A New Security Protocol Based on Elliptic Curve Cryptosystems for Securing Wireless Sensor Networks

Seog Chung Seo, Hyung Chan Kim, and R.S. Ramakrishna

Department of Information and Communications,
Gwangju Institute of Science and Technology (GIST),
1 Oryong-dong, Buk-gu, Gwangju 500-712, Rep. of Korea
{gegehe, kimhc, rsr}@gist.ac.kr

Abstract. In this paper, we describe the design and implementation of a new security protocol based on Elliptic Curve Cryptosystems (ECC) for securing Wireless Sensor Networks (WSNs). Some public-key-based protocols such as TinyPK and EccM 2.0 have already been proposed in response. However, they exhibit poor performance. Moreover, they are vulnerable to man-in-the-middle attacks. We propose a cluster-based Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA) for efficiency and security during the pairwise key setup and broadcast authentication phases, respectively. We have implemented our protocol on 8-bit, 7.3828-MHz MICAz mote. The experimental results indicate the feasibility of our protocol for WSNs.

1 Introduction

Wireless sensor networks (WSNs) have been proposed for a wide variety of applications such as emergency medical care, vehicular tracking, and building monitoring systems. Because these sensor networks are composed of small, resource-constrained sensor nodes and are deployed in harsh, unattended environments, some combination of authentication, integrity, and confidentiality are required for reliable and lasting network communications. However, achieving security in WSNs is a challenging job in that the absence of any supervisor makes the application of conventional security protocol infeasible for WSNs. Furthermore, the limited resources at the sensor nodes are targets of Denial-of-Service (DoS) attacks. Therefore, it is essential to build a security protocol taking into account the inherent characteristics of WSNs such as low computing power, low bandwidth, high susceptibility to physical capture, and dynamic network topology [1]. Besides, the security protocol should cope with a number of threats including eavesdropping, injecting malicious messages, and node compromise.

Most of the existing security protocols are based on symmetric key. The symmetric key system provides efficient cryptographic operations of encryption and decryption. However, it is not appropriate for setting up pairwise keys and broadcast authentication because it generates heavy traffic and involves complex

architecture. Moreover, symmetric-key-based security protocols are vulnerable to node compromise. Some public-key-based protocols such as TinyPK [8], EccM 2.0 [7] and Blass' [4] have addressed these issues. However, they exhibit poor performance. They are vulnerable to man-in-the-middle attack.

This paper presents a new security protocol based on the ECC. Our protocol consists of mainly two phases: pairwise key setup and broadcast authentication. We propose a cluster-based ECDH and ECDSA for security of key agreement and efficiency of broadcast authentication, respectively. Our contributions are summarized below:

- The proposed ECDH provides the generic key agreement mechanism which does not require any knowledge of network topology. The proposed scheme can prevent the man-in-the-middle attack by verifying the signature of the public key from other nodes. The built pairwise keys are used to distribute the cluster key, a key that is common to all the cluster elements.
- The cluster-based ECDSA offers efficient broadcast authentication which can reduce the overheads on network-wide verification. This is why only the clusterheads are responsible for verifying broadcast messages in our protocol.
- We have implemented the proposed protocols on the 8-bit, 7.3828-MHz MICAz mote [11] which is one of the most popular sensor motes. The experimental results testify to the viability of our protocol for WSNs. Furthermore, the proposed protocol outperforms existing ECC-based protocols over $GF(2^p)$ for WSNs with the aid of efficient algorithms such as width-w Mutual-Opposite-Form (wMOF) [14] and shamir's trick [13].

The remainder of this paper is organized as follows: In Section 2 we take a look at related work. Section 3 describes the proposed security protocols. Security is analyzed in Section 4. In section 5 we present implementation and experimental results. Conclusions are presented in Section 6. The details of main idea for efficient implementation of ECDH and ECDSA can be found in the Appendix.

