of FPGAs [10].

The remainder of this paper is divided into five sections. Section 2 provides a background information concerning Elliptic Curve Cryptography (ECC), Elliptic Curve

Manuscript received June 5, 2011
Manuscript revised June 20, 2011

Diffie-Hellman (ECDH) and picoblaze soft-core. Section 3 discusses the related work of this research area. In Section 4, implementation of ECDH in picoblaze soft-core is discussed. This is followed by a section which provides results obtained and analysis of such results. Finally, conclusions are drawn, and future work is recommended, in Section 6.

2. Background Information

2.1 Elliptic Curve Cryptography

Ever since its invention in the 1980s, ECC has gained a huge popularity due to the fact that compared to other public key cryptography systems, it (ECC) can provide same level of security with smaller key sizes [12], [13], [14], [15], [16]. Thus, compared to their public key cryptography counterparts, Elliptic Curve Cryptosystems are computationally more efficient, and offer better security with smaller key sizes [12]. This makes ECC an appropriate cryptographic scheme for constrained environments such as smart cards and wireless networks, where power, processing time and memory resources are limited. Additional details on the theory behind ECC can be found in [7], [8], [17], [18], [19], [20] and [21].
ECC relies on efficient algorithms for finite field arithmetic operations such as inversion, multiplication and addition [12]. This public key cryptography scheme can be defined over two popularly used fields, namely; prime Galois Field, GF (p), or over binary extension Galois Field, GF (2^m). In GF (p), the equation of Elliptic Curve is given by:

$$Y^2 \bmod p = x^3 + ax + b \bmod p \qquad (2)$$

Where:

$$4a^3 + 27b^2 \bmod p \neq 0 \qquad (3)$$

with elements of GF (p) as integers between 0 and p-1 [23].

In GF (2^m), the equation of Elliptic Curve is given by:

$$y^2 + xy = x^2 + ax^2 + b \qquad (4)$$

where:

$$b \neq 0 \qquad (5)$$

Over GF (2^m), algebraic rules for point addition and point doubling could be implemented [7], [8], [13], [14], [15], [23].

2.2 Elliptic Curve Diffie-Hellman Key Exchange Protocol

Elliptic Curve Diffie-Hellman (ECDH) is a public key agreement protocol which allows two parties; Alice and Bob, to establish a shared secret key for use in symmetric key algorithms [23]. It enables the implementation of Diffie-Hellman key exchange algorithm using a group of points on an Elliptic Curve over a Galois Field GF (2^m) [14], [24], [25]. In order to generate a shared key between Alice and Bob using ECDH key exchange protocol, both Alice and Bob should agree beforehand to use the same Elliptic Curve domain parameters [23], [25]. An algorithm for computing a shared key using ECDH is given below:

- Alice computes key $k = (x_K, y_K) = d_{Alice} * Q_{Bob}$, where d_{Alice} is Alice's private key and Q_{Bob} is Bob's public key
- Bob computes key $l = (x_L, y_L) = d_{Bob} * Q_{Alice}$, where d_{Bob} is Bob's private key, and Q_{Alice} is Alice's public key
- Since $d_{Alice} * Q_{Bob} = d_{Alice}\, d_{Bob}\, G = d_{Bob}\, d_{Alice}\, G = d_{Bob} * Q_{Alice}$, then $k = l$, hence $x_K = x_L$
- Hence the shared key is x_K .

2.3 Picoblaze

Picoblaze is a compact 8-bit soft-core for Xilinx FPGA devices, which is provided as a free cell-level HDL and can be synthesized along with other logic [26], [27]. It is optimized for efficiency and low deployment cost [28], [29]. Picoblaze has the following features [30]: *8-bit data width; 8-bit Arithmetic Logic Unit (ALU) with carry and zero flags; 16 8-bit general purpose registers; 64-byte data memory; 18-bit instruction width; 10-bit instruction address, which supports a program of up to 1024 instructions; 31-word call/return stack;* 256 input and output ports*; 2 clock cycles per instruction; and 5 clock cycles for interrupt handling.*

Picoblaze processor is shown in **Fig. 1**, while its top-level design is shown in **Fig. 2**.

