Pairing-based Secure Timing Synchronization for Heterogeneous Sensor Networks

Sk. Md. Mizanur Rahman
Department of Computing and Information Science, University of Guelph, Guelph, Ontario, N1G, Canada
srahman@uoguelph.ca

Nidal Nasser
Department of Computing and Information Science, University of Guelph, Guelph, Ontario, N1G, Canada
nasser@cis.uoguelph.ca

Tarik TALEB
Graduate School of Information Sciences, Tohoku University, Sendai, Japan
talebtarik@ieee.org

Abstract— **Secure time synchronization is one of the key concerns for some sophisticated sensor network applications. Most existing time synchronization protocols are affected by almost all attacks. In this paper we consider heterogeneous sensor networks (HSNs) as a model for our proposed novel time synchronization protocol based on pairing and identity based cryptography (IBC). This is the first approach for time synchronization protocol using pairing-based cryptography in heterogeneous sensor networks. The proposed scheme reduces the key spaces of nodes as well as it prevents from all major security attacks. Security analysis indicated that the proposed scheme is robust against reply attacks, masquerade attacks, delay attacks, and message manipulation attacks.**

I. INTRODUCTION

Time synchronization in sensor networks is a challenging task due to some non-determinism in the network dynamics such as physical channel access time and operation system overhead (e.g., system calls). All network time synchronization methods rely on some kind of message exchanges between nodes. Thus, secure time synchronization is more challenging. Additionally, resource constraints and lack of efficient key management in sensor networks make it more difficult. In this paper we consider secure time synchronization for heterogeneous sensor networks (HSNs), as several researches show HSNs have greater performance over homogeneous WSNs. In a recently proposed time synchronization scheme for HSN [1], High-end nodes (H-nodes) and Low-end nodes (L-nodes) need to pre-load a number of keys. Some existing time synchronization schemes for HSNs, such as the Reference-Broadcast Synchronization (RBS) scheme [2], the Timing-sync Protocol for Sensor Networks (TPSN) [3], and the Flooding Time Synchronization Protocol (FTSP) [4] are also available. But none of these protocols [2-4] was designed with security in mind.

A. Background

Most existing time synchronization schemes [2-4] are vulnerable to the following attacks [1].

1) Masquerade attack: This is a type of attack in which one system entity illegitimately poses as another entity to gain access to confidential systems; that is one system assumes the identity of another. Suppose that a node A sends out a reference beacon to its two neighbors B and C. An attacker E can pretend to be B and exchange wrong time information with C, disrupting the time synchronization process between B and C.

2) Replay attack: A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. Suppose Alice wants to prove her identity to Bob. Bob requests her password as proof of identity, which Alice dutifully provides (possibly after some transformation like a hash function); meanwhile, Eve is eavesdropping the conversation and keeps the password. After the interchange is over, Eve connects to Bob posing as Alice; when asked for a proof of identity, Eve sends Alice's password read from the last session, which Bob must accept.

3) Message manipulation attack: In this attack, an attacker may drop, modify, or even forge the exchanged timing messages to interrupt the time synchronization process [1].

4) Delay attack: The attacker intentionally delays some of the time messages, e.g., the beacon message in the RBS scheme, so as to fail the time synchronization process.

In the branch of Cryptography, Identity-Based Cryptography (IBC) [5] is an exception where an information that uniquely identifies users (e.g., IP or email addresses) can be used to both exchange keys and encrypt data, and thus Public Key Infrastructure (PKI) is unnecessary. It only has become truly practical with the advent on Pairing-Based Cryptography (PBC) [6]. The first known implementation of pairings for sensor nodes based on the 8-bit/7.3828-MHz ATmega128L microcontroller (e.g., MICA2 and MICAz motes) has been investigated in [7]. The investigation concludes that cryptography from pairings is indeed viable in resource-constrained nodes.

In this paper, we propose an efficient secure timing synchronization scheme in the form of pairing-based cryptography. Hence this scheme reduces key space and communication overhead comparing to the existing protocols.

