ECC Implementation on Wireless Sensor Nodes

Mohamed Said Albahri

***Abstract*—This paper is concerned with the implementation and performance evaluation of Elliptic Curve Cryptography in constrained devices such as wireless sensor nodes. Experimental evaluation for Elliptic Curve Digital Signature (ECDSA) on an 8-bit (Arduino mega2560) and a 32-bit (Arduino Due) processor using the Relic-toolkit has been carried out and comparative implementation results are given. It is shown that by adopting appropriate optimizations an ECDSA can be achieved in 83ms.**

***Index Terms*— WSN, ECC, Software Implementation, Relic toolkits.**

I. INTRODUCTION

The recent and expected proliferation of wireless sensor networks (WSN) with all its economical and societal benefits across a range of applications spanning healthcare, home, environment and defense will face serious limitations if security concerns are not addressed. Cryptography plays a very important role in achieving security.

Elliptic Curve Cryptography (ECC) is increasingly becoming the first choice for public key cryptography implementation as it requires much shorter key sizes compared to the RSA for the same level of security.

The implementation of ECC on sensor node platforms remains a challenge due to the resources limitation in these nodes.. Therefore, optimal low resource ECC implementations are required with optimization techniques to speed up the ECC operations and to reduce the memory usage without prohibitive complexity.

The Relic- toolkit developed by [1] is an attractive platform for providing security in WSN. It has many features compared to the other ECC open sources libraries and [2]–[4] and sup-ports many modern cryptography functions and protocols such as (ECDSA, ECDH,RSA and ECMQV).

The contribution of this work is to present the design, implementation and practical evaluation of ECC for con-strained environments by deploying an efficient cryptography library (the relic-toolkit) in platforms representative of wireless sensor node platforms. Experimental analysis and evaluation for Elliptic Curve Digital Signature (ECDSA) on both an 8-bit and a 32-bit platform (Arduino mega2560 and Arduino Due) has been carried out and comparative implementation results are given. To our knowledge no such analysis and results have been reported to date.

The implementation results obtained, show that ECDSA key generation on Arduino Due can be achieved in (90ms) com-pared to (263ms) on the Arduino Mega for m=163. Further-more, implementation optimization (such as multi-precision GF (2m) arithmetic) configurations are shown to enhance the performance of the ECDSA on the Arduino Due to (83 ms). These results will act as a useful benchmark and guidance in selection of the optimization techniques provided by the tool for a given WSN application.

This paper is organized as follows: Section II provides ECC background. The third section provides the design principle. The optimizations provided by the relic-toolkit are presented in section IV. The implementation work is described in section V. In section VI we present the results analysis. Finally, we conclude this paper in section VII.

II. ECC BACKGROUND

In 1985 both Neal Koblitz and Victor S.Miler proposed independently elliptic Curve Cryptography. Elliptic Curve Cryptography is based on Elliptic Curve theories. Currently, ECC is considered to be one of the main players for implementing security in different applications. Basically, ECC has better features and future for cryptography as it has the capability to provide many cryptography schemes, such as key Management, Digital Signature and Verification. Beside these services and its powerful security, ECC has more powerful computation with shorter key length sizes compared to the other public key cryptography solutions such as RSA and Diffie-Hellman. ECC could be defined over prime fields and binary fields . However, for a purpose of this work we consider Elliptic Curve over binary fields. The equation below represents the elliptic curve over binary fields [5]:

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

where $b \neq 0$ and the value of x; y; a and b are polynomials representing n bit words.

Finding points on the curve could be achieved by using generator for polynomials and irreducible polynomial. The rules for points addition in GF (2^m) is different from GF (P) Therefore, if P = (x1, y1) and Q = (x2, y2) and $Q \neq P$, then can be found as shown below:

$$= (y2 + y1)/(x2 + x1)$$

$$x3 = \lambda^2 + \lambda + x1 + x2 + a$$

$$y3 = \lambda (x1 + x3) + x3 + y1 \quad (2)$$

and if Q = P then R = P + P or R = 2P can be found as below:

$$\lambda = x1 + y1/x1$$

$$x3 = \lambda^2 + \lambda + a$$

$$y3 = x2 + (\lambda + 1)x3 \quad (3)$$

Revised Version Manuscript Received on August 01, 2015.
Mohamed Said Albahri, German University of Technology, Oman.

