

An Efficient Pairwise and Group Key Management Protocol for Wireless Sensor Network

Musfiq Rahman¹ · Srinivas Sampalli²

Published online: 2 April 2015

© Springer Science+Business Media New York 2015

Abstract Although security services depend on the key management system, they do not specify how to exchange keys securely. Many security-critical applications depend on key management processes to operate but also demand a high level of fault tolerance when a node is compromised. Due to constraints on wireless sensor networks (WSNs) traditional security mechanisms cannot be applied directly. Moreover, to be robust, the key management protocol should not only be lightweight but also should support both pairwise and group-wise communications. In this paper, we propose an efficient key management protocol for WSNs that provides support for both pairwise and group-wise key management. We have implemented our scheme in the TelosB mote and present some experimental results on computational, communication, energy and storage overheads. We compared the performance with other symmetric and public key based key management protocols used in WSNs. The results show our protocol is efficient in terms of computation, communication and energy overhead. It can update both pairwise and group keys to support key revocation, key refresh and new node addition.

Keywords Key management in WSN · Group key management · WSN security · Key distribution · Key refresh

The paper is a revised and extended version of a paper entitled “A robust pair-wise and group key management protocol for wireless sensor network” presented at the IEEE Globecom Workshop on Web and Pervasive Security (WPS), December, 2010, Florida, USA.

✉ Musfiq Rahman
mrahman@tru.ca
Srinivas Sampalli
srini@cs.dal.ca

¹ Department of Computing Science, Thompson Rivers University, Kamloops, BC, Canada

² Faculty of Computer Science, Dalhousie University, Nova Scotia, NS, Canada

1 Introduction

A wireless sensor network (WSN) is an emerging network technology that will take the convenience of using networks one step further. In such networks, tiny battery operated wireless sensors are deployed typically in harsh environments to monitor and collect continuous, real-time data such as temperature, moisture, light intensity, pressure, etc. All the wireless sensor nodes send the collected data to a base station node either directly or via other sensor nodes. The base station node then sends the data to a remote location via the Internet or a WiMAX network for further processing.

The proliferation of advanced research in this area shows the possibility of deploying this kind of network in numerous applications ranging from monitoring the growth of the crops to crucial military operations. However, WSNs are prone to many types of malicious attacks, such as impersonating, masquerading, interception, denial-of-service attacks. Therefore, the practical deployment of the technology is not possible without considering the security implications of the technology.

Providing security in WSNs is a challenging task because of its characteristics such as low power sensor nodes, wireless communication medium, limited computational power, lack of infrastructure and topology, dense network size, and unattended deployment mode. Due to the different types of constraints, other existing security mechanisms for wired networks cannot be adapted to these networks directly. Many security services depend on cryptographic operations such as encryption and decryption, which require a key to be fed in to perform the operations. A critical task of key management is to establish and manage the key between the pair of sensor nodes. Although, security services depend on the key management system, they do not specify how to exchange keys securely. This leaves open the key management problem that is the focus of much recent research.

In WSNs, many security-critical applications depend on key management processes to operate, but also demand a high level of fault tolerance when a node is compromised. Moreover, depending on the deployment, WSNs require a node-to-node (pairwise) and group-oriented communication. To facilitate these modes of communications a robust, lightweight yet secure security protocol is necessary.

In the recent past, many researches such as [6, 30, 31, 33] have focused on finding a distributed low cost pairwise key establishment techniques that can survive node compromises. However, very few works [23, 29] address both the pairwise and group oriented key management problem together. Moreover, new node admission problem, key revocation and periodic key material update have received very less attention, whereas these tasks are very crucial to keep the network operational for longer periods of time.

In this paper, we propose a new, efficient key management protocol that combines both the pairwise key and group key establishment together. In addition, our protocol facilitates the new node addition, and key revocation or refresh without compromising the security. Our protocol is fast, requires very little memory and data exchange. This makes it suitable for dynamic sensor networks where the key computation must be achieved quickly and cheaply.