2 Related Work

As WSNs are becoming attractive in ubiquitous computing environments, security of WSNs is understandably attracting attention. Many security protocols have been proposed to date. They are divided into two main categories: symmetric-key-based protocols and public-key-based protocols. The security protocols taking advantage of symmetric keys such as SPIN assume the complete impracticability of public key system due to their high computational overhead [2, 3]. However, symmetric key systems are not as versatile as public key system, so that they complicates the design of security architecture such as key distribution, and broadcast authentication. The complicated security architecture generates heavy network traffic. Currently, many researchers are attempting to apply the public key cryptosystem for securing WSNs.

Watro et al. presented the TinyPK for authentication and key agreement between 8-bit MICA2 motes [8]. The TinyPK makes use of RSA-based Diffie-Hellman for key agreement. However, TinyPK takes more than 2 minutes to

establish a pairwise key between two sensor nodes. Kumar et al. [6] developed a communication protocol employing ECDH key exchange. Their work involves optimal extension fields where field multiplication is quite efficient. However, it is vulnerable to the Weil descent attack. The work of Wander et al., compared the performance of RSA and ECC on the Atmega128L processor in respect of energy consumption [12]. They tried to integrate the RSA and ECC into SSL handshake to provide mutual authentication. Gaubats et al. have compared Rabin's scheme, NtruEncrypt, and ECC on a low power device in [5]. The results of experiments show that the ECC is more appropriate for WSNs than Rabin's scheme and NtruEncrypt. The EccM 2.0 has implemented ECDH key agreement protocol on MICA2 mote [7]. In the EccM 2.0, the established pairwise key between two sensor nodes is used for the symmetric key of TinySec [9] which is the link-layer security architecture in TinyOS [10]. Blass and Zitterbart have also analyzed the performance of the ECDH, ECDSA and El-Gamal on MICA2 mote [4].

3 Proposed Protocol

3.1 Assumptions and Preliminaries

- The sensor network consists of several clusters. They are interconnected by gateway nodes which are involved in more than two clusters.
- Clusterheads are computationally very powerful and have larger storage capacity than normal sensor nodes.
- Each sensor node has one public key and its corresponding signature signed by BS's private key. BS's public key is also stored in every node before deployment. All public keys of clusterheads are stored in the BS.
- N_A is the normal node named A in a cluster. N_C and N_G represent the clusterhead and gateway node, respectively. K_A is the private key of N_A. P_A is public key of N_A and S_{PA} is the signature of the public key in N_A. K_{AB} is the pairwise key between N_A and N_B. K_C refers to the cluster key. $E_{K_{AB}}(m)$ indicates that a message m is encrypted with pairwise key K_{AB} shared by node A and node B. The concatenation of messages is expressed with the operator $||$. G is the global point in ECC operation.

3.2 Pairwise Key Establishment

We combine the ECDH key agreement and the clustering scheme. We can categorize the pairwise key setup into four types. This is illustrated in Fig. 1. The categories involve keys between clusterhead and normal nodes adjacent to it (1), between normal nodes (2), between either gateway node and clusterhead or gateway node and normal node (3), and between clusterheads (4). In the process of pairwise key setup between normal nodes and clusterhead, the validity of normal nodes which are adjacent to clusterhead should be verified because they can modify the data from other normal nodes or generate malicious data directly.

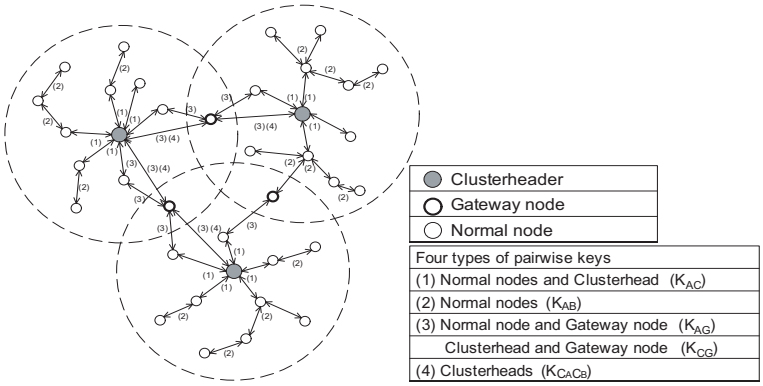

Fig. 1. Pairwise key establishment process

Furthermore, the legality of gateway nodes should be examined to prevent attackers from pretending a valid node. Otherwise, the attacker can control as to where the gathered data should go by impersonating the gateway node.

