Performance Evaluation of EC-ElGamal Encryption Algorithm for Wireless Sensor Networks

Soufiene Ben Othman[1], Abdelbasset Trad[1], Hani Alzaid[2], and Habib Youssef[1]

[1] UR PRINCE, ISITcom, Hammam Sousse University of Sousse, Tunisia
[2] Computer Research Institute, King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia
ben_oth_soufiene@yahoo.fr, abdelbasset.trad@isigk.rnu.tn, hmalzaid@kacst.edu.sa

Abstract. The rapid development in the Wireless Sensor Networks (WSNs) filed has allowed this technology to be used in many applications. In some of these applications, wireless sensor devices must be secured, especially when the captured information is valuable, sensitive, or for military usage. However, the implementation of security mechanisms on WSNs is a non-trivial task. Limitations in processing speed, battery power, bandwidth and memory constrain the applicability of existing cryptography algorithms for WSNs. The security of WSNs poses challenges because of the criticality of the data sensed by a node and in turn the node meets severe constraints like minimal energy, computational and communicational capabilities. Taking all the above said challenges energy efficiency or battery life time plays a major role in network lifetime. Providing security consumes some energy used by a node, so there is a need to minimize the energy consumption of any security algorithm that will be implemented in WSNs. As a solution, we apply an additive homomorphic encryption scheme, namely the elliptic curve ElGamal (EC-ElGamal) cryptosystem, and present the performance results of our implementation for the prominent sensor platform MicaZ mote.

Keywords: Wireless sensor network, security, elliptic curve ElGamal, Energy consumption analysis.

1 Introduction

Wireless Sensor Networks (WSN**s**) **have** emerged as an important new area in wireless technology. A wireless sensor network [1] is a distributed system interacting with physical environment. It consists of motes equipped with task-specific sensors to measure the surrounding environment, e.g., temperature, movement, etc. It provides solutions to many challenging problems such as wildlife, battlefield, wildfire, or building safety monitoring. A key component in a WSN is the sensor mote, which contains (a) a simple microprocessor, (b) application-specific sensors, and (c) a wireless transceiver. Each sensor mote is typically powered by batteries, making energy consumption an issue. Security is vital aspect in WSN applications.

B. Godara and K.S. Nikita (Eds.): MobiHealth 2012, LNICST 61, pp. 271–285, 2013.

The implementation of security policies is a complex and challenging issue because of resource constrained nodes. Short transmission distances reduce some of the security threats, but there are risks, for example, related to spoofing, message altering and replaying, and flooding and wormhole attacks [2]. It is important therefore to consider security solutions that guarantee data authenticity, freshness, replay protection, **integrity and confidentiality.** For secure communication in WSNs, efficient cryptographic algorithm suitable for WSNs environment is required. It is ideal to choose the most efficient cryptographic algorithm in all aspects; operation speed, storage and power consumption. However, since each cryptographic algorithm applied in WSNs has distinguished advantages, it is important to choose a cryptographic algorithm suitable for each environment WSNs are exploited.

The data encryption algorithms used in WSNs **are** generally divided into three major categories: symmetric-key algorithms, asymmetric-key algorithms, and hash algorithms. A number of papers, [1–2], have investigated using asymmetric-key algorithms in WSNs. However, the results they present reveal that despite the use of energy efficient techniques, such as elliptic curve cryptography or dedicated cryptography coprocessors, asymmetric-key algorithms consume more energy than symmetric-key algorithms. Hash functions, on the other hand, are typically used for verifying the integrity of the exchanged messages and may increase the transmission cost [3,4]. To prevent information and communication systems from illegal delivery and modification, message authentication and identification need to be examined through certificated mechanisms. Therefore, the receiver has to authenticate messages transmitted from the sensor nodes over a wireless sensor network. This is done through cryptography. It is a challenge to find out suitable cryptography for wireless sensor network due to limitations with respect to power, computational efficiency, and enough storage capabilities [2].

