An Implementation of Digital Signature and Key Agreement on IEEE802.15.4 WSN Embedded Device

Amang Sudarsono and Mike Yuliana
Division of Telecommunication Engineering, Dept. of Electrical Engineering,
Electronic Engineering Polytechnic Institute of Surabaya (EEPIS), Surabaya, Indonesia.
EEPIS Campus, Jalan Raya ITS Sukolilo, Surabaya 60111
Tel: +62(31) 594 7280; Fax: +62(31) 594 6114
Email : amang@eepis-its.edu, mieke@eepis-its.edu

Abstract

A wireless sensor network (WSN) now becomes popular in context awareness development to distribute critical information and provide knowledge services to everyone at anytime and anywhere. However, the data transfer in a WSN potentially encounters many threats and attacks. Hence, particular security schemes are required to prevent them. A WSN usually uses low power, low performance, and limited resources devices. One of the most promising alternatives to public key cryptosystems is Elliptic Curve Cryptography (ECC), due to it pledges smaller keys size. This implies the low cost consumption to calculate arithmetic operations in cryptographic schemes and protocols. Therefore, ECC would be strongly required to be implemented in WSN embedded devices with limited resources (i.e., processor speed, memory, and storage). In this paper, we present an implementation of security system on IEEE802.15.4 WSN device with the employment of Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol. Our experimental results on Intel Mote2 showed that the total time for signature generation is 110 ms, signature verification is 134 ms, and ECDH shared key generation is 69 ms on the setting of 160-bit security level.

Keywords: WSN, ECC, ECDSA, ECDH key exchange, embedded device.

1. Introduction

A sensor in a WSN is simply refer to a device that raises an electrical signal that contains a valuable information and usually has a property such as low cost, low performance, and limited resources (i.e., processor speed, memory, and storage) to be used for developing context awareness. The main goal of WSN is to distribute and deliver critical information and knowledge services to everyone at anytime and anywhere. A WSN may encompass a wide range of area and support a variety of applications (e.g., environment monitoring, disaster/crisis management, home utilization control, telematics, mobile RFID, etc). Besides, sensor networks are essentially connected to user networks through common use networks, such as Internet or other public network connections. Thus, there should be many potential threats and attacks in transferring information via a WSN. To overcome this situation, an effective security system is required to address those problems.

A wireless sensor network (WSN) now becomes popular, hence many researchers have taken in account this area of interest, such as in [2],[4],[5],[6]. They have considered in the research both in the system design and the implementation.

However, the data transfer in a WSN potentially encounters many threats and attacks. Hence, particular security schemes are required to prevent them. A WSN usually uses low power, low performance, low processor speed, limited memory and storage size, such as embedded devices [2],[4],[5]. One of the most promising alternatives to public key cryptography (e.g., RSA and ElGamal) is Elliptic Curve Cryptography (ECC)[1],[7], due to it offers smaller keys size rather than RSA or ElGamal for the same level of cryptographic strength. For example, the setting of 160-bit in ECC provides the same security level with 1024-bit length of RSA or ElGamal. This implies that ECC promises the requirements of small space and memory in the implementation on embedded devices [2],[4]. In addition, ECC consumes a low cost calculation of arithmetic operations in cryptographic schemes and protocols [5]. Therefore, ECC is the best candidate and suitable algorithm to be implemented on the WSN embedded devices.

One of small operating systems implemented for constructing a secure WSN based on the use of ECC is TinyOS [2],[4]. There also has been introduced a low cost ECC on the implementation of WSN [5], and the construction of secure ECC on WSN [6]. In this paper, we present an

implementation of security scheme on IEEE802.15.4 WSN device, namely Intel Mote2 platform [10] with the employment of Elliptic Curve Digital Signature Algorithm (ECDSA) and Diffie-Hellman (DH) key exchange protocol to realize our security system in WSN.