In the other hand the point doubling 2P can be found as below :

Let P 1 = (x1, y1) E(GF (2^m)) where P1 ≠ -P and 2P = (x3, y3) then,

$$x3 = x_1^2 + \frac{b}{x_1^2} \;\&\; y3 = x_1^2 + \lambda x3 + x3 \qquad (4)$$

where $\left[\frac{y2+y1}{x2+x1}\right]$ P1 ≠ P2& $\lambda = x1 + \frac{y1}{x1} P1 = P2$

Elliptic Curve Digital Signature (ECDSA) is used for digital signature purposes and it consists of three main procedures which are key pair generation, signature generation and sig-nature verification. The Elliptic Curve Diffie Hellman protocol is used for exchanging the keys between two parties over an insecure channel. The purpose for having the ECC schemes is to provide high level of security with smaller key sizes. Therefore, it is important for both parties involved in the communication to have per-defined and agreed domain parameters for each scheme. The detailed specification can be found in [6].

III. DESIGN PRINCIPLES

The primary objective for the relic-toolkit is to construct efficient and configurable cryptographic software capable to implement a certain level of security and algorithms. Therefore, we achieved these objectives through different design principles that we considered during the implementation stages.

Security: The relic-toolkit is designed to provide cryptography protocols such as (RSA, ECDH, ECDSA, ECSS and ECMQV). In addition to that relic-toolkits support the implementation of ECC over prime field and ECC over bi-nary field. This includes different Elliptic Curve parameters recommended by the Standard for Efficient Cryptography Group (SECG) such as Secp160k1, Secp160r1 and Secp160r2 detailed by [7].

Configurability: The Configurability principle is a achieved by allowing the user to select the desired components for the targeted platform during the library building procedures. Furthermore, the required performance can be achieved by combining and selecting different type of mathematical optimization provided by the tool.

Portability: The Relic-toolkit can be used with different type of Wireless Sensor platform such as ARM, AVR and MSP. Additionally, the library could be built in different type of operating system such as windows (using MingW), Ubuntu and Mac OS. In this work we consider importing and testing the relic library in Arduino mega260 (AVR- 8-bit processor) and Arduino Due(ARM-cortex-32 bit processor).

Efficiency: To better achieve the desired efficiencies from the tool we decided to implement the ECC over binary fields based on the potential result reported by [end to end security]. We also used an assembly version (shown as K163-asm) file in order to achieve better performance as recommended by [8].

Functionality: This principle is insured through the practical implementation for different public key cryptography schemes provided by the relic-toolkit such as ECDH and ECDSA.

IV. OPTIMIZATION

In this section we aim to provide relevant optimization techniques accomplished with the optimization algorithms available in the tool. The detail provided with this regard is limited to the optimization techniques used in this paper.

A. Optimization for Multiple Precision Arithmetic

The multiple precision is required for big number arithmetic. It is highly efficient for public key cryptography implementations in resolving memory limitations as well as overcoming overflow issues. The contribution of multiple precision on solving such problems is through in-creasing the integer representation while using single precision data type [9]. The implementation procedure consists of three different phases. These phases are temperance initialization, column calculation and carries propagation phase. Further to optimistic results reported by [10] the author of [11] shows better performance compared to the school book multiplication method. However, the relic-toolkits allow the user to select from different type of multiple precision arithmetic algorithms beside the comba algorithm such as School-Book multiplication, Karatsuba multiplication and others.

Montgomery-comb Modular Reduction Algorithm:

A modular reduction is a process of finding the reminder of dividing two products:

$a \equiv b(modc)$ where b is restricted with range $0 \le b < c^2$

Modular reduction is important in ECC public key cryptography computations.

The implementation of the Montgomery modular reduction algorithm involves fewer single multi precision multiplications in comparison with Barrett Modular reduction which requires two modified multipliers. Previous software implementation of the Montgomery algorithm reported slower speed. This challenge has been tackled and resolved by the researchers through combing the Montgomery modular reduction and comba algorithms. The combination methodology could be achieved by allowing the comba algorithm to act as multiplier.

Comba Squaring Algorithm:

Multiple Precision Squaring is a process of multiplying two equal multiplicands and influences overall implementation performance.. The software implementation for squaring could be performed using multiplication algorithms or using specialized squaring methods. Using specialized squaring helps to reduce the load operand approximately by half over using multiplication algorithms. Additionally it helps to enhance the computation performance for the duplicated partial products. Moreover, the specialized squaring algorithms contribute to overcome the limitation of base line multiplication algorithms. These limitations could be summarized into two main points. Firstly, the needs for processing single precision shift inside the nested loop. Secondly, the challenges of performing the products doubling process inside the inner loop.