The rest of this paper is organized as follows. We discuss the review of related works in Sect. 2. Section 3 presents our proposed key management protocol. Sections 4 and 5, describes the security and performance analysis of our proposed protocol. We compare the performance of our scheme with other schemes in Sect. 6. Finally, we conclude in Sect. 7.

2 Related Work

In this section, we first present some related works on key management in WSNs and then we discuss the background necessary to understand our proposed scheme and finally we summarize our main contributions.

2.1 Literature Review

A key management protocol is one of the first requirements for setting up a secure communication in a WSN. A naive solution like the use of a *single network wide key* would not work in WSNs because compromising one node in the network, which is the most common type of attack in WSNs, will expose the network wide key [23, 33]. To overcome this problem, the *pairwise* keying model employs $n - 1$ keys in each node in the network. Where n is the total number of nodes in the network. Although, this scheme performs well for a small sized network, for a large network this scheme becomes impractical. Moreover, the new node addition is a big concern since all the other nodes must store the key of the newly added node in order to communicate with this node. Similarly, key revocation and refreshing processes also suffer from the scalability issue. The *group* keying model combines the features of *single network* keying and *pairwise* keying mechanisms. Within a group, communications are performed using a single shared key similar to the network keying, but, communications between groups employ a different key between each pair of groups in a manner identical to the pairwise keying scheme. This scheme has better resilience against node capture and better scalability than the *single network* keying scheme and *pairwise* scheme respectively. However, the problem with this scheme is that it is difficult to set up and the formation of the groups varies with applications.

Public key cryptography (PKC) such as Rivest, Shamir, and Adleman (RSA) or elliptic curve cryptography (ECC) has been used in Internet for solving the key management problem. However, due to its high-energy consumption and storage requirements, it is not suitable for many WSN architectures. Besides, authentication through certificate verification is computationally expensive for WSN nodes. In contrast, symmetric key based schemes are efficient in terms of computation and energy consumption. Therefore, symmetric key based schemes are widely used in WSNs. However, the challenge, in this case, is to provide a balance between the key storage and maximum secure connectivity.

Many researchers have tried to solve the key storage problem by considering the pre-deployment knowledge in the key management scheme. For example, [16] and [32] respectively propose key management schemes using the Gaussian distribution for deployment knowledge and location knowledge for relatively static sensor networks. On the other hand, a mix of heterogeneous node capabilities in terms of memory and computational power is considered for key management protocols such as [6, 11, 19, 27].

Eschenauer and Gligor in [8] propose a random key pre-distribution scheme based on probabilistic key sharing among the nodes of a random graph. In this scheme, initially, every node randomly selects a subset of keys from a large key pool and stores the keys and their identities in memory. If two neighboring nodes want to communicate securely, they first exchange and compare the list of identities in their key-chains. If they happen to find a match, then they can establish a direct secure link between these them, otherwise they need to select some intermediate nodes with a common key between the two sensor nodes to establish a common session key. That session key is then used as the path key to the selected sensor-node pairs. This approach is better in terms of resilience to node capture

since in case a malicious attacker captures a node it only reveals a subset of keys. However, this scheme still requires a large number of keys to be stored in the sensor node and a large number of messages need to be exchanged between the nodes to get shared key. Moreover, this approach does not define any process for revoking and refreshing keys. Many researchers have taken this approach and modified it for better performance [5, 6, 34]. However, in all of these schemes, a small portion of compromised nodes affects a large portion of networks since every node compromise rapidly increases the fraction of affected pairwise keys.