B. Contributions and Related Work

In [1], before deployment or at deployment, all the L-sensors and H-sensors need to store secret as well as public keys of the nodes. All the L-nodes under an H node are indexed with their corresponding secret keys. This incurs additional communication overhead. So, if a L-node moves from one group to another, re-synchronization becomes required for the existing scheme [1], which incurs extra communication overhead for both the leaving group and the new group. On the other hand, L-nodes are hierarchically grouped under H-nodes. Therefore, if there are N L-nodes under the same H-node and grouped into K groups, the H-node is required to broadcast the same timing message N/K times in the same group. Obviously, this shall incur significant communication overhead as the network scales up.

In our proposed scheme, nodes do not need to store any private or public key of other nodes, during the deployment or in its life time. In fact, nodes generate their own secret key by sharing their identity with their neighboring nodes. Thus key space is less than any other existing scheme. Every nodes need to know its own ID and its corresponding secret that are taken from the trusted base (sink) station. In our scheme, nodes do not need to be static; they can be dynamic as long as adequate clustering is performed in the network. Nodes in the network can dynamically generate their own secret sharing keys. In addition, nodes do not need to be indexed and ordered. Finally, any node can move from one group to another without any extra overhead communication.

C. *Organization of the Paper*

The rest of this paper is organized as follows. Section II describes preliminaries which are useful for understanding the proposed protocol. In Section III, the architecture of the proposed protocol is described. In Section IV, the key management and authentication procedure is described. In Sections V and VI, the secure time synchronization procedure and some security analysis - in terms of prevention of attacks - are given, respectively. Finally, conclusions drawn from the paper are presented in Section VII.

II. Preliminaries

In this section, we describe some preliminaries and mathematical properties which are useful to understand our proposed protocol.

A. *Bilinear Maps*

Let G_1 be an additive group and G_2 be a multiplicative group of the same prime order q. Let P be an arbitrary generator of G_1. aP denotes P added to itself a times. Assume that discrete logarithm (DL) problem is hard in both G_1 and G_2. We can think G_1 as a group of points on an elliptic curve over F_q, and G_2 as a subgroup of the multiplicative group of a finite field F_{q^k} for some $k \in Z_q^*$, where $Z_q^* = \{y \mid 1 \leq y \leq q-1\}$. A mapping $e: G_1 \times G_1 \rightarrow G_2$, satisfying the following properties is called a cryptographic bilinear map.

- *Bilinearity:* $e(aP, bQ) = e(P,Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$. This can be restated in the following way. For $P, Q, R \in G_1$, $e(P+Q, R) = e(P,R)\, e(Q,R)$ and $e(P, Q+R) = e(P,Q)\, e(P,R)$.
- *Non-degeneracy:* If P is a generator of G_1, then $e(P,P)$ is a generator of G_2. In other words, e(P,P) not equal to one.
- *Computable:* A mapping is efficiently computable if $e(P,P)$ can be computed in polynomial-time for all $P, Q \in G_1$.

Modified Weil Pairing [8] and Tate Pairing [9] are examples of cryptographic bilinear maps.

B. *Diffie-Hellman Problems*

With the group G_1 described in Subsection II.A, we can define the following hard cryptographic problems applicable to our proposed scheme.

- *Discrete Logarithm (DL) Problem:* Given $P, Q \in G_1$, find an integer n such that $P=nQ$ whenever such integer exists.
- *Computational Diffie-Hellman (CDH) Problem:* Given a triple $(P, aP, bP) \in G_1$ for $a, b \in Z_q^*$, find the element abP.
- *Decision Diffie-Hellman (DDH) problem:* Given a quadruple $(P, aP, bP, cP) \in G_1$ for $a, b, c \in Z_q^*$, decide whether $c=ab \bmod q$ or not.
- *Gap Diffie-Hellman (GDH) Problem:* A class of problems where the CDH problem is hard but DDH problem is easy.
- *Bilinear Diffie-Hellman (BDH) Problem:* Given a quadruple $(P, aP, bP, cP) \in G_1$ for some $a, b, c \in Z_q^*$, compute $e(P,P)^{abc}$.

