MHIP: Effective Key Management for Mobile Heterogeneous Sensor Networks

Dulal Kar[1], Rashad Tatum[2], and Keith Zejdlik[1]
(Corresponding author: Dulal Kar)
School of Engineering and Computing Sciences, Texas A&M University-Corpus Christi[1]
6300 Ocean Dr, Corpus Christi, TX 78412-5825, USA
Department of Mathematics, Southern Polytechnic State University[2]
1100 South Marietta Pkwy, Marietta, GA 30060, USA
(Email: dulal.kar@tamucc.edu, {tatum.rashad, kealze}@gmail.com)

(Received Dec. 27, 2011; revised and accepted Feb. 8, 2012)

Abstract

Security for mobile wireless sensor networks has many unique challenges. Existing security protocols have serious drawbacks revoking compromised nodes from the network, replenishing the network with new nodes in secure manner, and performing in network processing if nodes migrate from one cluster to another. In this work, we propose a new energy-efficient protocol that provides security for a heterogeneous mobile WSN. The protocol is based on Identity Based Encryption (IBE) which facilitates communication-efficient key establishment among the nodes in a WSN as well as revocation of compromised nodes from or addition of new nodes to the network. It also has a mechanism to recover the nodes of a compromised cluster and bind them to other neighboring clusters. The proposed protocol is scalable as the protocol's storage, communication, and computation overheads remain constant regardless of the size of the network. Our analysis of the protocol shows that it can provide security against various threats and attacks including the Sybil attack.

Keywords: Elliptic curve cryptography, identity based encryption, pairing based cryptography, security protocol, wireless sensor network

1 Introduction

A wireless sensor is a simple data sensing, computing, and communicating device which is designed to be powered by battery. As such, it has very limited memory capacity and processing and communicating capabilities. Because of their simple architecture, wireless sensor nodes are inexpensive and can be deployed in large numbers cost-effectively in many situations. A wireless sensor network (WSN) is a collection of such sensor nodes that communicate wirelessly to collect environmental data as well as monitor and control activities within the environment. Specific applications of wireless sensor networks include wildlife monitoring, seismic activity monitoring, volcanic activity monitoring, target tracking, battlefield reconnaissance and surveillance, and emergency rescue operations [1].

As for operation of a typical wireless sensor network is concerned, all sensor nodes communicate with their neighbors, a base station as well as intermediate nodes such as cluster heads. A base station is a relatively powerful computing and communicating node which often acts as a gateway or a storehouse of collected data. Figure 1 shows a typical configuration of a wireless sensor network.

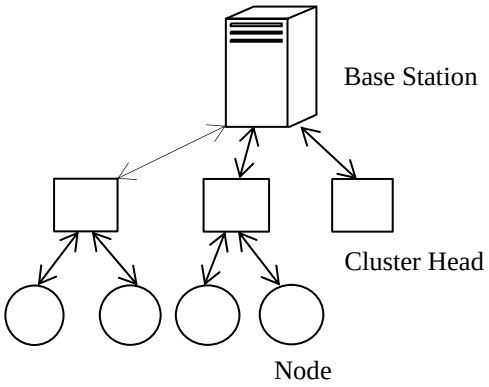

Figure 1: Typical wireless sensor network

However, it is possible to have a complex communicating configuration of a network with multiple base stations and multiple levels of communications among the sensor nodes.

Security of a wireless sensor network is crucial as it is typically deployed in an area where there is no physical security thus making it very vulnerable for easy attacks [3, 4, 13, 17, 26]. It is very challenging to secure a wireless sensor network mainly due to its resource-constrained sensor nodes which cannot run the conventional cryptographic algorithms or protocols that are being used to guarantee security of traditional network communications. Data aggregation (ability to aggregate reported values from other nodes) and passive participation (ability to not send overhead values) are also the crucial issues for sensor network security. Often implementing security on resource-starved sensor devices imposes extra computational and communication overhead that can be

viewed excessive in some applications. This is due to the fact that a security application has to compete for resources with the main application. As such, a lightweight yet effective security solution is sought for wireless sensor networks.

Recent research on security of wireless sensor networks has produced many promising results. For example, two symmetric key algorithms, Skipjack and RC5 are found to be very suitable for resource constrained wireless sensor networks [11]. Similarly, elliptic curve based public key cryptosystems (e.g., identity based encryption) are found to be very promising for wireless sensor networks. A good number of security schemes of significant performance using Skipjack, RC5, Elliptic Curve Cryptography (ECC), and Identity Based Encryption (IBE) for sensor network applications have been proposed in literature [9, 16, 20, 23]. However, these protocols are found to be deficient or not readily useful for WSNs where nodes can migrate from one part of the network to another, nodes need to be revoked from the network, or new nodes to be added to the network. This is very true for any mobile wireless sensor network. Indeed there are many situations where mobility of sensor nodes cannot be avoided, particularly for applications in any marine or aquatic environment where water current or waves can displace sensor nodes within a network.

One of the key challenges in mobile WSNs is how to efficiently and securely gather data from sensor nodes while minimizing energy consumption. In this regard, cluster based hierarchical WSNs have been found to be very energy-efficient [18]. Cluster based WSNs reduce energy consumption on nodes by localizing data transmission within a cluster and incorporating data aggregation in each cluster head [8]. Since the cluster head requires more computation and communication compared to a regular sensor node, it is best for the cluster based networks to use a heterogeneous design by incorporating and integrating more powerful cluster heads with less powerful sensor nodes in the network. While cluster based networks help to effectively gather data, but for many applications data must also be transmitted securely. In this work, we propose a new security protocol for semi-mobile wireless sensor networks of hierarchical network configuration.

We find that Identity-Based Encryption (IBE) based on Elliptic Curve Cryptography (ECC) is very suitable for such security applications in wireless sensor networks [9, 14, 20]. Some of the key challenges of key management may be easily addressed using IBE. Cluster keys for nodes must be established securely to assure that cluster keys have not been compromised. We can establish cluster pairwise keys using IBE to securely distribute these keys. IBE based on ECC can also be used to effectively revoke compromised nodes by updating the secret keys of non-compromised nodes. IBE uses a hashing and mapping function with the identities of nodes to establish pairwise keys. Thus, nodes do not need to communicate to establish pairwise keys; they only need to know identities of nodes with which they wish to communicate. Since the identities of nodes are used to establish pairwise keys, IBE can be used to provide a secure protocol that is effective against a variety of attacks such as the Sybil attack, node compromise, or cluster head compromise. Accordingly, in this work, we propose a new security protocol for key distribution and management in wireless sensor networks.

The rest of the paper is organized as follows. In Section 2, we review cluster based protocols and protocols that use IBE. In Section 3, we state the assumptions for our protocol, MHIP (Mobile Heterogeneous Identity based Protocol). In Section 4, we provide a detailed description of our protocol. In Section 5, we discuss how our protocol handles some common security issues. In Section 6, we evaluate the performance of MHIP before concluding in Section 7.