Bloms key distribution approach [2], which is also a special case of Blundo et al. [3] polynomial based key generation scheme allows any pair of nodes in a network to be able to find a pairwise secret key. This approach is unconditionally secure and can tolerate up to λ compromised nodes without breaking any security of the network. Lui and Ning [16] combine the Blundos scheme [3] with the Eschenauer and Gligor scheme [8] to increase network resilience to node capture attack. Similarly, Du et al. [6] developed a multi-space random key pre-distribution scheme, which combines the Eschenauer and Gligor scheme [8] and the Blom scheme [2] to substantially improve network resilience against node capture over existing schemes, without increasing the memory overhead. In addition, Mee et al. proposed a symmetric key pre-distribution scheme [30, 31] suitable for WSNs by modifying the Blom scheme [2] to use multiple-keys. However, all these schemes are more complex and increases the overhead cost. In addition, they lack the key revocation or key refresh process.

Although all the above works discuss efficient schemes for key management, very few works [23, 29, 33] have been done on group key management along with pairwise key distribution using the same secret key materials. Moreover, key material refresh, updating the key for group membership change, new node addition, and key revocation issues are not addressed in many papers.

2.2 Background: Blom's Scheme

Blom presented a key distribution method [2] that allows any pair of nodes in a network to be able to find a pairwise secret key as long as no more than λ nodes are compromised. It involves two phases, *key pre-distribution* and *key agreement*.

2.2.1 Key Pre-distribution Phase

In this phase, the base station first constructs a $(\lambda + 1) \cdot N$ matrix G over a finite field $GF(q)$, where N is the number of nodes in the network and $GF(q)$ is a prime number large enough to accommodate the key size. The matrix, G , is considered public information and everyone including adversary may have information about it. Then the base station creates a random $(\lambda + 1) \cdot (\lambda + 1)$ symmetric matrix D over $GF(q)$, and computes $N \cdot (\lambda + 1)$ matrix $A = (D \cdot G)^T$, where $(D \cdot G)^T$ is the transpose of $(D \cdot G)$. Matrix D is the secret information and should not be disclosed to anyone (although only one row of A will be distributed to each sensor node). It is easy to see that since D is symmetric matrix, the key matrix $K = A \cdot G$ can be written as:

$$K = A \cdot G = (D \cdot G)^T \cdot G = G^T \cdot D^T \cdot G = G^T \cdot D \cdot G = (A \cdot G)^T = K^T \quad (1)$$

This means that K is also a symmetric matrix. Therefore, $K_{ij} = K_{ji}$, where K_{ij} is the element of the i th row and j th column. Node i uses K_{ij} and node j uses K_{ji} as pairwise key to

communicate with each other. Figure 1 depicts an illustration of the pairwise key $K_{ij} = K_{ji}$ generation process. Finally, the base station distributes, for $k = 1 \dots N$: k th row and k th column of matrix A and G respectively to node k as its key material.

2.2.2 Key Agreement Phase

When any pair of nodes, i and j want to find a pairwise key between them, they first exchange their columns of G matrix, then they can compute K_{ij} and K_{ji} , respectively, using their private rows of A . Note that the exchange of the columns of G matrix can take place in plain text since G is public information. It has been shown in [2, 3] that the above scheme is λ -secure when any $\lambda + 1$ columns of G matrix are linearly independent such as Vandermonde matrix.

2.3 Our Main Contributions

The original Blom's scheme [2] was not proposed for sensor networks. Therefore, to use it in resource constraint WSNs, we have modified the original Blom's scheme such that without any message exchange and only knowing each other IDs, any pair of nodes is able to generate pairwise keys. The modification reduces significant memory and communication overheads from the original Blom's scheme while retaining all the benefits of the original scheme, i.e., mutual authentication, simple computations, and little energy consumption. In addition, we have proposed mechanisms for key material update/refresh, new node addition, and node revocation mechanisms. We have also presented a dynamic, on-the-fly secure group key generation mechanism that uses the pairwise keys. This makes it flexible and especially suitable for many sensor network applications where the nodes are mobile and must support light-weight pairwise and group-wise communications.