Groups where the CDH problem is hard but DDH problem is easy are called GAP Diffie-Hellman (GDH) groups. Details about GDH groups can be found in [10].

C. *Routing Structure in HSNs*

A HSN consisting of two types of sensors: a small number of H-sensors and a large number of L-sensors. Both H-sensors and L-sensors are powered by batteries and have limited energy supply. Clusters are formed in a HSN. For a HSN, it is natural to have powerful H-sensors serve as cluster heads and form clusters around them. The assumptions and discussion are given in [11]. In brief, *i)* each L-sensor (and H-sensor) is static and aware of its own location (sensor nodes can use a secure location service such as that in [12] to estimate their locations; no GPS receiver is required at end-nodes), *ii)* each L-sensor (and H-sensor) has a unique node ID. 3) The sink (base) is trusted.

1). Cluster formation in HSNs: After sensor deployment, clusters are formed in a HSN [11]. An efficient clustering scheme for HSNs is given in [12]. For the sake of simplicity, we assume that each H-sensor can communicate directly with its neighbor H-sensors (if not, then relay via L-sensors can be used). All H-sensors form a backbone in a HSN. After cluster formation, a HSN is divided into multiple clusters, where H-sensors serve as the cluster heads. For reader's convenience, the HSN clustering scheme is described briefly as follows, more details are in [12]. During the network initialization, each H-sensor broadcasts a *Hello* message to nearby L-sensors using the maximum power and with a random delay. The random delay is to avoid the collision of Hello messages from two neighbor H-sensors. A *Hello* message includes the ID and location of the H-sensor. A L-sensor may receive *Hello* messages from one or more H-sensors. Each L-sensor sets a timer after receiving the first *Hello* message. When the timer expires, each L-sensor chooses the H-sensor whose *Hello* message has the best signal strength as the cluster head. Each L-sensor also records other H-sensors from which it receives the *Hello* messages, and these H-sensors serve as backup cluster heads in case the primary cluster head fails. If a L-sensor *U* does not receive any *Hello* message during the initialization phase, it actively looks for a nearby H-sensor by broadcasting an *Explore* message. Neighbor L-sensors relay the *Explore* message until it reaches a L-sensor *V* that has already found a cluster head. Then sensor *V* sends a message back to sensor *U* and informs sensor *U* of the location of the cluster head. This scheme ensures that all connected L-sensors have a cluster head. A sensor network is divided into multiple clusters, where each H-sensor serves as the cluster head.

2). Routing in HSN: In a HSN, the sink, H-sensors and L-sensors form hierarchical network architecture [11]. Clusters are formed in the network and H-sensors serve as cluster heads. All H-sensors form a communication backbone in the network. Powerful H-sensors have sufficient energy supply, long transmission range, high date rate, and thus provide many advantages for designing more efficient routing protocols. In [13], an efficient routing protocol for HSNs is designed. Routing in a HSN consists of two phases: 1) Intra-cluster routing: each L-sensor sends data to its cluster head via multi-hops of other L-sensors; and 2) Inter-cluster routing: a cluster head (a H-sensor) aggregates data from multiple L-sensors and then sends the data to the sink via the H-sensor backbone. More details on routing are given in [13]. An intra-cluster routing scheme determines how to route packets from a L-sensor to its cluster head. The basic idea is to let all L-sensors (in a cluster) form a tree rooted at the cluster head H. It has been shown in [14] that: *i*) If complete data fusion is conducted at intermediate nodes, (i.e., two k-bit packets come in, and one k-bit packet goes out after data fusion) then a minimum spanning tree (MST) consumes the least total energy in the cluster. *ii*) If there is no data fusion within the cluster, then a shortest-path tree (SPT) consumes the least total energy. *iii*) For partial fusion, finding the tree that consumes the least total energy is a NP-complete problem.