2 Related Works

Many security protocols already exist for wireless sensor networks. In the following, we discuss some existing significant protocols for wireless sensor networks.

SPINS (Security Protocols for Sensor Networks) is one of the first and well-known security protocols developed for wireless sensor networks using symmetric key cryptography [17]. Perrig et al. proposed two security blocks in SPINs which are Secure Network Encryption Protocol (SNEP) and "micro" Timed Efficient Stream Loss-tolerant Authentication (μTESLA). While SNEP provides data confidentiality, two-party data authentication, and data freshness, μTESLA is developed to provide authenticated broadcast for resource-constrained environments. However, SPINs only deals with three kinds of communication patterns: 1) Node to base station, 2) Base station to node, and 3) Base station to all nodes. In SPINS, each sensor node shares a pre-distributed master secret key with the base station. All other keys are bootstrapped from the initial master secret key. One of the drawbacks of SPINS is that it does not consider different security requirements for different types of messages, which may reduce lifetime of sensor networks unnecessarily. For example, routing control information may not require confidentiality whereas sensor readings and aggregated reports should be encrypted before they are sent to the base station. Depending on the requirements, different security mechanisms should be used for different types of messages in wireless sensor networks since one single key mechanism is not enough to satisfy different security requirements [26]. Also, SPINS does not support mobility of sensor nodes within the network.

Another promising protocol is the Localized Encryption and Authentication Protocol (LEAP) which is based on symmetric key cryptography [26]. LEAP establishes four keys for different types of communication. It has built-in node revocation and also has cluster keys. The disadvantage is that LEAP assumes a static network topology. It does allow for additional nodes to be added to an already existing network but assumes that once nodes enter the network their positions never change. Depending on the density of the network, LEAP can also incur relatively high storage costs.

TinySec is a security architecture which operates on the data link layer. Unlike SPINS and LEAP, TinySec is not limited to any keying mechanism [11]. TinySec uses a pair of Skipjack keys to encrypt data and compute MACs of packets. TinySec uses three different keying mechanisms to secure

sensor network applications: 1) Single network-wide key shared by all the authorized sensor nodes in the network, 2) Group key shared by a group of neighboring nodes, and 3) Per-link key shared by each pair of sensor nodes. Each mechanism has its own advantages and disadvantages. For example, if any authorized node is compromised and the network-wide key is revealed, an adversary can eavesdrop or inject messages in the network. Being a link layer protocol, TinySec does not have any mechanism to recover from such compromises.

Many common deficiencies in symmetric key cryptographic protocols for wireless sensor networks can be overcome by using public key cryptography. However, some good studies on applying public-key cryptographic schemes on wireless sensor networks such as RSA and Diffie-Hellman are found to be computationally intensive [24]. Relatively, ECC is less computationally intensive compared to RSA for the same level of security [15]. However, experiments show that ECC based schemes with only software implementation incur a delay up-to tens of seconds [22].

Fortunately, recent development on ECC has made applications of identity based encryption possible for wireless sensor networks particularly due to development of pairing based cryptography using Weil and Tate pairing functions [2, 10, 19, 25]. In particular, TinyPBC based on IBE offers a way to establish keys between nodes with no communication overhead. This is a very useful feature since communication is one of the biggest drains on battery life for sensor nodes. It also allows for a mobile WSN since the keys do not have to be established during the setup phase. Another benefit of this protocol is that authentication is handled automatically. The ability of a node to decrypt a message implies that the sender is a member of the network. The disadvantage is that TinyPBC offers no easy way to revoke a compromised node from the network. Another flaw is that TinyPBC does not establish an efficient way to establish cluster keys and thus makes data aggregation more challenging.

In this work, we focus on heterogeneous hierarchically structured networks because current research suggests that they are more efficient and scale better than homogeneous networks [7]. TinyIBE is a protocol already in existence for heterogeneous networks using IBE [21]. However, TinyIBE is extremely vulnerable to attacks on the cluster heads. This is another issue we aim to resolve in our proposed protocol.

3 Assumptions

The following are the major assumptions regarding development of our protocol for wireless sensor networks:

Mobility. For our protocol we assume a semi-mobile network, in which the network topology changes gradually rather than abruptly. This is a reasonable assumption for many applications of WSNs. For instance, the nodes in a WSN deployed in some aquatic environment may shift their positions gradually with tide or water current. Such gradual topological changes can cause some nodes to become detached from their original cluster, which necessitates them either to form a new cluster or join a different cluster. In our protocol, we address the issue how nodes can be integrated to an existing cluster or how a cluster can be formed in a secure manner. This requires mutual authentication as well as establishing new keys between a cluster head and a node.

Base Station's Capacity. Another crucial assumption of our proposed protocols is that the base station is unconditionally secure and has unlimited energy and computational power many times greater than the cluster heads. This is a typical assumption for all existing security protocols for WSNs as it is the case that the base station is normally housed in a physically secure environment. Very often, the base station is a dedicated standard desktop computer whose computational power greatly exceeds that of a cluster head.

Heterogeneity of Nodes. We also assume sensors with higher capabilities (H-sensors) to be cluster heads, and sensors with lower capabilities (L-sensors) to be the normal cluster members. In other words, we assume a heterogeneous WSN where the capabilities of the sensor devices vary in terms of computation power and energy requirement. Current research has shown that this type of network is more scalable and has increased life expectancy over a homogeneous WSN.

Topology. The topology of a wireless sensor network may not be known in advance. Typically a WSN is deployed in an inaccessible or hostile environment and sensor nodes are airdropped. As such, the nodes cannot be carefully deployed in a set pattern but are randomly scattered over the region of interest. Accordingly, we identify the need for establishing clusters immediately after deployment in secure manner.

Node Capture. In wireless sensor networks, adversaries can compromise sensor nodes and use them to attack the networks. With the ability of full control on compromised nodes, the attackers can read all data stored in nodes' memory including information of secret keys. They can also change the behavior of captured sensor nodes to inject malicious code into the network. Although special secure memory devices can be used to prevent attackers from reading compromised nodes' memory, this solution considerably increases the cost of tiny sensor nodes.

We assume that the nodes and cluster heads can be captured, and in that case all data they contain would be known to the adversary.

4 Proposed Security Protocol

The protocol uses Identity Based Encryption and implements five distinct key types that are used for different purposes as explained in the following. We use the following keys for communication:

Global broadcast key. The primary use of this key is to send some encrypted broadcast message by any node including the base station and the cluster heads. Each node is preloaded with this key before deployment. It is to be noted that the global broadcast key G should mostly be used for general messages so its compromise should not greatly affect the security of the network.

Unique key between a node and the base station. This is a

symmetric key shared between a node and the base station that allows a node to communicate directly with the base station and vice versa. Each node is preloaded with this key.