The comba squaring algorithm could be used to solve these drawbacks. The concept of comba squaring is to some extent similar to the comba multiplication algorithm with some differences that help to accommodate the single precision shifting and doubling processes. The relic tool-kit supports three different multiple precision square algorithms besides the comba squaring which are the Karatsuba Squaring, the recursive karatsuba and the School book method [9]. In this work we configure the Relic library with Comba squaring to gain better performance.

B. Optimization for Elliptic curve Arithmetic

Point Representations: There are different coordinate systems that can be used to represent the elliptic curve. The affine coordinates and projective coordinates are the most well known. The projective coordinates can be considered as an option that can help avoid the costly and expensive multiplication and inversion operations. The results reported by [12] show better performance achievement compared to the affine. The relic-toolkit has been designed to support both and we selected the projective coordinate to achieve better performance.

Point Multiplications: Point Multiplication or scalar multiplication is implemented through a series of point addition and point doubling operations. The key is to be obtained after conducting full cycle of addition and doubling operations. The point multiplication over the binary elliptic curve can be implemented with different algorithms such as Left-to-right binary algorithm, Halving. Right-to -left width-w and others. The relic library consists of six different algorithms such as the Basic binary point multiplication algorithms, Lopez-Dahab point multiplication and Right-to-left width-w (T)NAF. Since the sliding windows method is more helpful on speeding up the scalar multiplication we selected the right to left width
(T) NAF algorithm for performing the point multiplication. The concept of sliding windows is based on scanning bit at a time and perform the point doubling for the at the same time [3].

Simultaneous Point Multiplications: Enhancing the efficiencies and speeding up computation of point multiplication has been intensively considered by many researchers due to its importance in some ECC schemes. For example the implementation of ECDSA required two types of point multiplication. The first one is needed for signature generation where is fixed in . Whereas, the second type is to be used in signature verification process where is also fixed but is unknown. However, speeding up the signature verification processes can be achieved using simultaneous multiple point multiplication. Different methods have been proposed for simultaneous point multiplications such as Shamir's trick, Joint sparse form and Interleaving. With this aspect the relic-toolkits support all of these methods plus the basic simultaneous point multiplication methods that can be selected during the relic building process.

V. IMPLEMENTATION

Point Representations: We imported the Relic-toolkit [1] into the arduino mega 2560 (8-bit AVR processor) [13] and arduino Due (32-bit ARM processor). Our selection for these platforms is based on fact that we targeted to implement the ECC schemes on a processor that does not require an operating system support. Furthermore, the 8-bit to 32-bit processor range is a representative range for constrained applications. We imported relic-0.3.1 onto the two platform boards and we experimented with the performance of ECDSA and ECDH over binary fields using different NIST curves standard (NIST-K163,NIST-B163). For obtaining better performance we exam-ined the presets provided by [2]. The execution timings of the codes were measured using inbuilt millis() function provided by Arduino.h library. Furthermore, we measured the amount of RAM using "MemoryFree.h" library beside the avr-size and arm-none-eabi-size tools.

Experiment Setup: In order to build the library we installed the avr-gcc version 4.5.3 compiler and cmake cross-platform version 2.8.7. The recommended presets by [1] shown in Figures 4 and 5 in the Appendix were used for building the library with low memory optimization algorithms and faster time execution respectively. For importing the relic-toolkit in arduino Due we installed arduino extension plug-in (embedxcode) in Xcode IDE MAC OS X version 10.7.3 then we imported the relic-toolkits into the XCode IDE.

VI. RESULTS ANALYSIS

Due to the importance of time and memory usages we considered evaluating our ECC implementation based on these two factors. The arduino mega2560 is an 8 bit micro-controller but it has the capability to manipulate 16X16 bit operations by using two separate registers. From the other perspective, the arduino Due is a 32bit micro-controller and can easily handle 8 and 16 bit operations. We measured the execution time using the inbuilt function millis() provided by the ardunio library. This function returns the timing result in milliseconds using the arduino internal timer #0 or TCNT0. However, the timer runs at 16 MHz in arduino mega2560 and at 84 MHz in arduino DUE. On the other hand, we measured the amount of RAM using arm-none-eabi-size tool for arduino DUE and we used the avr-size tool for measuring the RAM utilization in arduino mega2560.