In this paper, an efficient implementation of EC-ElGamal scheme on MicaZ is presented in order to get better understanding of the usage of public encryption in WSNs. It is important to consider minimizing the code size of the implementation of ECC since sensor nodes keep in its memory other information required to make these sensors alive and functioning. For the data memory usage a similar motivation holds. In comparison to code and memory size, the execution time is not as critical as them. Therefore, this work focuses respectively on the optimization of code size, memory usage, and computation time. In comparison with similar implementations from the literature, the proposed implementation requires less storage for code**,** consumes less memory**,** and offers faster operation. Note that the EC-ElGamal scheme shares many properties with other standard EC algorithm. Thus, the major parts from this proposed work are also applicable to other EC implementations on small general purpose processors. The rest of the paper is structured as follows: Wireless Sensor Networks are discussed in Section 2. Then, the related work is highlighted in Section 3. Moreover, elliptic curve ElGamal cryptosystem is illustrated in Section 4. Performance results and evaluation of cryptographic algorithms are presented in Section 5. Finally, Section 6 concludes the paper and then final considerations and future works are given.

2 Wireless Sensor Network

A wireless sensor network (WSN) consists of a large number of tiny sensor nodes deployed over a geographical area also referred as sensing field. Each node is a low-power device that integrates computing, wireless communication and sensing capabilities [8] [9]. Nodes organize themselves in clusters and networks and then they cooperate to perform an assigned monitoring (and/or control) task without any human intervention. Sensor nodes are able to sense physical environmental information such as temperature, humidity, vibration, acceleration and then process locally the acquired data both at sensors and cluster level. The proposed information is then sent to the cluster (or the sink) as in Figure 1.

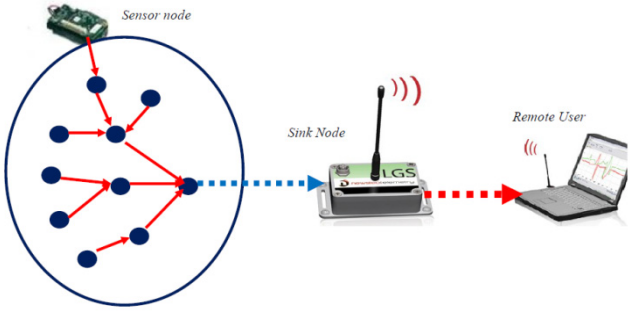

Fig. 1. A typical sensor network architecture

A WSN can thus be viewed as an intelligent distributed measurement technology adequate for many different monitoring and control contexts. In recent years, the number of sensor network deployments for real-life applications has rapidly increased [15]. Examples of WSNs applications in different domains are as follows: environmental monitoring [10], agriculture [11], production and delivery [12], military [10], structure monitoring [13] and medical applications [14]. However, energy consumption still remains one of the main obstacles to the diffusion of this technology, especially in application scenarios where a long network lifetime and a high quality of service are required. In fact, nodes are generally powered by batteries which have limited capacity and often can neither be replaced nor recharged due to environmental constraints. Despite the fact that energy scavenging mechanisms can be adopted to recharge batteries such as through solar panels, piezoelectric or acoustic transducers, energy is a limited resource and must be used judiciously. Interested reader can refer to [16] for more information on scavenging mechanisms. Hence, efficient energy management strategies must be devised at both sensor nodes level and cluster level to prolong the network lifetime as much as possible.

2.1 Security Goals and Challenges

Achieved security goals vary from one security mechanism to another due to the adversarial model considered at the design time. In other words, depending on the

attacks that need to be mitigated, the provided security goals may vary. These security goals are discussed as follows [18]:

- **Data Confidentiality:** Confidentiality means keeping information secret from unauthorized parties. A sensor network should not leak sensor readings to neighboring networks. In many applications (e.g. key distribution) nodes exchange highly sensitive data. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers can possess, hence achieving confidentiality. Since public-key cryptography is too expensive to be used in the resource constrained sensor networks, most of the proposed protocols use symmetric key encryption methods. The authors of TinySec [18] argue that cipher block chaining (CBC) is the most appropriate encryption scheme for sensor networks. They found RC5 and Skipjack to be most appropriate for software implementation on embedded microcontrollers. The default block cipher in TinySec is Skipjack. SPINS uses RC6 as its cipher.