Cluster broadcast key. This is a cluster-specific broadcast key primarily used by the corresponding cluster head to broadcast an encrypted message to all nodes in the cluster. The key cannot be preloaded in the nodes since clusters are formed dynamically after deployment based on nodes' positions. Accordingly, the key is established immediately after deployment.

Cluster pairwise key. The cluster pairwise key is an IBE key that a node needs to communicate in private with any other node within the cluster. This key is generated by the cluster head for each specific node in the cluster.

Global pairwise key. This is a node specific IBE key that a node can use to communicate privately with any other node within the network. Each node is preloaded with this specific key based on its identity.

The keys among the nodes are distributed or established in two phases, namely, the pre-deployment phase and the post-deployment phase. During the pre-deployment phase, sensor nodes are also loaded with functions and parameters that are necessary to establish other keys to handle mobility, addition, and revocation of nodes. It is important to load sensor nodes with keys, functions, and parameters as many as possible before deployment so that a sensor node does not need to spend energy and time to communicate with the base station for such items. However, additional keys are needed to provide cluster-specific security. Those keys and security parameters can be established once clusters have been formed by the network after deployment. In the following, we discuss key distribution and establishment processes in two phases and then examine how the protocol protects the network or reacts to certain anomalies by utilizing the keys, functions, and parameters.

4.1 Pre-deployment Phase

The base station generates a master secret key $\boldsymbol{s}$, a scalar and two elliptic curves $\boldsymbol{E}$ and $\boldsymbol{F}$ that do not intersect. All nodes and cluster heads are loaded with the following:

- $\boldsymbol{id_x}$: The unique identity of node $\boldsymbol{x}$.
- $\emptyset$: A function that maps $\boldsymbol{id_x}$ to a point $\boldsymbol{P_x}$ on $\boldsymbol{E}$ such as $\boldsymbol{P_x} = \emptyset(\boldsymbol{id_x})$
- $\boldsymbol{\delta}$: A function that maps $\boldsymbol{id_x}$ to a point on $\boldsymbol{F}$.
- $\boldsymbol{S_x}$: The global secret key of node $\boldsymbol{x}$ which is computed by the base station as $\boldsymbol{S_x} = \boldsymbol{sP_x} = \boldsymbol{s}\emptyset(\boldsymbol{id_x})$. Essentially $\boldsymbol{S_x}$ is a point on $\boldsymbol{E}$ corresponding to node $\boldsymbol{x}$'s identity and obtained after point multiplication by a secret scalar $\boldsymbol{s}$. Only the base station should know about $\boldsymbol{s}$.
- $\boldsymbol{e}$: A bilinear pairing function.
- $\boldsymbol{K_x}$: The unique symmetric key shared between node $\boldsymbol{x}$ and the base station.
- $\boldsymbol{G}$: The global broadcast key.
- $\boldsymbol{f}$: A function that maps a point on an elliptic curve to a scalar.
- $\boldsymbol{T_x}$: A timestamp embedded in an encrypted message by node $\boldsymbol{x}$.

4.2 Post-deployment Phase

Once the network is deployed, the following steps are to be executed to establish the additional keys that are needed to handle cluster-specific security issues. In the following, we examine how a node x joins a cluster that has a cluster head y. As shown in Figure 2, the scheme involves three steps: 1) Key parameters generation by cluster heads, 2) Cluster membership and node authentication, and 3) cluster head authentication.

4.2.1 Cluster Head Key Generation

It is to be noted that cluster head $\boldsymbol{y}$ is preloaded with necessary key parameters and functions just like any other node. However, it needs to derive additional key parameters to fulfill its role for secure communication within the cluster. Initially, cluster head $\boldsymbol{y}$ computes the cluster secret $\boldsymbol{c}$ by utilizing the function $\boldsymbol{f}$ and its global secret key $\boldsymbol{S_y}$ as: $\boldsymbol{f}(\boldsymbol{S_y}) = \boldsymbol{c}$. The cluster secret parameter, $\boldsymbol{c}$, is used to generate the IBE keys for all nodes in the cluster. In addition, cluster head $\boldsymbol{y}$ also generates a random cluster broadcast key, $\boldsymbol{G_y}$. Once these keys are generated, cluster head $\boldsymbol{y}$ is ready to receive nodes into the cluster. As shown in Figure 2, detailed protocol steps 1.1 and 1.2 depict the process of cluster head key generation.

4.2.2 Cluster Membership and Node Authentication

Organizing the nodes in a WSN into many disjoint clusters facilitates management of security within the network, particularly for damage control. The damage from a compromised node in a cluster can be made limited to a single cluster only. We assume that cluster head $\boldsymbol{y}$ periodically sends out beacon frames encrypted using the global broadcast key $\boldsymbol{G}$. Upon receiving the broadcast message from cluster head $\boldsymbol{y}$, node $\boldsymbol{x}$ decrypts the beacon message and sends an encrypted request to cluster head $\boldsymbol{y}$ to join. The request is encrypted using the key $\boldsymbol{K_{xy}} = \boldsymbol{e}(\boldsymbol{S_x}, \boldsymbol{P_y})$. It is to be noted that $\boldsymbol{P_y}$ is known to node $\boldsymbol{x}$ since $\boldsymbol{P_y} = \emptyset(\boldsymbol{id_y})$.The properties of the bilinear pairing function allow cluster head $\boldsymbol{y}$ to decrypt the message as it can derive the same key due to: $\boldsymbol{K_{xy}} = \boldsymbol{e}(\boldsymbol{S_x}, \boldsymbol{P_y}) = \boldsymbol{e}(\boldsymbol{sP_x}, \boldsymbol{P_y}) = \boldsymbol{e}(\boldsymbol{P_x}, \boldsymbol{P_y})^{\boldsymbol{s}} = \boldsymbol{e}(\boldsymbol{P_y}, \boldsymbol{P_x})^{\boldsymbol{s}} = \boldsymbol{e}(\boldsymbol{S_y}, \boldsymbol{P_x}) = \boldsymbol{K_{yx}}$. It is to noted that $\boldsymbol{P_x}$ is known to cluster head $\boldsymbol{y}$. If node $\boldsymbol{x}$'s identity is valid, cluster head $\boldsymbol{y}$ computes node $\boldsymbol{x}$'s cluster secret by $\boldsymbol{c\delta}(\boldsymbol{id_x}) = \boldsymbol{C_x}$ which is essentially a point on $\boldsymbol{F}$. The process also verifies node $\boldsymbol{x}$'s identity since cluster head $\boldsymbol{y}$ can successfully decrypt the encrypted request sent by node $\boldsymbol{x}$. No separate communication on the part of node $\boldsymbol{x}$ is needed for the verification purpose. The protocol

steps 2.1, 3.1, and 3.2 in Figure 2 describe how a node can join a cluster in secure manner.

