Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 15(3): 160-164, Sep. 2017 **Regular paper**

MoTE-ECC Based Encryption on MSP430

Hwajeong Seo[1] and Howon Kim[2*], *Member, KIICE*

[1]Department of IT Convergence Engineering, Hansung University, Seoul 02876, Korea
[2]Department of Computer Engineering, Pusan National University, Busan 46241, Korea

Abstract

Public key cryptography (PKC) is the basic building block for the cryptography applications such as encryption, key distribution, and digital signature scheme. Among many PKC, elliptic curve cryptography (ECC) is the most widely used in IT systems. Recently, very efficient Montgomery-Twisted-Edward (MoTE)-ECC was suggested, which supports low complexity for the finite field arithmetic, group operation, and scalar multiplication. However, we cannot directly adopt the MoTE-ECC to new PKC systems since the cryptography is not fully evaluated in terms of performance on the Internet of Things (IoT) platforms, which only supports very limited computation power, energy, and storage. In this paper, we fully evaluate the MoTE-ECC implementations on the representative IoT devices (16-bit MSP processors). The implementation is highly optimized for the target platform and compared in three different factors (ROM, RAM, and execution time). The work provides good reference results for a gradual transition from legacy ECC to MoTE-ECC on emerging IoT platforms.

Index Terms: Elliptic curve cryptography, Public key cryptography, Software implementation, 16-bit MSP processor

I. INTRODUCTION

Elliptic curve cryptography (ECC) was suggested independently by Neal Koblitz [1] and Victor S. Miller [2] in 1985. The main benefit of ECC is small key size, efficient computation, small storage and low transmission requirements. For this reason, ECC has been used in modern crypto systems from 2004. However, we cannot directly adopt the MoTE-ECC since the cryptography is not fully evaluated in terms of execution timing, code size, and ROM. Moreover, the upcoming computer environments, namely Internet of Things (IoT) which include all kinds of sensors, actuators, meters, consumer electronics, medical monitors, household appliances and vehicles, only support very limited computation powers and storage. For example, a typical wireless sensor node, such as the widely-used TelosB mote, features 16-bit MSP processor clocked at 8 MHz and a few kB RAM. Due to these limitations, implementing public-key algorithms on 16-bit processors poses a big challenge. Therefore, it is necessary to study how well both public key cryptography (PKC) systems are suited for the IoT environments.

This paper continues the line of research on the efficient implementation of the ECC on an IoT devices. The core contributions are several optimizations to reduce the execution time of encryption schemes and fine comparison results on IoT devices. More specifically, our contributions are listed as follows:

- MoTE curve based elliptic curve integrated encryption scheme (ECIES) implementation: We introduce the implementation of encryption scheme over MoTE curve. The key generation is efficiently performed over twisted Edwards curve and encryption/decryption operations are highly optimized with state-of-the-art techniques.

Received 21 April 2017, **Revised** 28 May 2017, **Accepted** 21 July 2017
***Corresponding Author** Howon Kim (E-mail: howonkim@pusan.ac.kr, Tel: +82-51-510-1010)
Department of Computer Engineering, Pusan National University, 2, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan 46241, Korea.

Open Access **https://doi.org/10.6109/jicce.2017.15.3.160** print ISSN: 2234-8255 online ISSN: 2234-8883

- We provide the full performance results, including code size, ROM, and execution timing, of ECC on 16-bit MSP processors.

Based on the above contributions, we present implementtations of ECC based encryption schemes on 16-bit MSP processor. The results are finely evaluated and provide a good reference for cryptography engineers.

The rest of this paper is organized as follows. In Section II, we review the background of ECC based cryptosystem. In Section III, we focus on the optimization techniques for encryption schemes on MSP platforms. In Section IV, we report the implementation results and compare with the state-of-the-art PKC implementations. Finally, we draw our conclusions in Section V.

II. RELATED WORK

A. Elliptic Curve Cryptography

Elliptic curve groups have been widely used to instantiate the discrete logarithm groups in cryptographic schemes. Unfortunately, ECC cryptosystems are easily attacked by a quantum computer. It means we need to replace current crypto infrastructure before quantum computer arrives. However, still majority of systems use ECC and quantum computer will take several years so the optimization of ECC is still valuable.