- **Data Authenticity:** In a wireless medium, an adversary can easily inject messages, if no mechanism to prevent unpermitted parties from participating in the network is in place. Thus, the receiver needs to make sure that the data used in any decision making process originates from the correct source. Data authentication prevents unauthorized parties from participating in the network and legitimate nodes should be able to detect messages from unauthorized nodes and reject them. In the two party communication case, data authentication can be achieved through a purely symmetric mechanism. The sender and the receiver share a secret key to compute a message authentication code (MAC) of all exchanged data. When a message with a correct MAC arrives, the receiver knows that it must have been sent by the sender. However, authentication for broadcast messages requires stronger trust assumptions on the network nodes. The authors of SPINS [19] contend that it is insecure to send authenticated data to mutually untrusted receivers, using a symmetric MAC is insecure since any one of the receivers know the MAC key, and hence could impersonate the sender and forge messages to other parties. SPINS constructs authenticated broadcast from symmetric primitives, but introduces asymmetry with delayed key disclosure and one-way function key chains. LEAP [20] uses a globally shared symmetric key for broadcast messages to the whole group. However, since the group key is shared among all the nodes in the network, an efficient rekeying mechanism is defined for updating this key after a compromised node is revoked. This means that LEAP has also defined an efficient mechanism to verify whether a node has been compromised.

- **Data Integrity:** Data integrity ensures the receiver that the received data is not altered in transit either maliciously or accidentally. This property can

help to filter out incorrect/altered data and save the processing energy if the data travelled all the way to the base station.

- **Data Freshness:** Data freshness implies that the data is recent, and no old messages have been replayed. A common defense (used by SNEP [19]) is to include a monotonically increasing counter with every message and reject messages with old counter values. With this policy, every recipient must maintain a table of the last value from every sender it receives. However, for RAM constrained sensor nodes, this defense becomes problematic for even modestly sized networks. Assuming nodes devote only a small fraction of their RAM for this neighbor table, an adversary replaying broadcast messages from many different senders can fill up the table. At this point, the recipient has one of two options: ignore any messages from senders not in its neighbor table, or purge entries from the table. Neither is acceptable; the first creates a DoS attack and the second permits replay attacks. In [21], the authors contend that protection against the replay of data packets should be provided at the application layer and not by a secure routing protocol as only the application can fully and accurately detect the replay of data packets (as opposed to retransmissions ,for example). In [18], the authors reason that by using information about the network's topology and communication patterns, the application and routing layers can properly and efficiently manage a limited amount of memory devoted to replay detection. In [19], the authors have identified two types of freshness: weak freshness, and strong freshness. On one hand, the weak freshness provides partial message ordering, but carries no delay information. This type of freshness is suitable sensor measurements. On the other hand, the strong freshness provides a total order on a request response pair, and allows for delay estimation. This type is useful for time synchronization within the network.

2.2 Types of Attacks on WSNs

Wireless networks are vulnerable to security attacks due to the broadcast nature of the transmission medium. Furthermore, wireless sensor networks have an additional vulnerability because nodes are often placed in a hostile or dangerous environment where they are not physically protected. This section summaries types of attacks may be launched in WSNs. These attacks are as follows:

- **Passive Information Gathering:** An intruder with an appropriately powerful receiver and well-designed antenna can easily pick off the data stream. Interception of the messages containing the physical locations of sensor nodes allows an attacker to locate the nodes and destroy them. Besides the locations of sensor nodes, an adversary can observe the application specific content of messages including message IDs, timestamps and other fields. To minimize the threats of passive information gathering, strong encryption techniques needs to be used.

- **Subversion of a Node:** A particular sensor might be captured, and information stored on it (such as the key) might then be obtained by an adversary. If a node has been compromised then how to exclude that node, and that node only, from the sensor network is at issue (LEAP [22] suggests an efficient way to do so).