In fact, the legitimacy of the clusterhead should be inspected by other normal nodes because it plays a pivotal role in gathering the data and forwarding it. Some of the normal nodes which are close to the clusterhead can investigate the identity of the clusterhead via the signature of its public key.

The sensor nodes make use of the established pairwise key as a symmetric key of TinySec [9] which is a link-layer security architecture in TinyOS [10]. In fact, our protocol utilizes the TinySec so that it can provide node-to-node confidentiality and authentication.

(i) *Between clusterhead and normal nodes.*
1. Normal node (N_A) sends a pair of public keys ($P_A = G * K_A$) and its signature (S_{PA}) signed by BS's private key to the clusterhead (N_C).
$$N_A \longrightarrow N_C\text{: } P_A||S_{PA}$$
2. The clusterhead verifies the validity of the public key using BS's public key. If the signature is authentic, the clusterhead sends its public key to N_A. Otherwise, it registers N_A as a malicious node.
3. If the signature proves to be valid, they can calculate the common pairwise key ($K_{AC} = K_{CA} = P_A * K_C = P_C * K_A = G * K_A * K_C$).
4. After completing the pairwise key setup between the normal nodes and the clusterhead, the latter can distribute the cluster key (K_C) which is the commonly shared key of a cluster. The cluster key is encrypted with pairwise keys between each normal node and clusterhead and is distributed to each normal node.
$$N_C \longrightarrow N_A\text{: } E_{K_{CA}}(K_C)$$

(ii) *Between two normal nodes.*
1. N_A sends its public key ($P_A = G * K_A$) to N_B.
2. N_B sends its public key ($P_B = G * K_B$) to N_A.

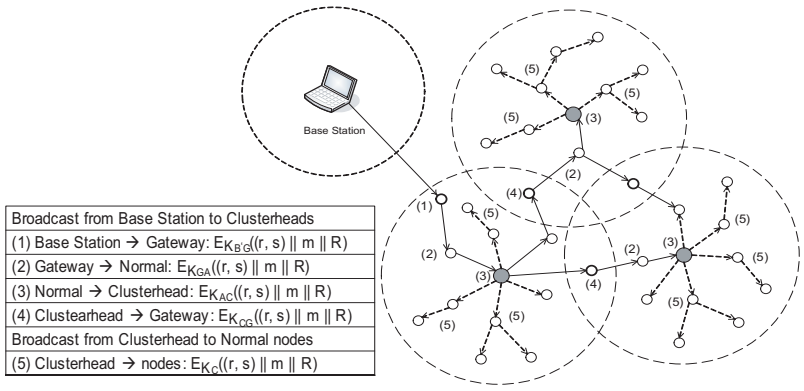

Fig. 2. Broadcast authentication process

3. They can calculate the common pairwise key ($K_{AB} = K_{BA} = P_A * K_B = P_B * K_A = G * K_A * K_B$).

(iii) *Between gateway node and (clusterhead or normal node).*

1. The gateway node (N_G) sends (P_G, S_{PG}).
2. If a clusterhead receives the message from the gateway node, it can verify the validity of the public key immediately.
3. If a normal node gets the message, it forwards the pair to the clusterhead to examine it. The clusterhead returns the result of the verification.

$$N_G \longrightarrow N_A\text{: } P_G||S_{PG}$$
$$N_A \longrightarrow N_C\text{: } E_{K_{AC}}(P_G||S_{PG})$$
$$N_C \longrightarrow N_A\text{: } E_{K_{CA}}(Valid\ or\ Not)$$

4. If the signature is authentic, the remaining steps are same as before.

(iv) *Between clusterheads.*

Assume that two clusterheads N_{C_A} and N_{C_B} try to set up a pairwise key.