B. MoTE-ECC

Montgomery [3] introduced in 1997 a special family of elliptic curves with outstanding implementation properties. A Montgomery curve E_M over F_P is defined as

$$E_{M,A,B}: By^2 = x^3 + Ax^2 + x, A, B \in F_P.$$

Montgomery curves allow a special ladder technique, namely Montgomery ladder, to perform a scalar multiplication. Instead of using conventional (x, y) coordinates, scalar multiplication on Montgomery curve can be computed through only the *x* coordinate of the base point. Twisted Edwards curves were introduced by Bernstein et al. [4] in 2008 and are currently considered to be one of the most efficient models for implementing ECC. Let $p>3$ be a prime number. A twisted Edwards curve over F_P can be defined as

$$E_{T,a,d}: ax^2 + y^2 = 1 + dx^2y^2,$$
$$a, d \in F_p, ad(a-d) \neq 0$$

It is known that $E_{T,a,d}$ is birationally equivalent to some Montgomery curve $\frac{E_{M,A,B}}{F_P}: By^2 = x^3 + Ax^2 + x$ where $\mathrm{A} = \frac{2(\mathrm{a+d})}{a-d}, B = 4/(a-d)$. These hybrid class of elliptic curves named MoTE curves with fast and complete group law were introduced in [5]. In this paper, we use efficient MoTE curves for accelerating the ECIES scheme.

III. PROPOSED IMPLEMENTATION

We implement the encryption scheme on the MoTE curve with 255-bit prime field F_P where $p = 2^{255} - 19$ is defined by $E_{M255}: -2556488y^2 = x^3 + 2556486x^2 + x$, which is birationally equivalent to the twisted Edwards curve as $E_{T255}: x^2 + y^2 = 1 + dx^2y^2$, where *d=56774488908756692490607663885244185034990411423271482398238275054452422842304*.

In terms of implementation, finite field operations are the basic building block of MoTE–ECC. Among the finite field operations, modular multiplication and squaring operations require the most expensive computations. The modular multiplication first performs 256×256 -bit integer operations, afterward reduction operation is performed.

In previous works [6, 7], both operations are highly optimized, so we took the part of optimized code from those implementations. They implemented 256-bit multiplication with multiply and accumulate mode in MSP430 hardware multiplier. The accumulation is done automatically by the MSP430 hardware multiplier and holds the lower 16-bit of the intermediate result still in register **RESLO**, the higher 16-bit of the intermediate result in register **RESHI** and the carry flag of the accumulation in register **SUMEXT**. The product-scanning method is the most optimal method with the multiplier. For 256-bit squaring, sliding block doubling technique is used [8]. For memory access optimization, operands are consecutively cached between each column change. In our implementation, we combined the multiplication/squaring operations with modular operations so we avoid several times of memory accesses.

In the MoTE–ECC operations, special 20-bit curve constant (c=639122) multiplication is needed. Unlike ordinary 256×256-bit multiplication, it only performs 20×256-bit multiplication and each word requires 2 16-bit multiplication in MSP430 platforms. This special case can be highly optimized together with modular operations. The constant multiplication outputs 276-bit results. Then the results should be reduced to 256-bit results through reduction operation with 5-bit modulus (p=19). Since the constant and modulus variables are smaller than 20 and 5-bit, we can avoid several carry handing steps.

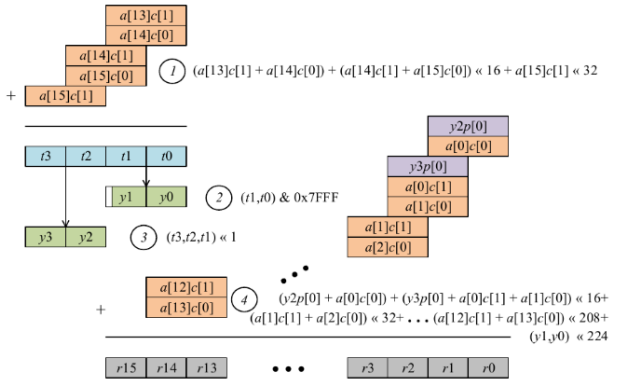


Fig. 1. Optimized 20 × 256-bit constant multiplication. 1: first multiplication; 2: masking; 3: shifting; 4: second multiplication.

More precisely, the modular constant multiplication process consists of four different basic operations, namely, Multiplication → Masking → Shifting → Multiplication, which are given in Fig. 1.