2. Security Requirements

In modern wireless communication systems including WSN, the security and efficiency features have to be considered. In addition to the common networking threats and attacks, especially in WSNs, these kind of threats and attacks include: (1) sensor node compromise that is caused when sensors in WSN are being attacked or compromised or may be an attacker inserting illegal sensors to an existing system; (2) eavesdropping which is performed by monitoring transmissions between nodes; (3) compromise or exploration of sensed data; (4) denial of service (DoS) attacks which try to block sensors and communications; and (5) malicious use or misuse in WSN for illegal purposes. To encounter these problems, in this section, we briefly describe the security requirements needed by WSN.

2.1 Mutual Authentication

The situation of illegally nodes to involve in a legitimate WSN could be happened in early sensor networks [3]. The main goal is to be granted for accessing the important sensed data from legal nodes or gateways. Hence, it is very important for all nodes to authenticate each other or between nodes and gateways before sensed data or other important information are being exchanged.

2.2 Nonrepudiation of Service

To satisfy a good WSN design and its implementations, it also has to be considered the possibility of a node to disclaim the charges of sending or receiving sensed data or services. Digital signature is one of solutions to meet this security requirement and this feature can easily be applied in security protocols.

2.3 Confidentiality

There must be many purposes on WSN including business or commercial tendencies. Thus, by today's technology, illegal nodes can easily intercept radio signals propagating over the air in WSN. As the results, there exists disposing and eavesdropping activities. To address this problem, all nodes in WSN and gateways must deal with on the use of keys to encrypt messages on every communication session. Key exchange, key agreement, and session keys are an important stage of the authentication mechanism between nodes and gateways.

2.4 Security Algorithm Selection

In the introduction section has already been mentioned that WSN comprises many sensors with usually adhere limited resources. For cryptosystem protocols, not only security is the most important concern, but also practical implementation should also be considered. Hence, a design and selection security algorithm to produce a very efficient cryptosystem protocol can be implemented in WSN. In addition, there are many exchanged messages involving in this protocol and it should consume amount memory, thus it is a challenging to design an efficient protocol.

3. Elliptic Curve over F_p and Complex Multiplication Method

In this section, we briefly describe the fundamental of Elliptic Curves over F_p and the method of Complex Multiplication (CM). The detail explanation about elliptic curves and CM method can be referred in [1] and [7].

Let denote $p > 3$ be an odd prime number, a prime field F_p which consists of a set of integer numbers $\{0, 1, \ldots, p\text{-}1\}$ and also the operations of arithmetic, such as addition and multiplication with modulo by p. Let also define an elliptic curve $E(F_p)$ over F_p is a set of point P which has coefficients (x, y) where $x, y \in F_p$ and must fulfill the following equation:

$$y^2 = x^3 + ax + b \pmod{p} \quad (1)$$

where $a, b \in F_p$ satisfies $4a^3 + 27b^2 \neq 0$. This set of point along with the *point at infinity*, O. The point O plays a role of identity component to perform special arithmetic operation and defines an Abelian Group called *Elliptic Curve Group*. The multiplication operation over $E(F_p)$ is done based on addition operation as follows:

$$Q = k \times P = P + \underbrace{P + \ldots + P}_{k} \quad (2)$$

where $k \in Z$, and Z is integer numbers. In addition, let denote Q and P are points on $E(F_p)$. The point $P = (x, y)$ has an order m, which is the smallest integer k that satisfies: $k \times P = O$. The order m of an elliptic curve $E(F_p)$ is the number of points on $E(F_p)$, where $k \leq m$. The discriminant Δ of $E(F_p)$ and j-invariant are defined by:

$$\Delta = -16(4a^3 + 27b^2), \text{ where } j = -1728(4a)^2/\Delta \quad (3)$$

On the given a j-invariant $j_0 \in F_p$ and $j_0 \neq 0.1728$. Then, an $E(F_p)$ can be easily generated by set of $a = 3k \bmod p$ and $b = 2k \bmod p$, where $k = j_0/(1728\text{-}j_0)$. The second elliptic curve called twist from the previous constructed elliptic curve can be defined as follows:

$$y^2 = x^3 + ac^2x + bc^3 \quad (4)$$

where c is any quadratic non-residue in F_p. After completing the construction of two elliptic curves

(equations (1) and (4)), the $E(F_p)$ with an order m ensures the intractability of solving the problem of Discrete Logarithm Problem (DLP) on the elliptic curve group when constructing an elliptic curve based cryptosystem.