- **False Node and malicious data:** An intruder might add a node to the system that feeds false data or prevents the exchange of true data. Such messages also consume the scarce energy resources of the nodes. This type of attack is called "sleep deprivation torture" in [23]. Insertion of malicious code is one of the most dangerous attacks that can occur. Malicious code injected in the network could spread to all nodes, potentially destroying the whole network, or even worse, taking over the network on behalf of an adversary. A seized sensor network can either send false observations about the environment to a legitimate user or send observations about the monitored area to a malicious user. By spoofing, altering, or replaying routing information, adversaries may be able to create routing loops, attract or repel network traffic, extend or shorten source routes, generate false error messages, partition the network, increase end-to-end latency, etc. Strong authentication techniques can prevent an adversary from impersonating as a valid node in the sensor network.

- **The Sybil attack**: In a Sybil attack [24], a single node presents multiple identities to other nodes in the network. They pose a significant threat to geographic routing protocols, where location aware routing requires nodes to exchange coordinate information with their neighbors to efficiently route geographically addressed packets. Authentication and encryption techniques can prevent an outsider from launching a Sybil attack on the sensor network. However, an insider cannot be prevented from participating in the network, but (s)he should only be able to do so using the identities of the nodes (s)he has compromised. Using globally shared key allows an insider to masquerade as any (possibly even nonexistent) node. Public key cryptography can prevent such an insider attack, but it is too expensive to be used in the resource constrained sensor networks. One solution is to have a shared unique symmetric key between each node and a trusted base station. Two nodes can then use a Needham- Schroeder like protocol to verify each other's identity and establish a shared key. A pair of neighboring nodes can use the resulting key to implement an authenticated, encrypted link between them. An example of a protocol which uses such a scheme is LEAP [22], which supports the establishment of four types of keys.

- **Sinkhole attacks:** In a sinkhole attack, the adversary's goal is to lure nearly all the traffic from a particular area through a compromised node, creating a metaphorical sinkhole with the adversary at the center. Sinkhole attacks typically work by making a compromised node look especially attractive to surrounding nodes with respect to the routing algorithm. For instance, an

adversary could spoof or replay an advertisement for an extremely high quality route to a base station. Due to either the real or imagined high quality route through the compromised node, it is likely each neighboring node of the adversary will forward packets destined for a base station through the adversary, and also propagate the attractiveness of the route to its neighbors. Effectively, the adversary creates a large "*sphere of influence*" [25], attracting all traffic destined for a base station from nodes several hops away from the compromised node.

- **Wormholes:** In the wormhole attack [26], an adversary records a packet at one location in the network, tunnels the packet to another location over a low latency link, and replays it at another part of the network. The simplest instance of this attack is a single node situated between two other nodes forwarding messages between the two of them. However, wormhole attacks more commonly involve two distant malicious nodes colluding to understate their distance from each other by relaying packets along an out-of-bound channel available only to the attacker. An adversary situated close to a base station may be able to completely disrupt routing by creating a well-placed wormhole. An adversary could convince nodes that would normally be multiple hops from a base station that they are only one or two hops away via the wormhole. This can create a sinkhole, since the adversary on the other side of the wormhole can artificially provide a high quality route to the base station, potentially all traffic in the surrounding area will be drawn through the adversary if alternate routes are significantly less attractive.

3 Related Works

One of the first requirements for providing a security mechanism is establishing the cryptographic keys to be used by the encryption algorithms. Due to the limited resources and the need for scalability in WSNs, the key establishment protocols used in other fields are not suitable for WSN environments. To address this problem, a lot of work has been done to develop and evaluate specialized key establishment protocols [7], [6], [27]. Publications like [17] mimic asymmetric signatures schemes by a relatively complex scheme of two party hash chains, so do [5] and [6]. Other work like [7] try to establish pairwise secret keys to avoid public and private key schemes or Diffie-Hellman like key exchanges.

In [27] and [28] the authors implement elliptic curve cryptography for sensor networks. However the underlying hardware is quite sophisticated consisting of 16 Bit microcontrollers with 16 MHz clock frequency. Therefore the results are only of limited value as typical sensor hardware does not dispose of such powerful computing resource. As mentioned before sensor networks can in general not afford high clock frequencies and potent CPUs, because of cost and energy saving issues associated.