During the tree setup, two or more parent nodes are determined for each L-sensor. One parent node serves as the primary parent, and other nodes serve as backup parents. Given the tree-based routing structure within a cluster, each L-sensor only needs shared keys with its communication neighbors, i.e., its parent-nodes and child-nodes.

III. Proposed Protocol's Architecture and Design

The base station (sink or system administrator) has the following extra tasks during the boot strap of the network.

- Determining two groups G_1 and G_2, of the same prime order q. We view G_1 as an additive group and G_2 as a multiplicative group as discussed in Subsection II.A.
- Determining bilinear map g: $G_1 \times G_1 \rightarrow G_2$, collision resistant cryptographic hash functions H_1 and H_2, where $H_1:\{0,1\}^* \rightarrow G_1$ is a mapping function from arbitrary-length strings to points in G_1 and $H_2:\{0,1\}^* \rightarrow \{0,1\}^\mu$ forms another mapping function from arbitrary-length strings to μ-bit fixed length output.
- Generating system's secret $\acute{\omega} \in Z_q^*$, where $Z_q^* = \{y \mid 1 \leq y \leq q-1\}$. Any one in the network does not know $\acute{\omega}$ except the base station (sink). Base station also uses this secret to generate the secret point of the non-adversary nodes.

Thus, the system parameters $<G_1, G_2, g, H_1, H_2>$ are known to the non-adversary nodes. The base station also provides the following parameters for nodes, regarding their IDs and secret points.

- Providing each nodes (L-nodes and H-nodes) with a unique ID:

$$ID_{R1} = H_1(ID_{R1}{}^{\Re}), ID_{R2} = H_1(ID_{R2}{}^{\Re}), \ldots,$$
$$ID_{RN} = H_1(ID_{RN}{}^{\Re}) \in G_1$$

  for N number of nodes and corresponding secret points $\{ SP_{R1}, SP_{R2}, \ldots, SP_{RN} \in G_1 \}$ which are defined as ($SP_{Ri} = \acute{\omega}\, ID_{Ri} = \acute{\omega}\, H_1(ID_{Ri}{}^{\Re})$), where i=1,2,...,N. The base station also provides each node with a different random number $R_{Ni} \in Z_q^*$. If a L-node is not directly connected to a H-node, it then uses its secret point and its communication neighbor's ID to generate the sharing secret key between the node and its communication neighbors. The secret point is also used to authenticate among nodes. If a L-node is directly connected to a H-node, it uses its secret point to generate the sharing secret key between the node and its corresponding H-node. The key is also used for authentication between the two nodes. The key generation and authentication technique is described in the next section. For a given set of $<ID_R, SP_R>$, no one can determine the system secret $\acute{\omega}$ as we discussed in Subsections II.A and II.B.
- Providing each H-node with the IDs of all corresponding L-nodes and a corresponding random number $R_N \in Z_q^*$. This random number is used to authenticate the L-node with its corresponding H-node. As will be explained in the next section, the random number of each L-node is periodically updated and notified by the base station to its corresponding H-node.

With the above information, any node can generate its own sharing secret key. Considering a node K, the node receives its

ID, ID_K, and its corresponding secret point, ($SP_K = \acute{\omega} ID_K = \acute{\omega} H_1(ID_K{}^{\mathfrak{R}})$), from the base station. Hence, K can generate its own sharing secret key with its communication neighbor, node M with an ID, ID_M and a corresponding secret point ($SP_M = \acute{\omega} ID_M = H_1(ID_M{}^{\mathfrak{R}})$). As a result, both K and M can generate their own secret sharing key without sharing their secret point as follows. K computes $K_{KM} = g(SP_K, ID_M) = g(ID_K, ID_M)^{\acute{\omega}}$ and M computes $K_{MK} = g(SP_M, ID_K) = g(ID_M, ID_K)^{\acute{\omega}}$. These equations also hold the property mentioned in Section II: no one can determine the system secret $\acute{\omega}$ for a given set of ID and corresponding secret point $<ID_K, SP_K>$.