1. A clusterhead (N_{C_A}) sends (P_{C_A}, S_{PC_A}) to the gateway node (N_G).
$$N_{C_A} \longrightarrow N_G\text{: } E_{K_{GC_A}}(P_{C_A}||S_{PC_A})$$
2. A gateway node (N_G) forwards the pair to another clusterhead (N_{C_B})
$$N_G \longrightarrow (N_{C_B})\text{: } E_{K_{GC_B}}(P_{C_A}||S_{PC_A})$$
3. If the signature is valid, the clusterhead (N_{C_B}) also sends (P_{C_B}, S_{PC_B}) to the clusterhead (N_{C_A}). The same procedure is followed to verify the validity of the clusterhead (N_{C_B}).
4. After authenticating mutually, the clusterheads can compute the common pairwise key.

3.3 Broadcast Authentication

In WSNs, the BS broadcasts its command or query to sensor nodes. If a broadcast authentication mechanism is not provided, an attacker can impersonate the BS and execute a kind of DoS attack by generating heavy traffic over the network.

Similarly, the clusterheads broadcast their aggregated data from normal nodes to the BS. Unless there is provision for authentication, it is possible for attackers to send malicious or bogus data to the BS. For broadcast authentication in WSNs, μTESLA has been proposed in [2]. However, all the sensor nodes must be synchronized with the BS in μTESLA. This constraint results in decreased lifetime of the sensor network. Furthermore, the delayed disclosure of the authentication keys causes time delay in message authentication in μTESLA. We can provide efficient broadcast authentication mechanism by exploiting the ECC, especially the ECDSA, due to its even smaller key size as compared with other digital signature algorithms. However, the overhead of verification in ECDSA is almost twice as large as that of signing. If all the sensor nodes in the network verify the broadcast messages from the BS, considerable energy is consumed. This is unacceptable in view of the limited resource of the entire network. Therefore, we propose a cluster-based ECDSA so that we may reduce the overhead of verification for broadcast authentication. Actually, only the clusterheads are responsible for verifying the broadcast message in our mechanism. This results in a sharp fall in resource consumption.

In Section 3.1, we have assumed that BS's public key is stored in each sensor node and the public keys of the clusterheads are also maintained by the BS. The public key of the BS is utilized by the clusterheads for verifying the signature of the broadcast message from the BS. Similarly, the public keys of the clusterheads are applied to verify the messages from the clusterheads to the BS.

Broadcast from Base Station to Clusterheads.
The process is depicted in Fig. 2. In the figure, the BS broadcasts a message to the clusterheads (1 through 4). The message is encrypted by the pairwise keys of the concerned nodes for providing confidentiality. The details are given below.

(i) *Signing the broadcast message.*

1. The BS generates the signature (r, s) based on the message (m) and its private key (d). The nonce value (R) is used to prevent replay attack. The (r, s) is computed as below:
   $r = x_1 \bmod n$, $kP = (x_1, x_2)$, $k \in [1, n-1]$, P is a point on curve
   $s = k^{-1}\{h(m||R) + dr\} \bmod n$, where h is SHA-1, n is large prime.
2. The BS $(N_{B'})$ sends a pair of signature (r, s) and a message (m) with random nonce (R) to gateway nodes for delivering it to clusterheads. The gateway nodes forward it to the clusterheads.
   $$N_{B'} \longrightarrow N_G\text{: } E_{K_{B'G}}((r, s)||m||R)$$
   $$N_G \longrightarrow N_C\text{: } E_{K_{GC}}((r, s)||m||R)$$

(ii) *Verifying the broadcast message.*

1. When a clusterhead receives the signed broadcast message, it verifies the message by comparing (v) and (r). In addition, it ignores the duplicate messages by checking the nonce value, which results in energy efficiency. The procedure for computing value (v) is given below:
   $v = x_1 \bmod n$, $u_1 * G + u_2 * P_{B'} = (x_1, y_1)$,
   $u_1 = \{h(m||R) * w\} \bmod n$, $u_2 = r * w \bmod n$, $w = s^{-1} \bmod n$.

2. If the calculated (v) is same as the received (r), the clusterhead accepts this message. And then it broadcasts a local query encrypted with the cluster key (K_C) to normal nodes in a cluster.
$$N_C \longrightarrow N_A\colon E_{K_C}(m)$$
3. Normal nodes begin on the assigned work and return the results.