In [29] a high-performance microcontroller offerings 24 MIPS, i.e. 3 times more than the usual ATMEGA 128, is utilized. The work is based on special Galois fields

called *optimal extension fields* where field multiplication can be done quite efficiently. However, the security of this fields is unclear because of the Weil descent attack [28]. The proposals trying to implement elliptic curves on 8Bit ATMEGA128 chips like in [30] and [31] reach extremely poor performance. For example, a signature generation over 1:08 min of expensive computing and battery time has to be spent, which surely is not affordable. In addition the cost for necessary field operations are not mentioned at all.

4 Elliptic Curve El-Gamal Encryption Scheme

The original ElGamal encryption scheme, see [32], is not additive homomorphic. However, the elliptic curve group is an additive group, which can be used to get an additive homomorphic scheme. Algorithm 1 and Algorithm 2 show the methods for EC-ElGamal encryption and decryption, respectively. Therein, E is an elliptic curve over the finite field $GF(p)$. The order of the curve E is denoted $n = \#E$ and G is the generator point of the curve E. The secret key is defined as integer number x 2 GF(p), while the public key is determined as Y = xG.

The function map () is a deterministic mapping function used to map values mi 2 GF(p) into plaintext curve points Mi 2 E such that

$$Map\ (m1 + m2 + .\ .) = \underbrace{map\ (m_1)}_{M_1} + \underbrace{map\ (m_2)}_{M_2} + \underbrace{\ldots \text{map}\ (\text{m}_\text{n})}_{M_N} \tag{1}$$

holds, whereby $m_1, m_2 \in GF(p)$. Since the addition operation over an elliptic curve requires both operands to be on that curve, prior to performing an addition of two integers, they should be mapped to the corresponding elliptic curve points. This explains why the mapping function is necessary. As proposed in [32] the homomorphic mapping function used in TinyPEDS is based on using multiples of the generator point G of the elliptic curve. This means that the mapping function converts a plaintext m to the point mG. The reverse mapping function *rmap* ( ) then extracts m from a given point mG. The mapping function, namely holds with $m_1, m_2, \ldots m_\text{n} \in GF(p)$, the generator point G, and the modulus p.

$$map: m \longrightarrow mG \text{ with } m \in GF\ (p) \tag{2}$$

fulfills the required homomorphic property due to the fact that the equation

$$\begin{aligned} M_1 + M_2 + \ldots + M_n &= map\ (m_1 + m_2 + \ldots + m_n) \\ &= (m_1 + m_2 + \ldots + m_n)\ G \\ &= m_1 G + m_2 G + \ldots + m_n G \end{aligned} \tag{3}$$

The mapping function is not security relevant, since it only converts an integer to an elliptic curve point. This means, it neither increases nor decreases the security of the EC-ElGamal encryption scheme. Note that the reverse mapping function is the same as

solving the *discrete logarithm problem* over an elliptic curve and, therefore, a weakness of this scheme. However, since the mapping function is only performed on the reader device, which is assumed to have unlimited resources, this disadvantage does not affect the performance and resource consumption within the network.

In conclusion, according to the analysis made in [32], the EC-ElGamal scheme becomes the most promising candidate for using in TinyPEDS, because of its efficiency both in computation and bandwidth. However, the main disadvantage of this scheme is that the reverse mapping function required during decryption may be in some cases too costly. However, since the number of the aggregated values is limited and the maximum length of the final aggregation is assumed to be at most three bytes, see [46], the reverse mapping of the point mG with 24-bit m can be calculated fast enough on the reader device.

Algorithm 1: EC-ElGamal encryption

Require: public key Y, plaintext m
Ensure: ciphertext (R, S)
1: choose random k 2 $[1, n - 1]$
2: $M := map(m)$
3: $R := kG$
4: $S := M + kY$ **(4)**
5: return (R, S)

Algorithm 2: EC-ElGamal decryption

Require: secret key x, ciphertext (R, S)
Ensure: plaintext m
1: $M := -xR + S$
2: $m := rmap(M)$
3: return m

5 Implementation

The implementation was done on the Mica-Z mote, the operating system employed in the implementation is TinyOS-2.0 [33], an open-source operating system designed for wireless embedded sensor networks. In TinyOS there are two kinds of components, namely configurations and modules. Configurations connect modules, while the required functionality, e.g. arithmetic operations, is implemented in modules [33]. Figure 2 depicts a graphical representation of the EC-ElGamal configuration.