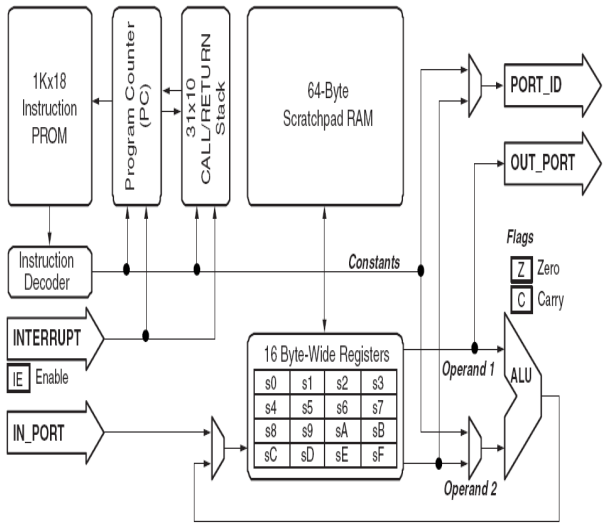

Fig. 1: Picoblaze processor

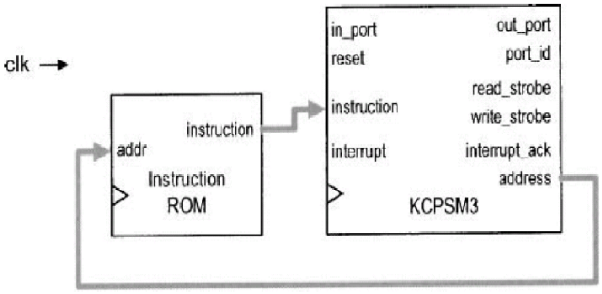

Fig. 2: Picoblaze's top-level diagram

3.0 Related Work

In ECC, only algorithms based on software-based processors or reconfigurable logic are implemented. Algorithms based on discrete circuitry are never implemented. This is due to the fact that designs based on discrete components are inflexible; and so this presents a huge problem for their applicability in cryptographic algorithms. In [19], [24], [25], [31] and [32], software implementation of ECC over extension binary field is discussed. Implementation of ECC over binary field using reconfigurable logic is discussed in [33], [34], [35], [36], [37] and [38]. However, of the references provided above, it is only in [37] and [38] where picoblaze was used to realize ECC algorithms. Furthermore, even though [37] and [38] use picoblaze soft-core, none of these two papers use picoblaze to compute scalar multiplications in ECDH. The contribution of this paper is the implementation of Elliptic Curve Diffie-Hellman key exchange protocol using Picoblaze soft-core processor.

4.0 Implementation

GF (2^{163}) was implemented for this research project. Elliptic Curve domain parameters were chosen based on the National Institute of Standards and Technology's (NIST's) recommendations [39], [40] for Koblitz curves such that:

- m, degree of a polynomial = 163
- a and b = 1
- cofactor coefficient, h = 1 and
- irreducible polynomial $p(x) = x^{163} + x^7 + x^6 + 1$.

For scalar multiplication k*G, an algorithm described in [18] and [33] was used. This algorithm was then implemented using picoblaze, which has a word length of 8-bits. Since GF (2^{163}) uses 163-bits, then these 163 bits were divided into 21 8-bit words, with zeroes being appended on the most significant byte. Assembly language was used (using picoblaze soft-core) to compute scalar multiplication. This was followed by top-level Very High Speed Integrated Circuit HDL (VHDL) design, where picoblaze core was initiated and instantiated. Also, in a top-level VHDL module, k*G mod p(x) was computed, using the outputs from picoblaze microcontroller. Finally, the design was targeted to Spartan 3AN FPGA.

Spartan 3AN FPGA is shown in **Fig. 3**.

Fig. 3: Spartan 3AN development board from Xilinx [41].

5.0 Results

The results for the paper are summarized in Table 1. Additionally, it took 1.34 ms to perform 163-bit scalar multiplication. These results are promising because from them, it could be observed that ECC algorithms could be implemented in relatively higher-performance

reconfigurable logic (compared to software-based processors) using easy programming language (as opposed to verbose HDLs).

Table 1: Device utilization summary

Logic Utilization	Used	Available	Utilization (%)
Slices	4725	5888	81
Slice flip-flops	6100	11776	52
4-input LUTS	6175	11776	53
Bonded IOBS	201	372	54
Number of BRAMS	1	20	5
Number of Gclks	1	20	4

Also, from the results, it can be observed that scalar multiplication is very costly in so far as computational resources are concerned. This is readily obvious on an 8-bit picoblaze microcontroller. This shows a need for enhancement of scalar multiplication algorithms, so they could be implemented in small cores.