Throughout the paper, we will use the following notations. Let a be one operand with a length of 256-bit that are represented by multiple-word arrays. The operand is written as follows: a = (a[15], ..., a[2], a[1], a[0]), where the word size is 16-bit. The optimized modular constant multiplication can be performed as follows:

1. First multiplication: We first perform the multiplication of (a[13]×c[1]+a[14]×c[0]) + (a[14]×c[1]+a[15]×c[0]) ≪ 16 + (a[15]× c[1]) ≪ 32, and store the result in (t3,t2,t1,t0).
2. Masking: We perform the logical--and operation of (t1,t0) and **0x7FFF**. The results are stored in (y1,y0).
3. Shifting: We left shift (t3,t2,t1) by one bit, and store the higher 32-bit results in (y3,y2).
4. Second multiplication: The fourth step is to multiply the constant and reduce the results (y3,y2).

In the first column, the partial products (y2×p[0]+ a[0]×c[0]) outputs 33-bit results. From second column, the partial products (y3×p[0] + a[0] ×c[1] + a[1] ×c[0]) are apparently less than 23-bit, which does not set overflow register (**SUMEXT**). This nice feature avoids carry handling procedures. In last column, masked results (y1,y0) are added to the intermediate results (r15–r0). The result we get in the step is always below 256-bit.

Modular addition is basic operations for ECC implementation and the operation can be divided into several subroutines. First, the addition of most significant words ((ε, r[15])←a[15] + b[15]) is performed, where r is the sum of operands a and b, ε is the carry bit that may be produced. Then r[15] is masked to get the 15-bit results and the most significant bit of r[15] and ε are outputted to overflow values (*temp*). Finally, the reduced results (*temp* × p) and remaining operands a[0–14] and b[0–14] are added. The modular subtraction follows same procedure with subtraction instructions.

With optimized finite field operations, the group operation are constructed. The point addition and doubling operations of Montgomery curve require (*3M, 2S, 6A*) and (*2M, 1C, 2S, 4A*), where *M*, *C*, *S* and *A* represent modular multiplication, constant multiplication, squaring and addition/subtraction operations.

The point addition and doubling operations of Twisted Edward curve requires (*7M, 6A*) and (*3M, 4S, 6A*), respectively. The random point multiplication is performed over Montgomery curve and Montgomery ladder algorithm is selected. The fixed point multiplication is performed over Twisted Edward curve and COMB algorithm is used.

Table 1. Performance comparison of software implementation of ECC-based cryptosystems on different processors in execution timing (clock cycles)

Implementation	Curve	Fixed (cycles)	Random (cycles)
8-bit AVR processors, e.g., ATxmega64, ATxmega128			
Liu et al. [5]	MoTE255	9,433,600	21,078,200
Liu et al. [7]	MoTE255	8,591,000	18,270,000
Wenger et al. [9]	NIST P256	N/A	34,930,000
Hutter and Schwabe [10]	Curve25519	N/A	22,791,579
Dull et al. [11]	Curve25519	N/A	13,900,397
16-bit MSP processors, e.g., MSP430F1611			
Wenger and Werner [12]	NIST P256	N/A	23,937,000
Wenger et al. [9]	NIST P256	N/A	22,170,000
Liu et al. [7]	MoTE255	4,798,000	10,952,000
This work	MoTE255	4,745,502	10,712,766
32-bit ARM processors, e.g., Cortex-M0			
Wenger et al. [9]	NIST P256	N/A	10,730,000
Dull et al. [11]	Curve25519	N/A	3,589,850

Table 2. Execution timing (clock cycles) and code size (bytes) of ECC-based encryption on MSP430 processors

Gen (cycles)	Enc (cycles)	Dec (cycles)	ROM (bytes)	RAM (bytes)
5.0 M	15.8 M	11.0 M	21 K	1.4 K

In order to perform encryption, we choose the ECIES whose security is based on the Diffie–Hellman problem. The scheme requires block cipher and hash function for key derivation and HMAC functions. We used the library offered by Texas Instruments to support both functions.

IV. EVALUATION

In Table 1, performance comparison of ECC implementations are given. The implementations of ECC are widely studied in three different embedded processors including 8-bit AVR, 16-bit MSP, and 32-bit ARM. The highest performance is observed in ARM processor, since the processor supports long word size and strong instruction sets. Particularly, in MSP430 processor, Liu et al. [5] suggested efficient implementation over MoTE-255 curve. In this paper we employed major operations from their implementations and further improve the constant multiplication part and modular operations. Finally, we improved the performance by 1.1% and 2.2% for fixed point and random point computations, respectively.