5.1 ECElGamalM

The module ECElGamalM implements the EC ElGamal encryption scheme and the arithmetic operations such as homomorphic addition operation.

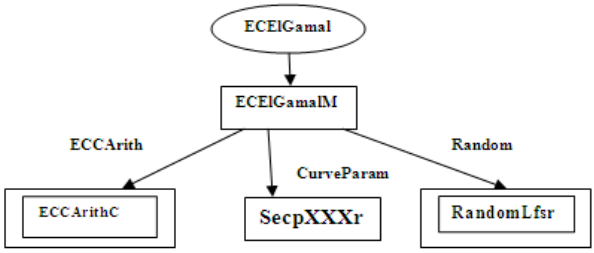

Fig. 2. Graphical representation of elliptic curve ElGamal implementation

Thus, in ECElGamalM following functions are implemented.

- **Void init ():** Initializes the parameters, e.g. G, Y and the pre-computed points, required by the mapping and the encryption function.
- **Void generateRandomNum (FF_DIGIT *k):** This function generates the random *k* required in the EC-ElGamal encryption. Note that the random number generation is based on the method rand16() from the module RandomLfsrC which is contained in TinyOS. Therefore, ECElGamalM calls the external method rand16() and this method call is represented as arrow in figure 2.
- **Void map(Point *M, FF_DIGIT *m, FF_DIGIT lengthOfm**): Software implementation of the mapping function shown in equation 2, *m* is mapped to a elliptic curve point *M*, whereby ***M = mG***.
- **Void enc (ECElGamalCipher *cipher, FF_DIGIT *m, FF_DIGIT lengthOfm):** Software implementation of the EC-ElGamal encryption scheme as described in algorithm 1, whereby *cipher = enc(m).*
- **Void homAdd(ECElGamalCipher *cipher, ECElGamalCipher *cipher1, ECElGamalCipher *cipher2):** Software implementation of the homomorphic addition operation ⊗, see equation 3, with ***cipher* = *cipher*1 ⊗ *cipher*2.**

5.2 ECCArithC

As depicted in Figure 2, the component **ECCArithC** consists of two modules, namely **ECCArithM** and **FFArithM**, which implement the arithmetic operations at elliptic curve and finite field level, respectively.

- **ECCArithM :** The module **ECCArithM** implements the following operations from the elliptic curve level.
- **FFArithM :** The module **FFArithM** implements the following finite field arithmetic operations.

5.3 SecpXXXr1

Elliptic curve parameters such as the base point **G** and the point **Y** and pre-computed points are set in this module.

5.4 RandomLfsrC

This module is already implemented in TinyOS and part of the operating system. The following method is employed from this module. Note that the **RandomLfsrC** does not generate good pseudo-random numbers, which may lead to security problems. However, as they are not within the scope of this paper, the security analysis of weak pseudo-random numbers is not covered in this work.

6 Performance Evaluations

This section presents a comparative performance and energy consumption analysis of this algorithm. We have selected three crucial parameters; memory efficiency, execution time (operation speed), and energy efficiency.

6.1 Memory Efficiency

Memory usually includes flash memory (ROM) and RAM. Flash memory is classified into programming flash memory and data flash memory. Programming flash memory is used to store downloaded application programming code. Data flash memory stores temporary or sensing data. RAM is used for program execution. Because memory in a sensor node is not only limited but also require energy to retain or store data, efficient usage of memory is important.

Besides computing time memory consumption is an important criteria for the use in sensor networks. Figure 3 gives an overview over the memory use of our implementation.

6.2 Operation Time

Operation speed is also an important factor when evaluating performance. After estimating operation time by repeatedly executing encryption and decryption process, we calculate the average of estimated value.

Table 1 shows the performance of the different realizations of the EC-ElGamal, which contains two point multiplications with n-bit scalar k and one short point multiplication with the sensed data m, see Algorithm 1. Note that for testing purposes m was chosen to be 8-bit.