6.0 Conclusions

In this paper, a 163-bit Elliptic Curve Diffie-Hellman key distribution protocol was implemented, and targeted to Spartan 3AN FPGA. The designed cryptosystem performed better than ECC cryptosystems using software-based processors. Even though this cryptosystem provided interesting results, a lot still needs to be done in order to create efficient Elliptic Curve scalar multiplication algorithms that could easily be implemented in small soft-cores.

References

[1] N. Ferguson, B. Schneier and T. Konho. Cryptography Engineering: Design Principles and Practical Applications.

[2] V. Makarov, "Quantum cryptography and cryptanalysis". PhD Thesis. Norwergian University of Science and Technology. 2007.

[3] S. Singh. The Codebook. Fourth Estate, London, 2000.

[4] W. Stallings. Cryptography and Network Security: Principles and Practice. (3rd ed.). Prentice Hall, Upper Saddle River, New Jersey, 2003.

[5] W. Diffie and M. E. Hellman. "New directions in cryptography". IEEE Transactions on Infornation Theory, Vol. 22, No. 6, pp 644-654, 1976.

[6] C. K. Koc. (ed.). Cryptographic Engineering. Springer, New York, NY, 2009.

[7] N. Koblitz. "Elliptic Curve Cryptosystems". Mathematics of Computation, Vol. 48, pp 203-209, 1987.

[8] V. Miller. "Uses of Elliptic Curves in cryptography". CRYPT'85, LNCS 218, pp 417-426, 1986.

[9] P. Bulens, G. M. de Dormale and J. J. Quisqauter. "Hardware for collision search on Elliptic Curve over GF (2m)". SHARCS, Ecrypt Workshop, 2006.

[10] J. G. Tong, I. D. L. Anderson and M. A. S. Khalid. "Soft-core processors for embedded systems". International Conference on Microelectronics, pp 170-173, 2006.

[11] D. Curry, A. Hofler, H. Dong, T. Allison, C. Hovater and K. Mahoney. "Implementation of an EPICS IOC on an embedded soft core processor using field programmable gate arrays". International Conference on Accelerator and Large Experimental Physics Control Systems, pp 0.0551-4, 2005.

[12] S. Backtir. "Frequency domain finite field arithmetic for elliptic curve cryptography". PhD Thesis. Worcester Polytechnic Institute. 2008.

[13] D. Hankerson, A. Menezes and S. Vanstone. Guide to Elliptic Curve Cryptography. Springer-Verlag, New York, NY, 2004.

[14] E. de Win and B. Prenel. "Elliptic curve public key cryptosystems – an introduction". LNCS 1528, pp 131-141, 1998.

[15] C. Koppensteiner. "Mathematical foundations of elliptic curve cryptography". PhD Thesis. Vienna University of Technology. 2009.

[16] N. Gura, A. Patel, A. Wander, H. Eberle and C. Shantz. "Comparing elliptic curve cryptography and RSA on 8-bit CPUs". Proceedings of CHES 04, pp 119-132, 2004.

[17] B. Kaliski, M. Liskov and Y. L. Yin. "Efficient finite field basis conversion techniques". Proposal for Inclusion in IEEE P1363, 1999.

[18] Y. Han, P. C. Leong, P. C. Tan and J. Zhang. "Fast algorithms for elliptic curve cryptosystems over binary finite field". In K. Y. Lam and E. Okamoto (eds.). Advances in cryptology – ASIACRYPT'99, Vol. 1716 of Lecture Notes in Computer Science, pp 75-85, Springer-Verlag, 1999.

[19] A. M. Fiskiran and R. B. Lee. "Workload characterization of elliptic curve cryptography and other network security algorithms for constrained environments". IEEE International Workshop on WWC-5, 2002.

[20] A. A. Gatub. "Preference of efficient architectures for GF (p) elliptic curve crypto operations using multiple parallel multipliers". International Journal of Security, Vol. 4, No. 4, pp 46-63, 2010.

[21] J. Z. Shi and H. Yan. "Software implementation of elliptic curve cryptography". International Journal of Network Security, Vol. 7, No. 1, pp 141-150, 2008.

[22] A. A. Gautub. "Area flexible GF (2k) elliptic curve cryptography". International Arab Journal of Information Technology, Vol. 4, No. 1, pp 1-10, 2007.

[23] M. S. Anoop. "Elliptic curve cryptography – an implementation tutorial".Technical Report, Tata Elxsi Ltd, 2007.