To construct elliptic curve by using CM method, firstly we need to select a suitable order m which can be defined by calculating j-invariant. Then, the CM method is started from the selecting of a prime p and finding the smallest D which is a discriminant value of CM method on p. Let denote two integer values, u and v that satisfies the following equations:

$4p = u^2 + Dv^2$ and $m = p + 1 \pm u$. (5)

Then, it must be checked whether $p + 1 - u$ or $p + 1 + u$ are suitable orders for constructing elliptic curve. Otherwise, such procedure must be repeated, if not the next step is based on the value of D to create Hilbert polynomials [1],[3] and find their roots. The root is created from j-invariant when generating elliptic curve and its twist or the second curve. Then Lagrange method is used to select a suitable one of the two elliptic curves by picking randomly point P from each elliptic curve randomly until the point fulfills equation: $m \times P \neq O$ is found. Then, another curve can be judged as the right one.

Computationally, CM method is costly when calculating Hilbert polynomials, due to a lot of coefficients, but it produces a high precision floating point and complex arithmetic operations. To address this problem, but still uses the advantage property of Hilbert polynomials, CM method adopts Weber polynomials in addition of Hilbert polynomials. This phenomenon is described detail in [1].

3.1 Elliptic Curve Digital Signature Algorithm (ECDSA)

An elliptic curve $E(F_p)$ over Galois Field $GF(p)$ with large of order p and a point P of large order are selected and made them public to all nodes. Then, a pair of public key and private key are generated. Furthermore, for each transaction, the signature generation and verification are implemented. We briefly outline the ECDSA as follows:

ECDSA *key generator*: the Node A performs the following steps:

I. Select a random $d \in [2, p-2]$.
II. Compute $Q = d \times P$
III. *Public key* (*pk*) and *secret key* (*sk*) for Node A is a couple of $(E(F_p), P, n, Q)$ and d, respectively.

ECDSA *signature generation*: Node A signs a message M by considering the following steps:

I. Select a random integer $k \in [2, p-2]$.
II. Compute $k \times P = (x_1, y_1)$ and $r = x_1 \bmod p$.
    A) If only if $x_1, y_1 \in F_p$.
    B) If $r = 0$, then return to the step I.
III. Compute $k^{-1} \bmod p$.
IV. Compute $s = k^{-1}(H(M) + d \times r) \bmod p$.
    A) H is a *hash* algorithm
    B) If $s = 0$, then return to the step I.
V. *Signature of message M* is a pair of integer (r, s).

ECDSA *signature verification*: Node B verifies Node A's *signature* (r, s) for a message M by applying the following steps:

I. Compute $c = s^{-1} \bmod p$ and $H(M)$.
II. Compute $u_1 = H(M) \times c \bmod p$ and $u_2 = r \times c \bmod p$.
III. Compute $u_1 \times P + u_2 \times Q = (x_0, y_0)$ and $v = x_0 \bmod p$.
IV. *Signature* can be said valid if only if $v = r$, otherwise it is invalid.

3.2 Elliptic Curve Diffie-Hellman (ECDH) Key Agreement

Let assume that Node A wants to establish a shared key with Node B. At first, two parties initiate and deal with the global parameter which is a couple of (p, a, b, G) in the prime number F_p. Also, each node must have a key pair suitable for elliptic curve $E(F_p)$ which consists of a private key d, namely a selected random number in the interval $[1, p-1]$ and a public key Q (where $Q = d \times G$ and G is a base point of elliptic curve). Let Node A's key pair be (d_A, Q_A) and node B's key pair be (d_B, Q_B). Each node must have the other node's public key, thus there should be a key exchange mechanism between them. Upon receiving Node B's public key, Node A computes a shared key $k = d_A \times Q_B$. Similarly, Node B computes $k = d_B \times Q_A$. The calculated shared key k by both nodes is equal, because $d_A \times Q_B = d_A \times d_B \times G = d_B \times d_A \times G = d_B \times Q_A$. The protocol is secure because nothing is disclosed (except for the public keys), and no node can derive the private key of the other nodes, unless the Elliptic Curve Discrete Logarithm Problem (ECDLP) was solved.