It is to be noted that an impostor claiming to be node x might know P_x since $P_x = \emptyset(id_x)$ and can try to join the cluster but it would be infeasible for the imposter to find S_x from id_x due to the difficulty of the elliptic curve discrete logarithm problem. Any attempt of using false S_x in the pairing function would lead to mismatching of the encryption key used by the imposter and the one derived by cluster head y. As a result, the imposter will be denied access to the cluster.

In the following we describe how cluster secret C_x is utilized to authenticate a cluster head y. This mechanism allows the network to defend against any false cluster head.

4.2.3 Cluster Head Authentication

A possible way to attack the network would be to use a compromised L-sensor to masquerade as a cluster head. To defend against such attacks, cluster head $\boldsymbol{y}$'s identity must be verified. One way to verify the authenticity of a cluster head is through the base station. It is to noted that the base station is considered trustworthy in all situations in our protocol. When cluster head $\boldsymbol{y}$ receives the request from node $\boldsymbol{x}$, it sends an encrypted message to the base station, $\boldsymbol{B}$, using the key $\boldsymbol{K_y}$. The message will include $\boldsymbol{C_x}$, $\boldsymbol{G_y}$ (cluster broadcast key), and node $\boldsymbol{x}$'s identity. Since only cluster head $\boldsymbol{y}$ and the base station know $\boldsymbol{K_y}$, cluster head $\boldsymbol{y}$'s identity is authenticated. The base station thus verifies that cluster head $\boldsymbol{y}$ is indeed a cluster head. Next the base station sends node $\boldsymbol{x}$ an encrypted message containing $\boldsymbol{C_x}$ and $\boldsymbol{G_y}$ using shared secret $\boldsymbol{K_x}$. Only the base station and node $\boldsymbol{x}$ know $\boldsymbol{K_x}$ so this verifies that the message is from the base station. If cluster head $\boldsymbol{y}$ was an impostor, then the base station would inform node $\boldsymbol{x}$ of this and revoke cluster head $\boldsymbol{y}$ from the network. Protocol step 4.1 in Figure 2 describes how cluster head authentication is performed.

1.1 Cluster head y computes a cluster secret $c = f(S_y)$ and chooses a random cluster broadcast key, G_y.
1.2 Cluster head y broadcasts an encrypted beacon message M using global key G as:

$$y \rightarrow *: id_y, \{M\}_G$$

↓

2.1 Upon receiving the beacon message from cluster head y, node x decrypts the message and sends an encrypted request R to join the cluster using pairwise key $K_{xy} = e(S_x, P_y)$ as:

$$x \rightarrow y: id_x, \{T_x, M, R\}_{K_{xy}}$$

↓

3.1 Upon receiving the request from node x, cluster head y verifies authenticity of node x by decrypting the message using key $K_{yx} = e(S_y, P_x)$and checking the plaintext for successful decryption.
3.2 Cluster head y computes a cluster secret for node x as $C_x = c\delta(id_x)$ and then sends an encrypted message to the base station B as:

$$y \rightarrow B: id_y, \{id_x, C_x, G_y\}_{K_y}$$

↓

4.1 The base station verifies and authenticates cluster head y by decrypting successfully the encrypted message received since it shares K_y with the cluster head and then sends node x an encrypted message using a shared key K_x as:

$$B \rightarrow x: id_x, \{T_b, C_x, G_y\}_{K_x}$$

Now node x contains all the keys needed to secure future communications.

Figure 2: Protocol steps for node addition immediately after deployment

At this point, node x has all of the keys it needs to be a member of the cluster. It can send and receive cluster broadcasts. If it needs to communicate individually with a node in the cluster, say node z, it uses the key

$(\boldsymbol{C_x}, \boldsymbol{P_z}) = \boldsymbol{e}(\boldsymbol{cP_x}, \boldsymbol{P_z}) = \boldsymbol{e}(\boldsymbol{P_x}, \boldsymbol{P_z})^{\boldsymbol{c}} = \boldsymbol{e}(\boldsymbol{P_z}, \boldsymbol{P_x})^{\boldsymbol{c}} = \boldsymbol{e}(\boldsymbol{C_z}, \boldsymbol{P_x})$. If node $\boldsymbol{x}$'s position changes and another cluster head is closer than node $\boldsymbol{x}$ will request access to that cluster. Once accepted, node $\boldsymbol{x}$ has to erase all previous keys related to the previous cluster. This ensures that if a node is captured the adversary can only compromise one cluster.

It is to be noted that a node needs to communicate once with the cluster head in its neighborhood to establish its remaining cluster parameters and verify authenticity of the cluster head. In this regard, the cluster key establishment process is very energy-efficient for general sensor nodes.

4.2.4 Message Encryption and Decryption

The encryption and decryption schemes that we use in our protocol are based on [2]. Messages sent are encrypted using a simple exclusive-or function. Let $\boldsymbol{A}$ be the pairwise key between nodes $\boldsymbol{a}$ and $\boldsymbol{b}$ and $\boldsymbol{M}$ be the message. Then the encrypted message is $\boldsymbol{V} = \boldsymbol{M} \oplus \boldsymbol{A}$. Note that for pairwise keys between nodes $\boldsymbol{a}$ and $\boldsymbol{b}$ can be computed as: $\boldsymbol{A} = \boldsymbol{f}(\boldsymbol{e}(\boldsymbol{S_a}, \boldsymbol{P_b})) = \boldsymbol{f}(\boldsymbol{e}(\boldsymbol{S_b}, \boldsymbol{P_a})$. Messages are decrypted by applying the exclusive-or operation again as:

$$\boldsymbol{M} = \boldsymbol{V} \oplus \boldsymbol{A} = \boldsymbol{M} \oplus \boldsymbol{A} \oplus \boldsymbol{A}.$$

4.3 Node Addition

As a network ages, its nodes will eventually stop functioning. This may be due to depletion of the battery, environmental damage, or physical damage by an adversary. In order to extend the life of the network, there must be a secure way to replenish the network with new nodes. Our protocol provides a simple way to accomplish this task. The base station preloads all new nodes the same way it does for the original nodes. Once the new nodes are dropped into the network each of them can request to join a cluster. A cluster head can authenticate a requesting node using the process described in

Section 4.2 and similarly, a new node can authenticate a cluster head as described in the same section. The process can be repeated as many times as needed to extend the life of the deployed network. Any false node will not be able to join the network because the corresponding cluster head will not be able to decrypt its request to join and the message will be ignored. There is no way an imposter node can pose as a valid node because the adversary cannot know the master secret $\boldsymbol{s}$ and therefore, cannot know $\boldsymbol{S_x}$ for any node.