IV. Key Management and Authentication

The base station works as a system administrator. It is responsible of generating IDs and corresponding secret points for all nodes in the network. At the boot strap of the network, each node knows its ID, its secret point, and its random number. Furthermore, each H-node knows IDs of all corresponding L-nodes as well as their random numbers. The random number of the nodes is periodically updated by the base station and is distributed to L-nodes via their corresponding H-nodes. The random number is used for authentication between a L-node and its L-node neighbors or between a L-node and its corresponding H-node. The remainder of this section describes the key establishment and authentication processes.

After boot strap and cluster formation of the network, all L-nodes know their corresponding H-nodes. Let's consider a H-node with an ID, ID_{RH1}, a corresponding secret point, SP_{RH1}, and a random number, R_{H1}. Similarly, we consider a L-node with an ID, ID_{RL1}, a corresponding secret point, SP_{RL1}, and a random number R_{L1}. To authenticate to the L-node, ID_{RL1}, H-node generates its sharing secret key $\{K_{H1L1} = g(SP_{RH1}, ID_{RL1}) = g(ID_{RH1}, ID_{RL1})^{\acute{\omega}}\}$ and an authentication code $\{Aut_0 = H_2(K_{H1L1} \| ID_{RH1} \| ID_{RL1} \| 0)\}$. Only Aut_0 is sent to ID_{RL1}. On the other hand, the L-node generates its sharing secret key $\{K_{L1H1} = g(SP_{RL1}, ID_{RH1}) = g(ID_{RL1}, ID_{RH1})^{\acute{\omega}}\}$ and a verification code $\{Ver_0 = H_2(K_{L1H1} \| ID_{RL1} \| ID_{RH1} \| 0)\}$. The node then compares Ver_0 with Aut_0. If Ver_0 matches with Aut_0, the node generates an another authentication code $\{Aut_1 = H_2(K_{L1H1} \| ID_{RL1} \| ID_{RH1} \| R_{L1} \| 1)\}$ and sends it back to H-node. Finally, H-node computes $\{Ver_1 = H_2(K_{H1L1} \| ID_{RH1} \| ID_{RL1} \| R_{L1} \| 1)\}$ and compares it with Aut_1. If a match is found, the authentication is successful. Otherwise, it fails. In this fashion, all nodes (H-nodes and L-nodes) are authenticated to each other; either with their communication neighbors or with their corresponding H-nodes. Therefore, nodes in the network can communicate securely with each other, as illustrated in Fig. 1.

When the base station changes the random numbers of the L-nodes, it encrypts the random number with the secret key shared between the base station and the corresponding H-nodes. As L-nodes are authenticated with their corresponding H-nodes, H-nodes encrypt the random numbers with the secret key shared between the H-nodes and the corresponding L-nodes and send them to the L-nodes. If a L-node is not directly connected with its corresponding H-node, the H-node performs double encryptions. In the first encryption, it uses the secret key shared with the destination L-node. In the second encryption, it uses the secret key shared with its nearby L-node and sends the encrypted packet to its nearby L-node. Consequently, the L-node decrypts the packet and looks at the destination ID and again encrypts the packet using the secret key shared with its L-node neighbor. This operation is repeated till the random number eventually reaches at the destined L-node. It should be noted that intermediate L-nodes cannot get any information about the random number as it is double encrypted by the H-node.