4. Implementation

In this section, we describe our implementation of digital signature and key agreement on Intel Mote2 and Laptop PC. To show the effectiveness of our system implementation, we compare the implementation on the Intel Mote2 and on the Laptop PC. In addition, we also compare the experimental results of using ECDSA technique and RSA-based one.

4.1 The Architecture of Intel Mote2 IEEE802.15.4 WSN

The Intel Mote2 is a sensor device which has a set of features especially for constructing a WSN and its supporting applications, such as industrial vibration, structural monitoring, acoustic and visual monitoring. The main processor with PXA271 XScale platform runs up to 416Mhz. It is also equipped with IEEE802.15.4 wireless sensor network. It also exposes a basic sensor board. Intel

Mote2 has internal 256kB SRAM, SDRAM of 30Mbytes and flash memory of 30Mbytes. The PXA271 XScale processor uses the SRAM for both instruction and data which is targeted for mobile ad-hoc routing and bridging functionalities. All working functions are supported by the *Openembedded Linux* operating system resides in the flash memory. This operating system is licensed under the GPL that includes a standard C library, many applications supported, libraries, tools, and root file systems. Table 1 shows the detail specifications of Intel Mote2 device used in this implementation. For detail specifications can be found in [10].

Table 1. Intel Mote2 specifications used in the experimental.

Hardware	PXA271 XScale processor 416MHz, 256kB SRAM, 30MB SDRAM, 30MB flash memory, IEEE802.15.4.
Software	*Openembedded* Linux O/S kernel-2.6.29, gmplib-5.0.5, arm-linux-gcc-3.4.3.

4.2 ECDSA and ECDH Implementation on Intel Mote2

We embed the CM method variant which has been presented in [1] to the Intel Mote2. At first, we determine a suitable elliptic curve parameters a and b to construct an elliptic curve. Furthermore, we perform the following basic steps:

I. Choose a suitable discriminant value D to find out Hilbert and Weber polynomials.
II. Pick randomly a prime number p. This number is selected if and only if the equation $4p = u^2 + Dv^2$ can be solved (find the two integer values u and v). Otherwise, repeat to select other p.
III. Pick the order of elliptic curve m if one of either this equation $m = p + 1 - u$ or $m = p + 1 + u$ is satisfied. Otherwise, repeat step I. In addition, m also has to be satisfied the following requirements:
   A) $m \neq p$.
   B) $\forall k, 1 \leq k \leq 20, p^k \not\equiv 1 \pmod{p}$.
   C) m is a big number (at least 2^{160}).
IV. Based on the first elliptic curve and the twist elliptic curve, compute the roots of Weber polynomials by using j-invariants.
V. Select one from the two elliptic curves that satisfies equation $mP = O$. P is selected repeatedly until one of P on each elliptic curve fulfills $mP \neq O$.

We adopt ECC-LIB [8] and GMP [9] libraries in our implementation, due to the libraries provide high precision calculations both in integer and floating point arithmetic operations. As an embedded device, Intel Mote2 also has a limited memory, thus adopting the features in ECC-LIB and GMP libraries for constructing CM method based elliptic curve is suitable for Intel Mote2.

4.3 ECDH Key Agreement Protocol

Again ECC-LIB and GMP libraries are used to construct the implementation of ECDH key agreement protocol. Figure 1 shows the mechanism of implemented ECDH protocol in our secure WSN system.