In Table 2, the performance of ECC based encryption on MSP430 processors is described. The result shows that the encryption and decryption take 0.988 and 0.688 seconds under 16 MHz operation frequency, respectively. This proves that the practical PKC based encryption is available in 16-bit MSP processors.

V. CONCLUSION

This paper presented several optimizations for efficiently implementing ECC based encryption schemes on 16-bit MSP platform. In particular, we introduce the implementation of encryption scheme (ECIES) over MoTE curve. The key generation is efficiently performed over Twisted Edward curve and encryption/decryption operations are highly optimized with state-of-art techniques. Finally we provide the practical results of ECC based encryption on 16-bit MSP processors.

ACKNOWLEDGMENTS

This research was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2012-0-00265, Development of high performance IoT device and Open Platform with Intelligent Software). This research of Hwajeong Seo was financially supported by Hansung University.

REFERENCES

[1] N. I. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, 1987.

[2] V. S. Miller, “Use of elliptic curves in cryptography,” in *Conference on the Theory and Application of Cryptographic Techniques*. Heidelberg: Springer, pp. 417-426, 1985.

[3] P. L. Montgomery, “Speeding the Pollard and elliptic curve methods of factorization,” *Mathematics of Computation*, vol. 48, no. 177, pp. 243-264, 1987.

[4] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, “Twisted Edwards curves,” in *Progress in Cryptology AFRICACRYPT 2008*. Heidelberg: Springer, pp. 389-405, 2008.

[5] Z. Liu, E. Wenger, and J. Groschadl, “MoTE-ECC: Energy-scalable elliptic curve cryptography for wireless sensor networks,” in *ACNS 2014: Applied Cryptography and Network Security*. Cham: Springer International Publishing, pp. 361-379, 2014.

[6] Z. Liu, X. Huang, Z. Hu, M. K. Khan, H. Seo, and L. Zhou, “On emerging family of elliptic curves to secure internet of things: ECC comes of age,” *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 3, pp. 237-248, 2017.

[7] Z. Liu, H. Seo, Z. Hu, X. Hunag, and J. Groschadl, “Efficient implementation of ECDH key exchange for MSP430-based wireless sensor networks,” in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, Singapore, pp. 145-153, 2015.

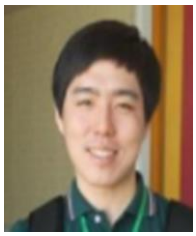
[8] H. Seo, Z. Liu, J. Choi, and H. Kim, “Multi-precision squaring for public-key cryptography on embedded microprocessors,” in *Progress in Cryptology–INDOCRYPT 2013*. Heidelberg: Springer, pp. 227-243, 2013.

[9] E. Wenger, T. Unterluggauer, and M. Werner, “8/16/32 shades of elliptic curve cryptography on embedded processors,” in *Progress in Cryptology–INDOCRYPT 2013*. Heidelberg: Springer, pp. 244-261, 2013.

[10] M. Hutter and P. Schwabe, “NaCl on 8-bit AVR microcontrollers,” in *Progress in Cryptologygy–AFRICACRYPT 2013*. Heidelberg: Springer, pp. 156-172, 2013.

[11] M. Dull, B. Haase, G. Hinterwalder, M. Hutter, C. Paar, A. H. Sanchez, and P. Schwabe, “High-speed curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers,” *Designs, Codes and Cryptography*, vol. 77, no. 2-3, pp. 493-514, 2015.

[12] E. Wenger and M. Werner, “Evaluating 16-bit processors for elliptic curve cryptography,” in *CARDIS 2011: Smart Card Research and Advanced Applications*. Heidelberg: Springer, pp. 166-181, 2011.



Hwajeong Seo
received the B.S.E.E. degree in 2010, and the M.S. degree in 2012 and the Ph.D. degree in 2016 in Pusan National University. He is currently an assistant professor in Hansung University.

Howon Kim
received the B.S.E.E. degree from Kyungpook National University, Daegu, Korea, in 1993, and the M.S. and Ph.D. degrees in electronic and electrical engineering from the Pohang University of Science and Technology, Pohang, Korea, in 1995 and 1999, respectively. From 2003 to 2004, he studied with the COSY Group, Ruhr University Bochum, Germany. He was a Senior Member of the Technical Staff with the Electronics and Telecommunications Research Institute, Daejeon, Korea. He is currently an Associate Professor in Pusan National University.