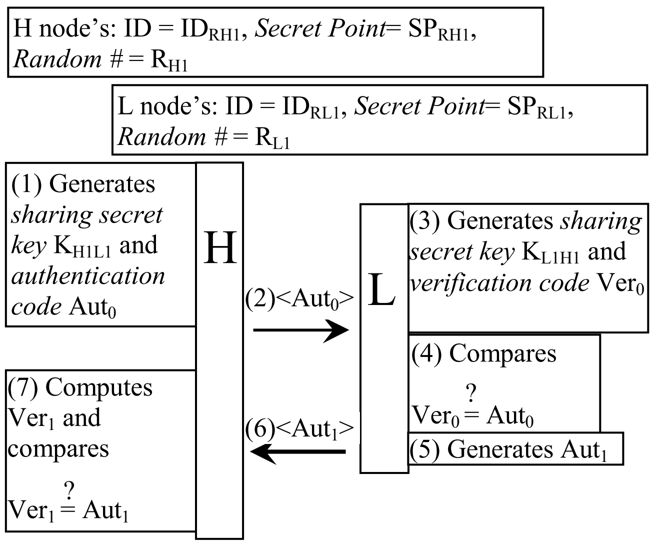
H node's: ID = ID_{RH1}, *Secret Point*= SP_{RH1}, *Random #* = R_{H1}
L node's: ID = ID_{RL1}, *Secret Point*= SP_{RL1}, *Random #* = R_{L1}
H
L
(1) Generates *sharing secret key* K_{H1L1} and *authentication code* Aut_0
(2)<Aut_0>
(3) Generates *sharing secret key* K_{L1H1} and *verification code* Ver_0
(4) Compares $Ver_0 \overset{?}{=} Aut_0$
(5) Generates Aut_1
(6)<Aut_1>
(7) Computes Ver_1 and compares $Ver_1 \overset{?}{=} Aut_1$

Fig. 1. Authentication procedure between two nodes "H" and "L".

V. Secure Time Synchronization Scheme

After sensor deployment and secure key establishment, nodes are authenticated to each other as discussed in the previous section. Time synchronization among the nodes is performed in a hierarchical fashion. The base station first sends the *timing message* to the H-nodes encrypted with the corresponding secret key of H-nodes. The *timing message* contains the following fields: *Position+ Timestamp+ Sequence# +MAC*, where *position* indicates the position of the base station; *timestamp* indicate the local time read from the base station's clock; *sequence#* number is the sequence number of each synchronization message; and *MAC* is calculated with the secret private key over *Position+ Timestamp+ Sequence#*. H-nodes can check the authenticity of the message by referring to the *MAC* values.

Upon receiving a timing message from the base station, H-nodes replace the *position* information of the base station by their own *position* information and then calculate the *MAC* in the same way as the base station. Subsequently, they encrypt the *MAC* with the secret key shared with the corresponding L-nodes and forward the message to the L-nodes.

Upon receiving the timing message from higher-layer H-nodes, L-nodes use the signal strength to compute the distance between the sender and themselves. This distance is denoted as $Distance_{Signal}$, where $Distance_{Signal} = F(signal)$; F is a function that relates between the signal strength and the distance. L-nodes also calculate another distance based on their corresponding relative *position*. This distance is denoted as $\text{Distance}_{\text{Position}}$, where $\text{Distance}_{\text{Position}} = f(\textit{position})$; f is the function that calculates the geometrical distance based on given coordinates. Finally, $\text{Distance}_{\text{Signal}}$ and $\text{Distance}_{\text{Position}}$ are compared to check either they are equal or not. The authenticity is confirmed if the two distances are equal. Another authenticity check can be based on checking the MAC value.

VI. Security Analysis

In this section, we discuss how the proposed approach can cope with different attack types.

1). Masquerade attack: In our proposed scheme, all nodes along the communication path are authenticated. An intruder cannot disguise as a legitimate node and cannot exchange any false information between legitimate nodes.

2.) Reply attack: In our proposed scheme, an attacker node cannot pass the authentication process. Even if a malicious node succeeds in that once, it will more likely not be able to make it for another round: the base station periodically updates the random number used in the computation of the verification/Authentication code.

3). Message manipulation attack: To perform this attack, an attacker needs to take part in the message communication. This is not possible unless the attacker is a valid node in the network. In our scheme, an attacker cannot forge the path or the data packets. Thus, this attack cannot take place if the proposed protocol is in use.

4). Delay attack: In our proposed protocol, nodes can calculate the distance to the senders. They therefore can estimate the traveling time of a packet; a feature that renders delay attacks not effective if the proposed protocol is in use.