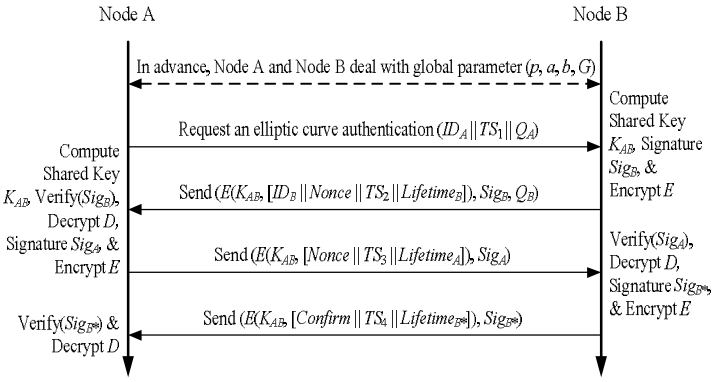

Figure 1. Proposed ECDH key agreement protocol.

Figure 1 demonstrates a key agreement protocol between Node A and Node B. In advance, two nodes have dealt with the global parameter (p, a, b, G), where a and b are elliptic curve coefficients, prime number p, and G is the base point which consists of coordinate values to construct an elliptic curve. The flow of this protocol is as follows:

I. Upon receiving a message from Node A that includes identity of Node A ID_A, timestamp TS_1, and Node A's public key $Q_A = d_A \times G$ (where d_A represents Node A's private key), Node B generates a random *Nonce*, his public key $Q_B = d_B \times G$ (d_B is Node B's private key), his timestamp TS_2, and $Lifetime_B = TS_1 - TS_2$. Here, the used of *Nonce*, timestamps, and *Lifetime* is to prevent reply attacks. Furthermore, he computes a shared key $K_{AB} = d_B \times Q_A$ and signs the message which includes *Nonce*, TS_2, $Lifetime_B$, and ID_B by using his private key. Then, the shared key K_{AB} is used for encrypting the message and sends the encoded message along with signature Sig_B and Q_B to Node A.
II. Node A computes a shared key $K_{AB} = d_A \times Q_B$, decrypts the encoded message into plaintext message which consist of *Nonce*, TS_2, $Lifetime_B$, and ID_B information, such that he can verify signature Sig_B. Furthermore, he checks whether $Lifetime_B = TS_1 - TS_2$ or not. The process will be continued to the next step if the equation holds. Otherwise, a reply attack may occur and the process is aborted. Then, Node A prepares a new timestamp TS_3, calculates $Lifetime_A = TS_2 - TS_3$, encrypts them together with *Nonce* received from Node B by using his shared key K_{AB}, and signs the message into signature Sig_A. Then, he responds Node B by sending a couple of ($E(K_{AB}$, [*Nonce* || TS_3 || $Lifetime_A$]), Sig_A) to Node B. Where E denotes an encryption algorithm and || is the concatenation of string in the message.

III. Upon receiving a response message from Node A, by using his own K_{AB}, Node B decrypts the message and proves that $Lifetime_A = TS_2 - TS_3$. Sig_A is verified successfully if only if the equation holds and the received *Nonce* is equal to his own *Nonce*. Otherwise, a reply attack may occur and the process is aborted. Furthermore, Node B prepares a new timestamp TS_4, calculates a new lifetime $Lifetime_{B*} = TS_3 - TS_4$, and defines a confirmation message *Confirm* for accepting or rejecting the connection. He signs the *Confirm* together with TS_4 and $Lifetime_{B*}$ into Sig_{B*}. Then, he sends Sig_{B*} along with encoded message to Node A.

IV. Finally, Node A verifies received message from Node B and decrypts it. He ensures that $Lifetime_{B*} = TS_3 - TS_4$ and the *Confirm* message is accept confirmation to establish a connection.

4.4 Data Confidentiality Approach

Data confidentiality in our implementation is achieved by encrypting every message exchanged in every communication session by sender Node. On the other hand, the encoded message is decrypted into original message by recipient Node. This approach is implemented both in key exchange and data exchange phases. The detail of our implementation is briefly described by the following steps of encryption scheme for a message *M*.