Broadcast from Clusterheads to Base Station.
This procedure in Fig. 2 is reversed.

(i) *Signing the broadcast message.*
1. A clusterhead collects data from normal nodes in a cluster. It signs the gathered data using its private key (The signing procedure is same as above). For prevention of replay attack, the nonce value (R') is used.
2. The clusterhead sends a pair of signature (r', s') and data (m') to a gateway node.
$$N_C \longrightarrow N_G\colon E_{K_{CG}}((r', s')||m'||R')$$
3. The gateway node forwards the pair to the BS through other clusters.
$$N_G \longrightarrow N_{B'}\colon E_{K_{GB'}}((r', s')||m'||R')$$

(ii) *Verifying the broadcast message.*
1. The BS can verify the message from clusterheads because it maintains the public keys of clusterheads (The verification procedure is same as above). It also achieves high energy efficiency by rejecting the duplicate messages through checking the nonce value.
2. If the signature proves to be innocent, the BS accepts this message.

4 Security Analysis

We analyze the proposed key setup protocol and broadcast authentication mechanism with regard to essential security properties such as confidentiality, integrity, authentication, and node compromise attack.

Confidentiality and Integrity. In a process of pairwise key setup, even if an attacker can eavesdrop on the information exchange such as the nodes' public key, the secret pairwise key continues to be secure. This is why an attacker must solve the $ECDLP$ to gain the pairwise key. After completing the process, the nodes use the pairwise key for a symmetric key in TinySec [9]. The TinySec provides efficient node-to-node confidentiality and authentication. Furthermore, broadcast messages from BS are encrypted by these pairwise key. Therefore, our protocol ensures confidentiality and integrity.

Authentication. We categorize the pairwise key setup into four types and then require that the concerned nodes verify each others' signature so as to thwart man-in-the-middle attack. For example, the identity of the clusterheads, the gateway nodes, and the normal nodes that are close to the clusterhead is examined because they play principal roles in our protocol. Furthermore, the BS broadcasts messages signed with its private key. In both cases, attackers cannot

forge the signature of the public key because it is signed by the BS's private key. Therefore, the proposed protocols provide authentication mechanism through the process of verifying the signature.

Node Compromise. Node compromise is a central attack that can destroy the entire mechanism. An attacker can examine the secret information and the running code by compromising a node. In our protocol, nodes maintain minimal information such as their own public/private key pair and the corresponding signature. Therefore, if an attacker compromises t nodes, the information about the $(t+1)^{th}$ node remains out of reach. In other words, the attacker must solve the $ECDLP$ in order to find out the private key of the $(t+1)^{th}$ node. This is an advantage of our protocol over symmetric-key-based protocols [2, 3] which are vulnerable to node compromise attack.

5 Implementation and Performance Evaluation

We have implemented the proposed protocol on an 8-bit, 7.3828-MHz MICAz mote [11]. For emphasizing the feasibility of our protocol in WSNs, we concentrate on efficient implementation of the proposed pairwise key establishment protocol and broadcast authentication mechanism based on the ECC rather than the cluster forming or the routing protocol.

5.1 Implementation Details

Elliptic Domain Parameters and Selection of Key Size. We make use of the recommended 113-bit Elliptic Curve Domain Parameters ($sect113r1$ of [15]) over $GF(2^p)$. Although the selected 113-bit key is shorter than NIST's recommended key size (163-bit), it is more in tune with the life time of the sensor nodes. In fact, the largest broken key size has 109-bit, and it took more than seventeen months with ten thousands computers.

Elliptic Scalar Multiplication. The ECDH and ECDSA are related to compute the scalar multiplication which computes ($Q = dP$) for a given point P and a scalar d. The performance of ECDH and ECDSA depends on the number of additions in the scalar multiplication. The number of additions should be reduced for efficiency. We have developed a scalar multiplication algorithm using wMOF which is a kind of signed representation. We can represent the equivalent value with reduced number of additions with the aid of the wMOF [14]. Even though the number of additions is reduced, an extended window size requires additional memory for precomputed points. Through experiments, we have found that the optimal window size is 3 on MICAz mote with regard to memory and efficiency. The verification procedure in ECDSA involves scalar multiplication of multiple points such as $vP + uQ$. If the sensor nodes are required to verify the signature quickly, the term $vP + uQ$ should be computed efficiently. Inspired by shamir's trick [13], we perform simultaneous elliptic scalar multiplication using wMOF. The details of our algorithm can be found in the Appendix.