5 Security Assessment and Key Management

WSNs are vulnerable to many different types of attacks. Particularly if they are scattered throughout a hostile environment, the physical compromise of individual nodes is a very real threat. There are many attacks that can be employed against WSNs [12]. In the following, we examine a few of them and demonstrate how our protocol is able to withstand them. We assume that using some monitoring or intrusion detection mechanism, the base station or some other supervisory station will be able to identify compromised nodes and cluster heads within the deployed network.

5.1 Node Compromise

When a node $\boldsymbol{x}$ in a cluster served by cluster head $\boldsymbol{y}$ is compromised, the adversary will be able to:

- Send encrypted messages within the cluster using $\boldsymbol{G_y}$ as well as correspondingly decrypt messages sent by other nodes in the cluster using $\boldsymbol{G_y}$.
- Send encrypted messages using the cluster pairwise keys and correspondingly decrypt messages sent by other nodes using the cluster pairwise keys.
- Send encrypted messages using $\boldsymbol{G}$ as well as decrypt any message encrypted using $\boldsymbol{G}$.
- Send and receive encrypted messages from the base station using $\boldsymbol{K_x}$.
- Join another cluster.

If detected on time, the damage due to a node compromise can be made limited to a single cluster with some careful considerations on the usage of the keys. The global broadcast key G should primarily be used for general messages so its compromise should not greatly affect the security of the network. Otherwise, a compromised node can pose as a base station to take over the network by sending encrypted key update messages using G. Therefore, it is imperative that no node should be allowed to modify any key parameters if instructed to do so by an encrypted message using G.

Alternatively, μTESLA, a protocol proposed by Perrig et al. to support authenticated broadcast in wireless sensor network, can be used by the base station only if there is such a need for extra security [17].

An adversary can use the captured information to create a duplicate node and try to join another cluster. This can be prevented by having the base station periodically scan the network for duplicate node identities. The adversary cannot create a node with a different identity with this information because the adversary would not know the master secret s.

Once a compromised node is detected, the following steps are to be executed to restore and mange security in the network by updating all relevant keys: 1) update master secret key, 2) update cluster keys, and 3) update node keys. Figure 3 depicts the protocol steps that can be followed to recover the network from a node compromise.

5.1.1 Update Master Secret Key

The master secret key $\boldsymbol{s}$ and all other associated keys need to be updated for recovery from node compromise. Accordingly, the base station generates a random scalar, $\boldsymbol{d}$ and a new global broadcast key $\boldsymbol{G'}$. It sends $\boldsymbol{d}$ and $\boldsymbol{G'}$ to each cluster head $\boldsymbol{y}$ using $\boldsymbol{K_y}$. The base station then performs the computation $\boldsymbol{ds = s'}$. This $\boldsymbol{s'}$ value is the new master secret key. In Figure 3, protocol steps 1.1 and 1.2 describe the process for updating the master secret.

5.1.2 Update Cluster Secret Keys

As stated in protocol step 1.2 in Figure 3, the base station sends an encrypted message containing a secret scalar $\boldsymbol{d}$ to all cluster heads. Each cluster head $\boldsymbol{y}$ receives the $\boldsymbol{d}$ value and performs the computation $\boldsymbol{dS_y = S'_y}$. This synchronizes their master secret with that of the base station since $\boldsymbol{S'_y = dS_y = dsP_y = s'P_y}$. Then each cluster head $\boldsymbol{y}$ updates its cluster secret key by computing $\boldsymbol{dc = c'}$. After that, the cluster heads in uncompromised clusters send $\boldsymbol{d}$ and $\boldsymbol{G'}$ to their respective nodes using their respective cluster broadcast key $\boldsymbol{G_y}$. In a cluster with compromised nodes, the cluster head sends $\boldsymbol{d}$ and $\boldsymbol{G'}$ to each uncompromised node using the bilinear pairwise keys generated using $\boldsymbol{C_y}$. (This is acceptable because a compromised node would not be able to decrypt pairwise communication that is not sent to it.) If a node of the compromised cluster is a subcluster head, then it resends the message to members in its cluster group using its cluster broadcast key $\boldsymbol{G_y}$ (this process continues for all levels of subclusters). Each cluster head $\boldsymbol{y}$ updates its cluster pairwise IBE key by computing $\boldsymbol{dC_y = C'_y}$. The update process for cluster secret keys is shown in protocol steps 2.1 and 2.2 in Figure 3.

5.1.3 Node Update and Key Deletion

As shown in protocol step 2.3 in Figure 3, a cluster head broadcasts the $\boldsymbol{d}$ value to all nodes in its cluster. Once a node $\boldsymbol{x}$ receives $\boldsymbol{d}$ from its cluster head, it computes $\boldsymbol{dS_x = S'_x}$ and $\boldsymbol{dC_x = C'_x}$. This synchronizes the nodes to the new master key held by the base station as stated above. It also synchronizes the node to the new cluster key since $\boldsymbol{C'_x = dC_x = dcP_x = C'P_x}$. Once the cluster heads are done sending the new keys, they delete all old keys. After the nodes finish the new key computations, they delete all old keys as well. The base station also deletes the key $\boldsymbol{K_x}$ that it shares with the compromised node $\boldsymbol{x}$. Protocol step 3.1 in Figure 3 shows the process for key update by nodes.

Now the network is secure from the compromised node. It no longer has the cluster or global broadcast keys, and it cannot communicate with the base station using K_x. If it attempts to communicate with a node in its former cluster using pairwise communication, the keys will no longer be the same. For example, the compromised node x attempts to contact node y:

$e(C_x, \delta(id_y)) = e(C_y, \delta(id_x)) \neq e(C'_y, \delta(id_x))$. For similar reasons, the compromised node will not be able to request access to another cluster.

5.2 Cluster Head Compromise

In the event that a cluster head is captured, the attacker can sever all nodes in the cluster from the rest of the network. Although the keys of the rest of the network can be updated using the procedure described in Section 5.1, but the nodes in the compromised cluster must be recovered by the base station separately. To recover the nodes, the base station can attempt to send a message notifying the nodes of the attack so they can join a new cluster. The base station must know which nodes belong to the compromised cluster.

1.1 The base station chooses a random number d and computes a new master secret $s'$where $s' = ds$.
1.2 The base station sends an encrypted message to each cluster head y as:

$$B \rightarrow y: B, \{T_b, d, G'\}_{K_y}$$

↓

2.1 Each cluster head y decrypts the message received from the base station and computes $dC_y = C'_y$, a new cluster key and $dc = c'$, a new cluster secret.
2.2 Cluster head y'containing the compromised node in its cluster sends selectively a message to each non-compromised node x as:

$$y' \rightarrow x: y', \{T_{y'}, d, G'\}_{K_{xy'}}$$

2.3 Cluster head y of each non-compromised cluster broadcasts:

$$y \rightarrow *: y, \{T_y, d, G'\}_{G_y}$$

↓

3.1 Upon receiving the message from the cluster head, node x decrypts the message and replaces G with G', S_x with S'_x where $S'_x = dS_x$, and C_x with C'_x where $C'_x = dC_x$.