[24] R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck. "Fast key exchange with elliptic curve systems". In D. Coppersmith (ed.). Advances in Cryptography – CRYPTO'95, Vol. LNCS 963 ,pp 43-56, Springer-Verlag, Berlin, 1995.

[25] S. Kumar, et al. "Embedded end-to-end wireless security with ECDH key exchange". In Proceedings of 46th IEEE International Midwest Symposium on Circuits and Systems, 2003.

[26] P. Chu. FPGA Prototyping by VHDL Example. John Wiley and Sons, Hoboken, New Jersey, 2008.

[27] M. Senekane. "Design and Implementation of reconfigurable VME exerciser". Masters Thesis. University of Cape Town. 2010.
[28] K. Chapman. "Picoblaze 8-bit embedded microcontroller user guide". Technical Report. Xilinx Corporation. 2008.
[29] P. Yu and P. Schaumont. "Executing hardware as parallel software for picoblaze networks". FPL 2006, pp 1-6, 2006.
[30] M. Senekane. "Implementation of picoblaze-based VMEbus exerciser". Unpublished paper.
[31] D. Hankerson, J. L. Hernandez and A. Menezes. "Software implementation of elliptic curve cryptography over binary field". 2000. Retrieved from: http://citeseer.ist.psu.edu/brown01.html. On: 20/05/2011.
[32] H. Eberle, A. Wander, N. Gura and S. Chang-Shantz. "Architectural extensions for elliptic curve curve cryptography over GF (2m). Proceedings of the 16th IEEE International Conference on Application-Specific Systems, Architectures and Processors, 2005.
[33] S. kumar and C. Paar. "Reconfigurable instruction set extension for enabling ECC on 8-bit processor". International Conference on Field Programmable Logic and Applications, Antwerp, Belguim, 2004.
[34] J. Riley and M. J. Schulte. "A hardware accelerator for elliptic curve cryptography over GF (2m)". 2008. Rettrieved from: www.citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.79.9975. On: 20/05/2011.
[35] T. Kerins, E. M. Popovici and W. P. Mamane. "An FPGA implementation of a flexible secure elliptic curve cryptography processor". Proceedings of ARC 2005, pp 22-30, 2005.
[36] T. Kerins, E. M. Popovici , W. P. Mamane and P. FitzPatrick. "Fullt parametrizable elliptic curve cryptography over GF (2m)". Proceedings of MIEL 2004, Vol. 2, Nis, Yugoslavia, pp 739-742, 2004.
[37] X. Guo and P. Schaumont. "Optimizing HW/SW boundary for ECC SOC design using control hierarchy and distributed storage". Design, Automation and Test in Europe, 2009.
[38] X. Guo and P. Schaumont. "Optimized system-on-chip integration of a programmable ECC Coprocessor". ACM TRETS, Vol. 4, No. 1, pp 6:1-6:21, 2010.
[39] E. Barker, D. Johnson and M. Smid. "Recommendations for pair-wise key establishment using discrete logarithm cryptography". NIST Special Publication, 800-56A. 2007.
[40] NIST. "Recommendations for elliptic curves for federal government use". NIST, 1999.
[41] Xilinx Spartan 3AN User Guide. Technical Report. Xilinx Corporation, 2009.

Makhamisa Senekane received the B.Eng. degree in Electronics and MSc.Eng degree in Electrical Engineering from National University of Lesotho and University of Cape Town in 2007 and 2011 respectively. He now plans on pursuing a PhD in Quantum Cryptography at the University of Kwazulu Natal in South Africa.

Sehlabaka Qhobosheane received the B.Eng. degree in Computer Systems and Networks Engineering from National University of Lesotho (NUL) in 2009. He is now completing his thesis in MSc.Eng Biomedical Electronics at Stellenbosch University and is expected to graduate in December 2011. He currently works as a Software Engineer at iThemba LABS Medical Radiation in Cape Town South Africa.

B.M. Taele received the B.Sc. degree from the National University of Lesotho in 1991. He received the M.Sc. and PhD. Degrees from Lancaster University (UK) in 1994 and 2000 respectively. After working as a lecturer (from 2000) in the Department of Physics and Electronics at National University of Lesotho, he has been a Senior Lecturer in Applied Physics and Electronics at the National University of Lesotho with teaching and research interests in Semiconductor device modeling and Applied Photovoltaics. He is a regular associate of the ICTP and TWAS young Affiliate.