Table 1. Performance of computing pairwise key in ECDH

	Time	Energy	CPU Utilization
EccM 2.0 [7]	22.72 sec	0.54518 Joules	1.6783×10^8 cycles
Blass' [4]	17.28 sec	0.41472 Joules	1.2767×10^8 cycles
Proposed	5.796 sec	0.13910 Joules	0.4282×10^8 cycles

Table 2. Performance of verification in ECDSA

	Time	Energy	CPU Utilization
EccM 2.0 [7]	23.63 sec	0.56712 Joules	1.7458×10^8 cycles
Blass' [4]	24.17 sec	0.58008 Joules	1.7857×10^8 cycles
Proposed	7.367 sec	0.17681 Joules	0.5443×10^8 cycles

5.2 Performance Evaluation

We compare our work with other implementations over $GF(2^p)$ using the same key size. Actually, the EccM 2.0's key is 163 bits long. We lowered the key size of EccM to 113-bit for a fair comparison. In Table 1, we present the performance of our pairwise key setup protocol based on ECDH and compare it with other existing implementations in aspect of time, energy, and CPU utilization. By signed representation of the multiplier, the proposed protocol can achieve better performance than other implementations. Furthermore, by preloading the public key and its signature on each sensor node before deployment, the sensor nodes do not have to compute their public key, which lowers the overhead of the pairwise key setup process. In fact, it takes only 5.796 sec for two normal nodes to share a pairwise key. Clusterheads can establish the pairwise key more rapidly because they have higher computational power than normal nodes.

Table 2 also presents the performance of broadcast authentication based on the ECDSA verification. We could reduce the verification overhead by using shamir's trick based on wMOF. Actually, this overhead is larger than that of computing a pairwise key. However, in our protocol, only the clusterheads or BS verifies the signature of the broadcast message. Therefore, they can complete this operation even within a period of 7.367 sec.

The experimental results show that our protocol outperforms existing ECC-based protocols such as EccM 2.0 [7] or Blass' [4]. Furthermore, it implies the feasibility of our protocol for WSNs.

6 Conclusion

For securing the WSNs, we propose pairwise key establishment and broadcast authentication protocol. By clustering the entire network, we can categorize the

pairwise key setup into four types involving the concerned members. In our protocol, the sensor nodes can establish the pairwise key efficiently with ECDH over an insecure channel. Furthermore, the proposed mechanism can prevent the man-in-the middle attack by verifying the other node's signature. Through the application of established pairwise key to Tinysec, our protocol provides node-to-node confidentiality and authentication. In the proposed mechanism, the clusterheads are required to verify the signature of the broadcast messages, thereby preventing the attackers from impersonating the BS. This prevents DoS attacks. Through experiments on the 8-bit, 7.3828-MHz MICAz mote, we provide performance analysis of our protocol. The feasibility of the proposed protocol for WSNs is borne out by the above analysis.

Acknowledgement. The authors would like to thank Dr. Jong-Phil Yang and anonymous reviewers for their helpful comments and valuable suggestions. This research was supported by Brain Korea 21 of Ministry of Education of KOREA.