Figure 3: Protocol steps for recovery from node compromise

The detailed protocol steps are shown in Figure 4. Once a compromised cluster head is detected, the base station sends an encrypted message using the base station to node key K_x for each node x, notifying each node in that cluster of the attack. If a node of the compromised cluster is a subcluster head, then it resends the message to members in its cluster group using its cluster broadcast key G_y (this process continues for all levels of subclusters). The message will also contain a new global broadcast key G' and a new d value that is used to generate a new secret $S'_x = dS_x$ for each node. The base station will also follow the node revocation scheme to update the keys of the non-compromised nodes. The message will reach the nodes if there is some node in a different cluster that can route the messages to them. Otherwise, the nodes would have to be recovered physically. If the message reaches the nodes, then they will attempt to join a new cluster group by either directly communicating with another cluster head or by communicating to a cluster head using multi-hop routing. Thus the nodes of a compromised cluster can be recovered by the protocol.

5.3 False Cluster Head

In this attacking scenario, an attacker may mimic a cluster head either without prior knowledge of the network or by using a compromised node. In the following, we describe how the network can be recovered from each case of security breaches involving false cluster heads.

5.3.1 False Cluster Head with No Prior Network Knowledge

A node joins a cluster by responding to a beacon frame broadcast by a cluster head. Since beacon frames are encrypted, the attacker with no prior knowledge of the network may only attempt to mimic another cluster head by replaying captured beacon frames. However, the nodes receiving the message will discard the frame since the timestamp in the message will be older than the threshold transmission time for beacon frames. It is to be noted that all protocol messages are time-stamped as shown in Figures 2, 3, and 4.

5.3.2 Compromised Node as Cluster Head

Each node, according to our protocol, has the global broadcast key $\boldsymbol{G}$. As such a compromised node can advertise itself as a new cluster head or as an existing cluster head (since all identities are public). This type of attack may only occur during the window in which the network keys of some of the nodes have not yet been updated (see Section 5.1). Nevertheless, such attacks will not succeed. If a node attempts to join the new cluster, then attacking node must communicate with the base station since the nodes receive their cluster keys from the base station ultimately, as stated in protocol steps in Figure 2. If the attacker sends a message to the base station, then the base station will detect the attack by some duplicate cluster head function or by noting that the attacker is not in the list of approved cluster heads. If the attacking node posing as cluster head does not communicate with the base station, then the node will not receive any cluster key from it. After all of the network keys have been updated by the recovery process, the attacking node will no longer be able to communicate with any other nodes.

5.4 Sybil Attack

Sybil attacks are defined by Douceur as "the forging of multiple identities" on a network [6]. A Sybil attack may be achieved by an attacker node presenting multiple identities to other nodes or by duplicating a compromised node. Douceur demonstrated that a central authentication authority is

necessary to prevent Sybil attacks. In our protocol, we use the base station as our trusted central authority. Because the keys for the nodes in our network are authenticated by the base station, an attacker may only attempt to execute a Sybil attack by compromising an existing node.

If an attacker compromises a node, we assume that the intrusion detection system will detect the attack. Then the base station will commence the process described in Section 5.1 that deals with recovery from a node compromise. However, signal interference may delay the update of network keys. During this short period, an attacker may either attempt to present multiple identities to cluster heads or to other nodes. However, the attacker will not be successful since it can only create bilinear pairwise key using the identity of the compromised node. Alternatively, the attacker may try to install duplicates of the compromised node in the network. However, the compromised node would not be able join any new clusters since the cluster node addition must be authenticated by our central authority, the base station. According to our protocol, such attempts for any unauthorized node addition will fail.

1.1 The base station chooses a random number d and computes a new master secret $s' = ds$.
1.2 The base station sends an encrypted message to each cluster head y except to the compromised cluster head as:

$$B \rightarrow y : B, \{T_b, d, G'\}_{K_y}$$

↓

2.1 Each cluster head y decrypts the message received from the base station and computes $dC_y = C'_y$, a new cluster key and $dc = c'$, a new cluster secret.
2.2 Cluster head y broadcasts a message to all nodes in the cluster as:

$$y \rightarrow *: y, \{T_y, d, G'\}_{G_y}$$

↓

3.1 Upon receiving the message from the cluster head, node x replaces G with G', S_x with S'_x where $S'_x = dS_x$, and C_x with C'_x where $C'_x = dC_x$.

a) Updating keys of nodes in non-compromised clusters

1.1 The base station sends an encrypted message to each node x in the cluster with the compromised cluster as:

$$B \rightarrow x: B, \{T_b, d, G'\}_{K_x}$$

↓

2.1 Upon receiving the message from the base station, node x in the compromised cluster replaces G with G', S_x with S'_x where $S'_x = dS_x$, and C_x with C'_x where $C'_x = dC_x$.

b) Updating keys of nodes in the compromised cluster

Figure 4: Protocol steps for recovery from cluster head compromise

5.5 Replay Attack

A replay attack refers to the replay of messages in a network. We consider replay attacks in our protocol because it could be used to drain the battery of a sensor node. An attacker could replay messages that require a response. Then the recipient would send out a response, expending energy. In addition, some replayed messages can cause erroneous updates at nodes. To prevent these scenarios, we require a clock based timestamp $\boldsymbol{T}$ to be placed on each message as discussed in [5]. We assume that the clocks of each node and cluster head are initially synchronized with the base station. Let $\Delta\boldsymbol{t}$ be the normal discrepancy between the clock of the base station and the local clock $\boldsymbol{C}$ of any node or cluster head. Then a recipient of a message may verify that a message has not been replayed by confirming that $|\boldsymbol{C} - \boldsymbol{T}| < \Delta\boldsymbol{t}$.

6 Performance Evaluation

In evaluating the performance of our protocols, we consider three factors: storage costs, computation costs, and communication costs. We consider these factors due to the constraints of sensor nodes and to show the feasibility and efficiency of our protocols.

6.1 Storage Costs

Memory requirement for storage of keys can be computed in straight-forward manner. Each sensor node maintains five keys: two of them are used for IBE and the remaining three are for symmetric key cryptography. As reported in [14], TinyPBC uses 272 bit keys for IBE, sufficient for providing equivalent security that can be achieved using 128-bit symmetric keys or 3072 RSA keys.

Thus, two of such keys require 544 bits. The other three keys are standard 128 bit keys. That means the total space for the keys is 928 bits or 116 bytes for each node. The amount of space required for keys is constant and not dependent on the number of nodes in the network. This makes our protocol scalable for large networks. There is only one instance where more storage space for keys would be necessary. That is, if a pair of nodes does a great deal of communication that cannot be broadcast with the cluster broadcast key. Instead of performing the pairing computation every time the nodes communicate, the nodes can set up a secret key between them. The exact conditions for setting up a secret key can be customized for each network based on the availability of storage space and the computational power of the nodes.