ECDSA: In this part we demonstrate the main obtained results with regards the ECDSA performance. Figure -1 below shows the time execution for ECDSA key generation on both platforms. As expected the arduino mega2560 takes more time to generate the ECDSA keys as it runs at much lower clock than the arduino Due. In addition, the binary field arithmetic with BASIC algorithm configuration resulted an improved performance on the DUE as shown in figure-2. The performance on the mega2560 was improved using the assembly code provided in the library. This enhancement is represented by the figures below which include the time execution improvement and memory usages respectively. These results show even better performance compared to the results reported by [3] and [4]

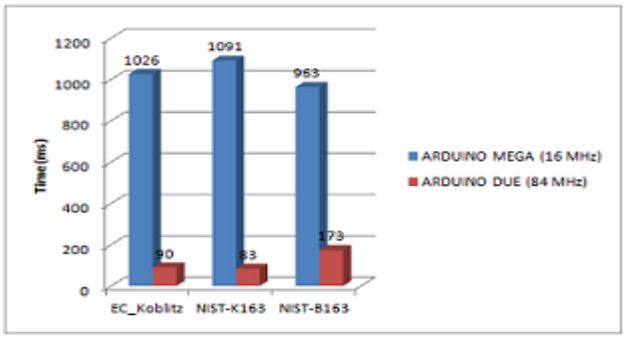

Figure 1. Execution time key generation for ECDSA

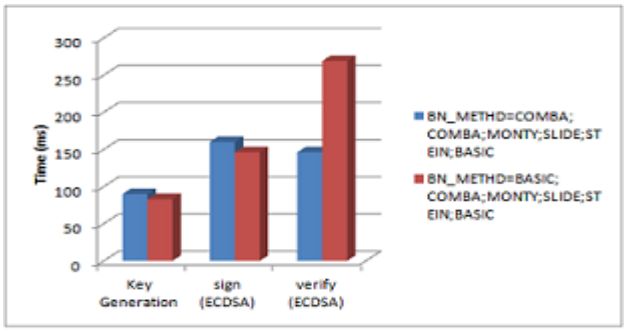

Figure 2. Execution time of ECDSA

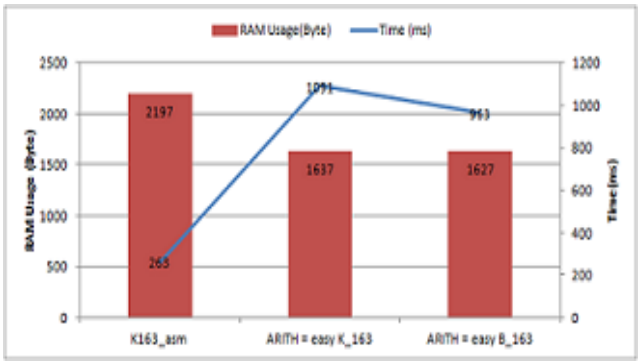

Figure 3. ECDSA Arduino mega2560 time improvement

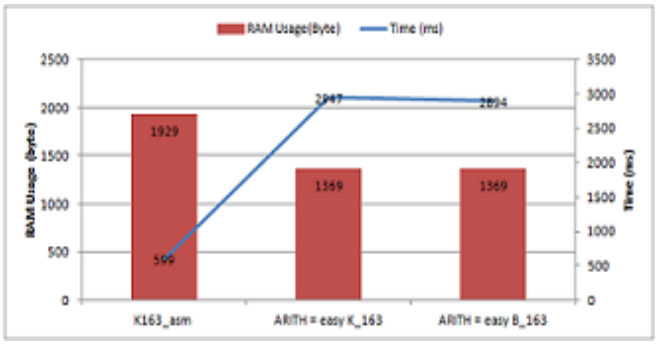

Figure 4. ECDSA Arduino mega2560 memory improvement

VII. CONCLUSION

In this work, we illustrated the potential of implementing Relic-tookits on sensor node platforms. We also evaluated some of the optimization methods and their effectiveness in the ECDSA implementation Performance. The configuration features provided by the relic-toolkit can help enhance the ECC performance which could be considered as a benchmark and guidance for the developer planning to use the relic in resource constrained processor platforms such as the ones presented in this paper.