I. Generate a one-time key pair (*R*, *c*) from the global EC parameters and let *R* as a point on the curve.

II. Pick the *x* component of $K = c \times B$ as a string *X*, where *B* is the public key of recipient.

III. Generate a mask *Y*. This is the same number of bytes as message *M* that uses the string *X* with the Mask Generation Function (MGF).

IV. The final step is concatenation of the point *R* with the encrypted message, whereas the encrypted message is the result of XOR between the mask and the message.

V. The process of message decryption to retrieve original message is performed by generating the mask and XOR with the encrypted message.

5. Experimental Results

In this section, we explain the experimental results of our implementation. At first experiment, we test the ECDSA and ECDH with elliptic curves over F_p and CM method on Laptop PC with the specifications: AMD Dual-Core processor 1.6GHz, 2GB RAM running on Ubuntu Linux kernel-2.6.35 with gcc-4.4 GNU C Compiler and GMP-5.0.5 Library.

Table 2. Comparisons of the total processing time in RSA-based and ECC-based Digital Signature Algorithm (DSA).

	RSA ECC	RSA ECC	RSA ECC	RSA ECC
Bit length	1024 160	1536 192	2048 224	3072 256
Signature generation time (ms)	1,182.93 6.81	3,663.65 8.19	7,076.35 11.45	17,223.12 13.87
Signature verification time (ms)	582.69 10.64	1,738.40 16.36	3,451.21 17.67	9,270.83 21.29

Table 2 shows the total time comparisons of signature generation and verification between RSA-based and ECC-based digital signature algorithm. The total time of signature generation varies from 7 ms to 14 ms, and signature verification varies from 11 ms to 21 ms in ECC-based algorithm. Meanwhile, the total time of signature generation varies from a second to 17 seconds, and verification time varies from 580 ms to 9 seconds in RSA-based one. Here, the security level of 160-bit, 224-bit, and 256-bit length in ECC equal to 1024-bit, 2048-bit, and 3072-bit length in RSA, respectively. This implies that the cost of computation time in elliptic curve technique is much more efficient than in RSA-based one.

Table 3. Comparisons of signature size in RSA-based DSA and ECDSA.

	RSA-DSA ECDSA	RSA-DSA ECDSA	RSA-DSA ECDSA	RSA-DSA ECDSA
Bit length	1024 160	1536 192	2048 224	3072 256
Signature size (Bytes)	9,868 96	14,799 113	19,715 134	24,482 146

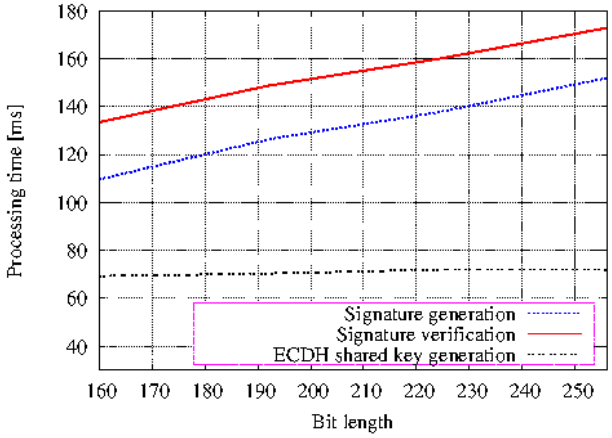

Figure 2. Total processing times on Intel Mote2.

We also compared the overhead of RSA-based DSA and ECDSA algorithm in term of signature size. The comparison of signature size is shown in Table 3. For the bit-length of security level from 160-bit to 256-bit in ECDSA, the signature size varies from 96 to 146 Bytes only. On the other hand, RSA-based DSA suffers from a big size of signature. It varies from 10 to 24Kbytes in the variety of security level

from 1024 to 3072 bit length. This good feature of ECDSA can efficiently be implemented into low end performance devices, such as embedded devices including Intel Mote2.

Table 4. Total processing time of message encryption and message decryption.