6.3 Energy Efficiency

The energy consumed by a processor during the execution of a piece of software, such as a block cipher, corresponds to the product of the average power dissipation and the

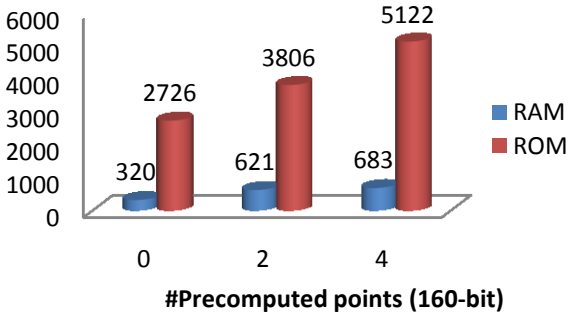

Fig. 3. Memory requirements of each algorithm

Table 1. Operation time requirements of each algorithm

# Recomputed points (160-bit)	Execution time [s]
0	**2.14**
2	**1.22**
4	**0.97**

total running time. The former depends on a number of factors including supply voltage, clock frequency, and the average current drawn by the processor while executing individual instructions of the program code. The computational complexity of an algorithm translates directly to its energy consumption. Assuming the energy per CPU cycle is fixed, by measuring the number of CPU cycle executed per byte of plaintext processed, we get the amount of energy consumed per byte.

We estimate CPU cycle by using Power TOSSIM, which is extension of TOSSIM, an event driven simulation environment for TinyOS applications. Power TOSSIM provides accurate estimation of power consumption for a range of applications and scales to support very large simulation. The energy consumed by a processor during the execution of a piece of software, such as a block cipher, corresponds to the product of the average power dissipation and the total running time.

Table 2 represents the power consumption of the implementations from this work, when those operations are performed.

Table 2. Energy Efficiency requirements of each algorithm

# Recomputed points (160-bit)	Execution time [s]
0	**10.556**
2	**5.560**
4	**5.918**

7 Conclusion

The performance evaluation of cryptographic algorithms is vital for the safe and efficient development of cryptosystem in devices with low computational power.
Due to the resource restrictions of sensor nodes, several algorithms required for implementing the EC-ElGamal cryptosystem are analyzed. Thus, the time efficiency, code size, and memory consumption of each candidate algorithm were compared and the most promising algorithms were selected and implemented.

Moreover, the programming style was selected such that unnecessary overhead in terms of code performance, code size, and memory usage were reduced to minimum.
One future research direction is to explore adaptive cryptographic mechanisms to optimize energy consumption by varying cipher parameters with timely acquisition of resource-context in WSN environment. The adaptability of the security system will improve sensor nodes battery's lifetime.

References

1. Bertoni, G., Breveglieri, L., Venturi, M.: Power aware design of an elliptic curve coprocessor for 8 bit platforms. In: PERCOMW 2006, p. 337 (2006)
2. Wander, A.S., Gura, N., Eberle, H., Gupta, V., Shantz, S.C.: Energy analysis of public-key cryptography for wireless sensor networks. In: PERCOM 2005, pp. 324–328 (2005)
3. Potlapally, N.R., Ravi, S., Raghunathan, A., Jha, N.K.: A study of the energy consumption characteristics of cryptographic algorithms and security protocols. In: IEEE TMC 2005, pp. 128–143 (2005)
4. Chang, C.-C., Muftic, S., Nagel, D.J.: Measurement of energy costs of security in wireless sensor nodes. In: ICCCN 2007, pp. 95–102 (2007)
5. Weimerskirch, A., Westhoff, D.: Zero Common-Knowledge Authentication for Pervasive Networks. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 73–87. Springer, Heidelberg (2004)
6. Weimerskirch, A., Westhoff, D.: Identity Certified Authentication for Ad-hoc Networks. In: 10th Workshop on Security of Ad Hoc and Sensor Networks (2003)
7. Balfanz, D., Smetters, D., Stewart, P., Wong, H.: Talking to strangers: Authentication in adhoc wireless networks. In: Symposium on Network and Distributed Systems Security (2002)
8. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Capirci, E.: Wireless Sensor Networks: a Survey. Computer Networks 38(4) (March 2002)
9. Romer, K., Mattern, F.: The design space of wireless sensor networks. IEEE Wireless Communications 11(6), 54–61 (2004)
10. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: Proc. ACM International Workshop on Wireless Sensor Networks and Applications, pp. 88–97 (2002)
11. Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., Welsh, M.: Monitoring volcanic eruptions with a wireless sensor network. In: Proceeedings of the Wireless Sensor Networks, pp. 108–120 (2005)
12. Lee, K.B., Reichardt, M.E.: Open standards for homeland security sensor networks. IEEE Magazine on Instrumentation & Measurement 8(5), 14–21 (2005)