In addition to storage for keys, each sensor node requires TinyPBC to be loaded in its memory, which, according to [14], requires 2867 bytes of stack, 368 bytes of RAM, and 47,948 bytes of ROM if implemented on an ATmega128L based sensor node.

6.2 Computation Costs

Three of the five keys use a simple XOR function to encrypt and decrypt. This is a simple computation and does not add any significant amount of overhead. The cluster request and cluster pairwise keys both require the computation of a bilinear pairing function. This is a computationally expensive

operation. Table 1 shows the number of bilinear pairing operations to be done by a node, a cluster head, or the base station under various situations. The bilinear pairing function we consider is the same one used in the TinyPBC protocol [14]. It is reported that it takes an ATmega128L node 1.90 seconds and a 13 MHz Imote2 node 0.14 seconds to evaluate the pairing. Since the cluster request key is used only for the purpose of joining a cluster, it will not be frequently used. The cluster bilinear pairwise key will mainly be used for a node to communicate with its cluster head. If the computation is deemed too expensive then a secret key can be established between a node and cluster head. This sacrifices storage space so the trade-offs will have to be weighed for each application.

Table 1: Computation costs

	Number of Pairings		
	Node	***Cluster Head***	***Base Station***
Initial Setup	1	$\boldsymbol{m}$	0
Migration of a Node	1	1	0
Recovery from Node Compromise	0	$(\boldsymbol{m}-\mathbf{1})^*$ or 0	0
Addition of a Node	1	1	0
Recovery from Cluster Head Compromise	0	0	0

Note: * For the compromised cluster only.

6.3 Communication Costs

Communication is the biggest drain on the battery of a sensor node. Because of this, keeping the communication overhead low is vital for any WSN security protocol. Our protocol requires low communication overhead to set up keys. We consider the communication costs of various important scenarios including cases for node addition, node compromise, and cluster head compromise. Table 2 summarizes communication costs in terms of number of communications performed by a node, a cluster head, and the base station.

Initial setup. Immediately after deployment, all nodes must communicate to establish their cluster membership and corresponding keys and secrets. During this initial setup, as shown Figure 2, node $\boldsymbol{x}$ communicates only once with cluster head $\boldsymbol{y}$ (step 2.1) and cluster head $\mathbf{y}$ communicates with base station $\boldsymbol{B}$ for each node in the cluster (step 3.1). If there are $\boldsymbol{m}$ nodes in the cluster for cluster head $\boldsymbol{y}$, then there are $\boldsymbol{m}$ communications done in all by cluster head $\boldsymbol{y}$. In addition, the base station carries out step 4.1 in Figure 2 for each node. If there are $\boldsymbol{n}$ nodes in the network, there will be $\boldsymbol{n}$ such communications by the base station during the initial setup for establishment of keys. Accordingly in Table 2, we provide communications costs for a node, a cluster head and the base station as 1, $\boldsymbol{m}$, and $\boldsymbol{n}$ respectively for the initial setup phase. In our protocol analysis for communication costs, we ignore periodic beacon messages sent by cluster heads. Also, communications performed by the base station to nodes can be ignored from the consideration of communication costs since the base station is typically connected to some external power source or an easily renewable power source.

Node Migration or Addition. Addition of a new node or migration of an existing node to a different cluster requires carrying out steps 2.1, 3.1, 3.2, and 4.1 in the protocol as described in Figure 2. Essentially a new or migratory node has to send a message to a cluster head in response to a beacon message broadcast by the cluster head (step 2.1). As a result, it involves only one communication by the new or migratory node. The corresponding cluster head has to communicate only once to the base station to authenticate the new or migratory node (step 3.2). Finally the base station communicates with the new or migratory node (step 4.1) by sending only one message. Table 2 summarizes the costs as one communication for the new or migratory node, one communication for the corresponding cluster head, and one communication for the base station.

Table 2: Communication costs

	Number of Communications		
	Node	***Cluster Head***	***Base Station***
Initial Setup	1	$\boldsymbol{m}$	$\boldsymbol{n}$
Migration of a Node	1	1	1
Recovery from Node Compromise	0	$(\boldsymbol{m}-\mathbf{1})^*$ or 1	$\boldsymbol{k}$
Addition of a Node	1	2	2
Recovery from Cluster Head Compromise	0	1	$\boldsymbol{m}+(\boldsymbol{k}-\mathbf{1})$

Note: * For the compromised cluster only.

Recovery from Node Compromise. As shown in Figure 3, if a node is compromised, the base station will have to communicate with each cluster head. As a result, if there are $\boldsymbol{k}$ cluster heads in the network, the base station will send $\boldsymbol{k}$ such messages altogether as shown in step 1.2 of Figure 3. Assuming there are $\boldsymbol{m}$ nodes including the compromised node in the cluster, the cluster head will have to send $(\boldsymbol{m}-\mathbf{1})$ messages to all $(\boldsymbol{m}-\mathbf{1})$ uncompromised nodes (except the compromised node) in its cluster to recover them. However, in non-compromised clusters, each cluster head only needs to broadcast a single message to all nodes in its own cluster. It is to be noted that there is no communication required by a sensor node in this case. Accordingly, Table 2 summarizes message transmissions by the base station as $\boldsymbol{k}$ and by a cluster head as 1 or $(\boldsymbol{m}-\mathbf{1})$ depending on whether the cluster contains a compromised node or not. The communication cost for a node in the recovery effort is 0 as shown in Table 2.

Recovery from Cluster Head Compromise. In the case of a cluster head compromise, as depicted in the protocol in Figure 4, the base station has to recover all $\boldsymbol{m}$ modes in the compromised cluster (step 1.2 in Figure 4(b)) as well as all other $(\boldsymbol{k}-\mathbf{1})$ clusters (step 1.2 in Figure 4(a)). Thus, altogether, the base station needs to send $(\boldsymbol{m}+\boldsymbol{k}-\mathbf{1})$ messages. However, each cluster head has to send only one broadcast message to all of its nodes just to forward the message received from the base station (step 2.2). It is to be noted that there is no need to communicate any messages by the sensor nodes. Accordingly in Table 2, the communication

costs for a node, a cluster head and the base station are shown as 0, 1, and $(\boldsymbol{m+k-1})$ respectively in terms of number of message transmissions.

It is to be noted that for all situations mentioned above, a sensor node does not need to communicate more than one message.