APPENDIX

```
CC=avr-gcc CXX=c++ LINK="-mmcu=atmega2560 -Wl,-gc-
sections" COMP="-O2 -ggdb -Wa,-mmcu=atmega2560 -
mmcu=atmega2560 -ffunction-sections -fdata-sections"

cmake -DARCH=AVR -DWORD=8 -DOPSYS=NONE -DSEED=LIBC -
DSHLIB=OFF -DSTBIN=ON -DTIMER=NONE -
DWITH="DV;BN;FB;EB;EC;CP;MD" -DBENCH=20 -DTESTS=20 -
DCHECK=off -DVERBS=off -DSTRIP=on -DQUIET=on -
DARITH=avr-asm-163 -DFB_POLYN=163 -
DBN_METHD="COMBA;COMBA;MONTY;SLIDE;STEIN;BASIC" -
DFB_METHD="INTBB;INTBB;QUICK;BASIC;BASIC;BASIC;EXGCD;BAS
IC;BASIC" -DFB_PRECO=off -DFB_TRINO=off -DBN_PRECI=160 -
DBN_MAGNI=DOUBLE -DEB_PRECO=off -
DEB_METHD="PROJC;BASIC;LWNAF;BASIC" -DEB_MIXED=on -
```

Relic low memory preset

```
CC=avr-gcc CXX=c++ LINK="-mmcu=atmega2560 -Wl,-gc-
sections" COMP="-O2 -ggdb -Wa,-mmcu=atmega2560 -
mmcu=atmega2560 -ffunction-sections -fdata-sections"
cmake -DARCH=AVR -DWORD=8 -DOPSYS=NONE -DSEED=LIBC -
DSHLIB=OFF -DSTBIN=ON -DTIMER=NONE -
DWITH="DV;BN;FB;EB;EC;CP;MD" -DBENCH=20 -DTESTS=20 -
DCHECK=off -DVERBS=off -DSTRIP=on -DQUIET=on -
DARITH=avr-asm-163 -DFB_POLYN=163 -
DBN_METHD="COMBA;COMBA;MONTY;SLIDE;STEIN;BASIC" -
DFB_METHD="INTBB;INTBB;QUICK;BASIC;BASIC;BASIC;EXGCD;B
ASIC;BASIC" -DFB_PRECO=off -DFB_TRINO=off -
DBN_PRECI=160 -DBN_MAGNI=DOUBLE -DEB_PRECO=on -
DEB_METHD="PROJC;LWNAF;LWNAF;INTER" -DEB_MIXED=on -
DEB_KBLTZ=on -DEB_ORDIN=off -DEB_SUPER=off -
```

Relic Faster time execution

REFERENCES

1. D. F. A. Gouv and C. P. L., "Relic is an efficient library for cryptography." [Online]. Available: ttp://code.google.com/p/relic-toolkit/
2. "Avrcryptolib," 2014. [Online]. Available: http://www.emsign.nl/
3. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on. IEEE, Conference Proceedings, pp. 245–256.
4. S. C. Seo, H. Dong-Guk, H. C. Kim, and H. Seokhie, "Tinyecck: Ef-ficient elliptic curve cryptography implementation over< i> gf</i>(< i> 2</i>< i>< sup> m</sup></i>) on 8-bit micaz mote," IEICE transactions on information and systems, vol. 91, no. 5, pp. 1338–1347, 2008.
5. N. Koblitz, "Elliptic curve cryptosystems," Mathematics of computation, vol. 48, no. 177, pp. 203–209, 1987.
6. "Sec1 final," 2014. [Online]. Available: http://www.secg.org/collateral/ sec1_final.pdf
7. "sec2 final," 2014. [Online]. Available: http://www.secg.org/collateral/ sec2_final.pdf
8. M. Sethi, J. Arkko, and A. Keranen, "End-to-end security for sleepy smart object networks," in Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on, Conference Proceedings, pp. 964–972.
9. T. S. Denis, BigNum Math: Implementing Cryptographic Multiple Pre-cision Arithmetic. Syngress Publishing, 2006.
10. P. G. Comba, "Exponentiation cryptosystems on the ibm pc," IBM systems journal, vol. 29, no. 4, pp. 526–538, 1990.
11. J. GroÃ§schÃd'dl, R. M. Avanzi, E. SavaÅ§, and S. Tillich, Energy-efficient software implementation of long integer modular arithmetic. Springer, 2005, pp. 75–90.
12. D. Hankerson, J. L. Hernandez, and A. Menezes, "Software implementa-tion of elliptic curve cryptography over binary fields," in Cryptographic Hardware and Embedded Systemsâ ATCHES 2000. Springer, Confer-ence Proceedings, pp. 1–24.
13. "Arduino - homepage," 2014. [Online]. Available: http://www.arduino. cc/