13. Baldus, H., Klabunde, K., Müsch, G.: Reliable Set-Up of Medical Body-Sensor Networks. In: Karl, H., Wolisz, A., Willig, A. (eds.) EWSN 2004. LNCS, vol. 2920, pp. 353–363. Springer, Heidelberg (2004)
14. Alippi, C., Galperti, C.: An Adaptive System for Optimal Solar Energy Harvesting in Wireless Sensor Network Nodes. IEEE Transactions on Circuits and Systems I: Regular Papers 55(6), 1742–1750 (2008)
15. Slijepcevic, S., Potkonjak, M., Tsiatsis, V., Zimbeck, S., Srivastava, M.B.: On communication security in wireless ad-hoc sensor networks. In: Proceedings of 11th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2002), pp. 139–144 (2002)
16. Carman, D.W., Krus, P.S., Matt, B.J.: Constraints and approaches for distributed sensor network security., Technical Report 00-010, NAI Labs, Network Associates Inc., Glenwood, MD (2009)
17. Anderson, R., Bergadano, F., Crispo, B., Lee, J., Manifavas, C., Needham, R.: A New Family of Authentication Protocols. ACMOSR: ACM Operating Systems Review (1998)
18. Karlof, C., Sastry, N., Wagner, D.: TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In: ACM SenSys 2004, November 3-5 (2004)
19. Xiao, Y. (ed.): Wireless Sensor Network Security: A Survey. Security in Distributed, Grid, and Pervasive Computing. Auerbach Publications, CRC Press (2006)
20. Estrin, D., Govindan, R., Heidemann, J.S., Kumar, S.: Next century challenges: Scalable coordination in sensor networks. In: Mobile Computing and Networking, pp. 263–270 (1999)
21. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 243–254. ACM Press (2000)
22. Zhu, S., Setia, S., Jajodia, S.: LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In: The Proceedings of the 10th ACM Conference on Computer and Communications Security (2003)
23. Stajano, F., Anderson, R.: The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In: 3rd AT&T Software Symposium, Middletown, NJ (October 1999)
24. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: A tiny aggregation service for ad-hoc sensor networks. In: The Fifth Symposium on Operating Systems Design and Implementation, OSDI 2002 (2002)
25. Wagner, C.K.D. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures
26. Hu, Y.C., Perrig, A., Johnson, D.B.: Wormhole detection in wireless ad hoc networks. Department of Computer Science, Rice University, Tech. Rep. TR01-384 (June 2002)
27. Huang, Q., Cukier, J., Kobayashi, H., Liu, B., Zhang, J.: Fast Authenticated Key Establishment Protocols for Self-Organizing Sensor Networks. In: International Conference on Wireless Sensor Networks and Applications (2003)
28. Huang, Q., Kobayashi, H.: Energy/security scalable mobile cryptosystem. IEEE Personal, Indoor and Mobile Radio Communications (2003)
29. Kumar, S., Girimondo, M., Weimerskirch, A., Paar, C., Patel, A., Wander, S.: Embedded End-to-End Wireless Security with ECDH Key Exchange. In: The 46th IEEE Midwest Symposium on Circuits and Systems (2003)
30. Malan, D.J., Welsh, M., Smith, M.D.: A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography. In: First IEEE International Conference on Sensor and Ad Hoc Communications and Networks (2004)

31. Lorincz, K., Malan, D.J., Fulford-Jones, T.R.F., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Moulton, S., Welsh, M.: Sensor Networks for Emergency Response: Challenges and Opportunities. IEEE Pervasive Computing (2004)
32. El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
33. http://www.tinyos.net