Bit length	160	192	224	256
Message encryption time (ms)	1.605	1.608	1.610	1.607
Message decryption time (ms)	1.098	1.103	1.101	1.099

Based on the reality that ECC is much more efficient than RSA, we have ported ECDSA and ECDH schemes into Intel Mote2 embedded device. Figure 2 shows the total time of signature generation, signature verification, and ECDH shared key generation. We vary the bit length of cryptosystem from 160-bit to 256-bit in our measurement. The total processing time of signature generation varies from 110 ms to 152 ms. Whereas, the total time of signature verification varies from 134 ms to 173 ms, and total time of ECDH shared key generation is relatively constant at 70 ms. In addition, our approach to guarantee data confidentiality, we implemented message encryption and message decryption. Table 4 shows the total processing time of message encryption and decryption algorithm. The total time of encryption and decryption processes are relatively constant for any bit-length of security level. They are only 1.6 ms and 1.1 ms for message encryption and decryption, respectively.

6. Security Issues

The proposed ECDH protocol is secure because nothing is disclosed (except for the public keys), and no node can derive the private key of the other nodes, unless the Elliptic Curve Discrete Logarithm Problem (ECDLP) was compromised. This is because all nodes are only allowed to have public keys of other nodes. All participant nodes hide their private keys by embedding them into shared key K_{AB}. Let consider to the following equation:

$$K_{AB} = d_A \times Q_B = d_A \times d_B \times G = d_B \times d_A \times G = d_B \times Q_A \quad (6)$$

We introduced the use of timestamps and packet lifetime in every session of communication or transaction between two nodes, and the use of random nonce to prevent a kind of reply attacks by un-authorized nodes. This protocol security is also supported by the use of digital signatures to provide a mechanism of mutual authentication scheme. We also encrypted every message exchanged in every communication session to guarantee the data confidentiality.

7. Conclusion

In this paper we have presented an implementation of elliptic curve digital signature and key agreement on an IEEE802.15.4 WSN device. The experimental results show the practicality of our secure WSN system. The total time of signature generation, verification, and ECDH shared key generation consume low cost of computation time. They only consume computation time within 500 ms with the overhead of signature size within 200 Bytes. Our future works include the implementation of secure data exchange over WSN between nodes and between nodes and the gateway; and the consideration of multi-hops communications among nodes over WSN.

References

[1] E. Konstantinou, Y. C. Stamatiou, and C. Zaroliagis, “Efficient Generation of Secure Elliptic Curves”, International Journal of Information Security, Springer-Verlag, pp. 47-62, 2006.

[2] D. J. Malan, M. Welsh, and M. D. Smith, “A Public-key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography”, International Conference on Sensor and Ad Hoc Communications and Networks, pp. 71-80, 2004.

[3] N. Constantinescu, “Authentication Protocol Based on Elliptic Curve Cryptography”, Annals of the University of Craiova, Mathematics and Computer Science Series, Volume 37(2), pp. 83-91, 2010.

[4] A. Liu and P. Ning, “TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks”, International Conference on Information Processing in Sensor Networks, pp. 245-256, 2008.

[5] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede, “Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks”, Proceedings of the Third European conference on Security and Privacy in Ad-Hoc and Sensor Networks, Springer-Verlag Berlin, Heidelberg, pp. 6-17, 2006.

[6] X. Huang, D. Sharma, M. Aseeri, and S. Almorqi, “Secure Wireless Sensor Networks with Dynamic Window for Elliptic Curve Cryptography”, Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International, pp. 1-5, 2011.

[7] I. Blake, G. Seroussi, and N. Smart, “Elliptic Curves in Cryptography”, London Mathematical Society Lecture Note Series 265. Cambridge University Press, 1999.

[8] http://www.ceid.upatras.gr/faculty/zaro/software/ecc-lib/ – ECC-LIB Library [Accessed: July 16, 2012].

[9] http://www.gmp.org – GNU Multiple Precision (GNUMP) Library [Accessed: July 16, 2012].

[10] http://ubi.cs.washington.edu/wiki/index.php/IMote2 – Intel Mote2 documentation [Accessed: July 18, 2012]

.