References

1. Perrig, A., Stankovic, J. and Wagner, D.: Security in Wireless Sensor Networks. Comm. ACM (2004) 47(6):53–57
2. Perrig, A., et al.: SPINS: security protocols for sensor networks. Wireless Networking. (2002) 8(5):521–534
3. Du, W., et al.: A pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. Proc. 10th ACM Conf. Comp. and Comm. Security. (2003) 42–51
4. Blass, E.O., Zitterbart, M.: Efficient Implementation of Elliptic Curve Cryptography for Wireless Sensor Networks. (2005)
5. Gaubatz, G., et al.: State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks. Proc. of 3th IEEE Conf. on Pervasive Comp. and Comm. (2005) 146–150
6. Kumar, S., et al.: Embedded End-To-End Wireless Security with ECDH Key Exchange. Proc. of IEEE Conf. On Circuit and Systems. (2003)
7. Malan, D.J., Welsh, M., and Smith, M.D.: A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography. Proc. of IEEE Conf. on Sensor and Ad Hoc Comm. and Networks. (2004)
8. Watro, R., et al.: TinyPK: Securing Sensor Networks with Public Key Technology. Proc. of SASN'04 (2004) ACM Press 59–64
9. Karlof, C., Sastry, N., and Wagner, D.: TinySec: Link Layer Security Architecture for Wireless Sensor Networks. Proc. of SenSys'04 (2004) 162–175
10. TinyOS forum. Available at "http://www.tinyos.net/".
11. MICAz Hardware Description Available at "http://www.xbow.com/Products".
12. Wander, A.S., et al.: Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks. Proc. of IEEE Conf. on Pervasive Comp. and Comm. (2005)
13. Hankerson, D., Hernandez, J.L.: Software Implementation of Elliptic Curve Cryptography over Binary Fields. Proc. of CHES 2000. LNCS 1965 (2000) 1–24
14. K. Okeya, et al.: Signed Binary Representation Revisited. Proc. of CRYPTO 2004. LNCS 3152. (2004) 123–139
15. Certicom Research: SEC 2-Recommended Elliptic Curve Domain Parameters.

Appendix

This section presents main idea for efficient implementation of ECDH and ECDSA by describing the proposed scalar multiplication algorithm. Algorithm 1 computes a scalar multiplication which is a dominant computation in ECC. To generate proper wMOF code on the fly, we have developed algorithm 2. It generates appropriate wMOF code from the MOF using a kind of weighted sum.

Our algorithms provide efficiency in aspect to both computation and memory. In fact, the scalar multiplication consumes only $\mathcal{O}(w)$ bits for signed representation of scalar multipliers. Furthermore, with wMOF code, we could reduce the number of additions from $\mathcal{O}(\frac{n}{2})$ to $\mathcal{O}(\frac{n}{w+1})$ given n-bit binary string.

Algorithm 1 Scalar Multiplication Algorithm using wMOF

INPUT: a point P, window width w, $d = (d_{n-1}, ..., d_1, d_0)_2$, $R \leftarrow \mathcal{O}$
OUTPUT: product dP
$d_{-1} \leftarrow 0; d_n \leftarrow 0$; $i \leftarrow c + 1$ for the largest c with $d_c \neq 0$
Compute $P_i = iP$, for $i \in \{1, 3, 5, \ldots, 2^{w-1} - 1\}$
while $i \geq 1$ **do**
  $R \leftarrow ECDBL(R)$
  if $d_{i-1} = d_i$ **then**
    $i \leftarrow i - 1$
  else if $d_{i-1} \neq d_i$ **then**
    $GenerationwMOF(d_{i,\ldots,i-w}, index_i, code[w])$
    for $k \leftarrow 0$ to $w - 1$ **do**
      $R \leftarrow ECADD(R, code[k] * P)$
      if $k \neq w - 1$ **then**
        $R \leftarrow ECDBL(R)$
      end if
    end for
    $i \leftarrow i - w$
  end if
end while

Algorithm 2 Generation of $wMOF$: $GenerationwMOF$ (On the fly)

INPUT: w-bit binary strings, index, and w-byte array
OUTPUT: w-byte $wMOF$ code
$check \leftarrow true, multiplier \leftarrow 1, SUM \leftarrow 0, position \leftarrow 0$
for $m \leftarrow index - w, n \leftarrow w - 1$ to $index$ **do**
  if $check$ && $b_m - b_{m-1}$ **then**
    $position \leftarrow n$; $check \leftarrow false$
  end if
  $SUM \leftarrow SUM + multiplier * (b_m - b_{m-1})$
  $multiplier \leftarrow multiplier * 2$
  $n \leftarrow n - 1$; $wMOF[n] \leftarrow 0$
end for
$wMOF[position] \leftarrow SUM/2^{w-position-1}$
return $wMOF[w]$