VII. Conclusion

In the recent literature, there are many existing time synchronization protocols for homogeneous sensors networks and only a few for heterogeneous sensor networks. Most (if not all) existing solutions exhibit drawbacks in terms of key space, security, and communication overhead. In this paper, we proposed a new secure time synchronization protocol for heterogonous sensor networks based on IBE and pairing based cryptography. The proposed protocol does not require any grouping or ordering of L-nodes under H-nodes and the otherwise-required communication overhead is consequently reduced. Additionally, the proposed protocol prevents different possible attacks, such as masquerade attacks, reply attacks, message manipulation attacks and delay attacks. Mathematical security analysis and performance evaluation via computer simulations form the basis of our future work in this progressive research work.

References

[1] X. Du, M. Guizani, Y. Xiao, and H.H. Chen, "Secure and Efficient Time Synchronization in Heterogeneous Sensor Networks," IEEE Transactions on Vehicular Technology, Vol. 57, No. 4, Jul. 2008

[2] M. L. Sichitiu and C. Veerarittiphan, "Simple, Accurate Time Synchronization for Wireless Sensor Networks," in Proc. of WCNC 2003, New Orleans, Louisiana, USA, Mar. 2003.

[3] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync Protocol for Sensor Networks," in Proc. of the 1st International Conference on Embedded Networked Sensor Systems (ACM SenSus'03), Losangeles, CA, USA, Nov. 2003.

[4] M. Maroti, B. Kusy, G. Simon, A. Ledeczi, "The Flooding Time Synchronization Protocol," in Proc. of the Second ACM SenSys, Baltimore, MD, USA, Nov. 2004.

[5] A. Shamir, "Identity-based Cryptosystems and Signature Schemes," in proc. of CRYPTO'84: on Advances in Cryptology, Springer-Verlag, Santa Barbara, California, USA, Aug. 1984.

[6] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems Based on Pairing," in proc. of Symposium on Cryptography and Information Security (SCIS2000), Okinawa, Japan, Jan. 2000.

[7] L. B. Oliveira, D. F. Aranha, E. Morais, F. Daguano, J. Lopez, and D. Ricardo, "TinyTate: Computing the Tate Pairing in Resource-Constrained Sensor Nodes," in Proc. of Sixth IEEE International Symposium on Network Computing and Applications, (NCA 2007), Cambridge, MA, USA, Jul. 2007

[8] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," SIAM Journal of Computing, Vol. 32, No. 3, Mar. 2003, pp. 586-615.

[9] P. S. L. M. Berreto, H. Y. Kim and M. Scott, "Efficient Algorithms for Pairing-based Cryptosystems," Advances in Cryptology - Crypto '2002, LNCS 2442, Springer-Verlag, 2002, pp.354-368.

[10] D. Boneh and M. Franklin, "Identity-based Encryption from the Weil Pairing," Advances in Cryptology – CRYPTO, Lecture Notes in Computer Sci. 2139, 2001, pp.213-229.

[11] X. Du, M. Guizani, Y. Xiao, and S. Ci, H.H. Chen, "A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks," IEEE Transactions on Wireless Communications, Accepted for publication, Apr. 2007

[12] T. Taleb, F. Nait-Abdesselam, A. Jamalipour, K. Hashimoto, N. Kato, and Y. Nemoto, "Design Guidelines for a Global and Self-Managed LEO Satellites-Based Sensor Network," in Proc. IEEE Globecom'06, San Francisco, USA, Nov. 2006.

[13] X. Du and Y. Xiao, "Energy Efficient Chessboard Clustering and Routing in Heterogeneous Sensor Network," International Journal of Wireless and Mobile Computing (IJWMC), Vol. 1, No. 2, Jan. 2006, pp. 121 -130.

[14] R. Cristescu and B. Beferull-Lozano, "Lossy Network Correlated Data Gathering with High-Resolution Coding," in Proc. of IEEE IPSN 2005, UCLA, Los Angeles, California, USA, Apr. 2005.