7 Conclusion

A new security protocol for semi-mobile heterogeneous wireless sensor networks is proposed based on pairing-based cryptography. Specifically, we propose a new security protocol for WSNs that can recover from any node or cluster head compromise as well as support mobility of nodes within the network. The protocol also allows addition of new nodes in an existing network in secure manner so that the life of an aging network can be extended without disruption of service. The protocol makes use of five different keys to provide security services within the network. These keys are needed to secure communication between the base station and a cluster head, between a cluster head and a node, among nodes within a cluster, and between the base station and nodes in the network. Our analysis shows that the proposed protocol is storage and communication efficient and scales well to large networks. Specifically, no more than one communication is needed by an energy-constrained sensor node to execute the protocol for a specific task, such as initial setup or recovery from some attack. Our protocol can provide defenses against several forms of attacks including Sybil attacks and replay attacks. If comprised nodes or cluster heads are detected, our protocol provides ways to recover the network from such compromises.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1004902. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation of the United States of America.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393-422, 2002.

[2] D. Boneh and M. Franklin, "Identity based encryption from the weil pairing," *Cryptology ePrint Archive*, Report 2001/090, 2001. http://eprint.iacr.org/

[3] G. V. Crosby, L. Hester, and N. Pissinou, "Location-aware, trust-based detection and isolation of compromised nodes in wireless sensor networks," *International Journal of Network Security*, vol. 12, no. 2, pp. 107-117, 2011.

[4] A. K. Das, "Improving identity-based random key establishment scheme for large-scale hierarchical wireless sensor networks," *International Journal of Network Security*, vol. 14, no. 1, pp. 1-21, 2012.

[5] D. E. Denning and G. M. Sacco, "Timestamps in key distribution protocols," *Communications of the ACM*, vol. 24, pp. 533-536, 1981.

[6] J. R. Douceur and J. S. Donath, "The sybil attack," *The 1st International Workshop on Peer-to-Peer Systems*, pp. 251-260, 2002.

[7] E. J. Duarte-Melo and M. Liu, "The effect of organization on energy consumption in wireless sensor networks," *IEEE Global Communications Conference (Globecom)*, 2002.

[8] A Gupta, A. D. C. Mazieux, and M. Becker, "Effect of topology on the performance of mobile heterogeneous sensor networks," *The Sixth Annual Mediterranean Ad Hoc Networking Workshop*, pp. 100-105, Corfu, Greece, 2007.

[9] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 2003.

[10] A. Joux, "The weil and tate pairings as building blocks for public key cryptosystems," *ANTS-V: the 5th International Symposium on Algorithmic Number Theory*, pp. 20-32, 2002.

[11] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," *The 2nd ACM SensSys*, pp. 162-175, 2004.

[12] T.-G. Lupu, "Main types of attacks in wireless sensor networks," *The 9th WSEAS International Conference on Signal, Speech and Image processing, and 9th WSEAS International Conference on Multimedia, Internet, and Video Technologies (SSIP '09/MIV'09)*, pp. 180-185, Wisconsin: World Scientific and Engineering Academy and Society (WSEAS), 2009.

[13] M. Mana, M. Feham, and B. L. Bensaber, "Trust management scheme for wireless body area networks," *International Journal of Network Security*, vol. 12, no. 2, pp. 71-79, 2011.

[14] L. Oliveira, M. Scott, J. Lopez, and R. Dahab, "Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks," *The 5th International Conference on Networked Sensing Systems*, pp. 173-180, 2008.

[15] L. B. Oliveira, D. Aranha, E. Morais, F. Daguano, J. L´opez, and R. Dahab, "TinyTate: Computing the tate pairing in resource-constrained nodes," *The 6th IEEE International Symposium on Network Computing and Applications*, pp. 318-323, 2007.

[16] G. Padmavathi and D. Shanmugapriya, "A survey of attacks, security mechanisms and challenges in wireless sensor networks," *CoRR*, vol. abs/0909.0576, 2009.

[17] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5. pp. 521-534, 2004.

[18] R. Rajagopalan and P. K. Varshney, "Data aggregation techniques in sensor networks: a survey," *IEEE Communications Surveys and Tutorials*, vol. 8, pp. 48-63, 2006.

[19] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," *Symposium on Cryptography and Information Security (SCIS'00),* pp. 26-28, 2000.

[20] A. Shamir, "Identity-based cryptosystems and signature schemes," *Advances in Cryptology* (G. Blakley and D. Chaum, eds.), *Lecture Notes in Computer Science*, vol. 196, pp. 47-53, New York: Springer-Verlag, 1985.

[21] P. Szczechowiak and M. Collier, "TinyIBE: Identity-based encryption for heterogeneous sensor networks," *The 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 319-354, 2009.

[22] A.Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," *The Third IEEE International Conference on Pervasive Computing and Communication (PerCom 2005)*, Hawaii, 2005.

[23] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 8, pp. 2-23, 2006.

[24] R. Watro, D. Kong, S.-F. Cuti, C. Gardiner, C. Lynn, and P. Kruu, "TinyPK: securing sensor networks with public key technology," *The 2nd ACM workshop on Security of ad hoc and sensor networks*, 2004.

[25] X. Xiong, D. Wong, and X. Deng, "TinyPairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks," *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1-6, 2010.

[26] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," *The 10th ACM Conference on Computer and Communication Security (CCS'03),* pp. 62-72, ACM Press, 2003.

Dulal Kar is currently an Associate Professor of Computer Science in the School of Engineering and Computing Sciences at Texas A&M University – Corpus Christi, Texas, USA. Previously, he was a faculty in the Department of Computer Science at Virginia Polytechnic Institute and State University, Virginia, USA; Mountain State University, West Virginia, USA; and Bangladesh University of Engineering and Technology, Bangladesh. He received the B.Sc.Engg. and the M.Sc.Engg. degrees from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh and the MS and the Ph.D. degrees from North Dakota State University, Fargo, North Dakota. He is the lead editor of the reference book, *Network Security, Administration and Management: Advancing Technology and Practice* published by IGI Global and an editorial board member of the International Journal of Network and Computer Applications, a publication by Elsevier. His research interests include wireless sensor networks, signal and image processing algorithms, network architecture and performance measurement, network and information security, information retrieval, and educational technology. He has published over fifty refereed journal and conference articles in those areas. Many of his research works have been funded by grants from NSF, NASA, and DoD of the United States of America.

Rashad Tatum is currently pursuing his BS degree in Computer Science and Mathematics at Southern Polytechnic State University, Georgia, USA. He was an NSF (National Science Foundation) undergraduate research participant for the summer 2011 REU (Research Experience for Undergraduates) program offered at Texas A&M University-Corpus Christi. His future plans include research in computational complexity, quantum computing, and algorithms. He is a student member of the Association of Computing Machinery (ACM) and the Mathematical Association of America (MSA).

Keith Zejdlik received his BS degree in Mathematics from Texas A&M University-Corpus Christi in 2011. He was an NSF (National Science Foundation) undergraduate research participant for the summer 2011 REU program offered at Texas A&M University-Corpus Christi. He now works in the IT department of Govind Development, an engineering firm near Corpus Christi, Texas and plans to pursue an MS degree in either Mathematics or Computer Science. His main research interest is cryptography.