3 Proposed Scheme

In this section, we introduce our proposed key management protocol. In our protocol, we assume that all sensor nodes have similar characteristics in terms of memory, processor, radio range, and power consumption, except the base station (BS), which is a trusted and powerful node with its own power source. In addition, in case of the group key generation, we assume that the clock speed of the node is loosely synchronized with the bases station node's clock. We do not assume any pre-deployment knowledge of the network, i.e., our

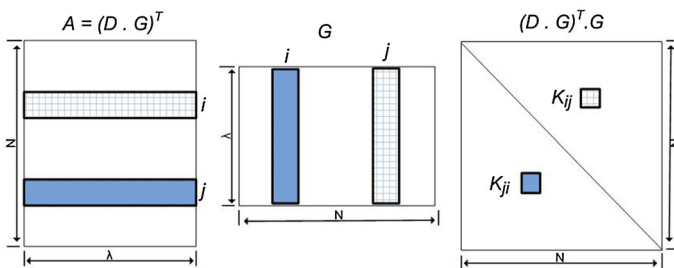


Fig. 1 Pairwise key generation in Blom's scheme

protocol should work in any heterogeneous deployment mode. In addition, our protocol does not assume any particular network topology. In other words, our protocol should work on both the flat and hierarchical networks. In our proposed protocol we have used the modified Blom’s scheme for both pairwise and group-wise key management.

3.1 Pairwise Key Management

Pairwise key generation involves the following phases in our protocol.

3.1.1 Key Pre-distribution Phase

Our key pre-distribution phase is similar to the key pre-distribution phase of modified Blom’s scheme, except for constructing G matrix, we used linearly independent *Vandermonde* matrix [17]. Although any linearly independent matrix can be used for generating the G matrix, we have chosen *Vandermonde* matrix because it is easy to generate and distinct node IDs can be used as seeds to generate it. Therefore, in our scheme, we use the node IDs as seeds for constructing G matrix. Since all node IDs are distinct, they constitute a linearly independent matrix. Figure 2 illustrates the construction of the G matrix. Selecting G as *Vandermonde* matrix with node ID as seeds provide twofold benefits in our scheme: (1) each node now does not need to store the whole column of G matrix; instead knowing the ID (i.e., seed for that column), it can compute the whole column, (2) when a pair of nodes want to find a pair-wise key between them, they do not need to share the columns of the G matrix beforehand. This modification greatly reduces the communication overheads of our protocol.

3.1.2 Key Agreement Phase

In our scheme, we do not require a key agreement phase. When a node i want to send a secure message to node j , it will encrypt the message using key K_{ij} . To generate the key K_{ij} , node i will perform the following steps:

1. Generate the column $Col_j(G)$ of G matrix using node ID j as the seed.
2. Then compute $K_{ij} = Col_j(G) \cdot Row_i(A)$, where $Row_i(A)$ is the i th row of matrix A , in other words the secret key materials of node i , and $Col_j(G)$ is the j th column of matrix G .

When node j will receive encrypted message from node i , node j will perform the same steps and compute the key K_{ji} . Since $K_{ij} = K_{ji}$, node j can use K_{ji} to decrypt the message.

Fig. 2 Example of G matrix construction

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & N \\ (1)^2 & (2)^2 & (3)^2 & \dots & N^2 \\ (1)^3 & (2)^3 & (3)^3 & \dots & N^3 \\ & & & \vdots & \\ (1)^\lambda & (2)^\lambda & (3)^\lambda & \dots & N^\lambda \end{pmatrix}$$

However, without knowing the private information of node j , no other node can compute K_{ji} .

3.1.3 Key Revocation/Refresh

Key revocation and key refresh are very important parts of any key management scheme. Given an indefinite time, it is easy to compromise a network if it does not refresh the key periodically. In addition, to continue with normal network operation, the key needs to be revoked once a compromised node is identified. In our scheme, the base station can revoke the keys of the compromised nodes. We assume that with the help of intrusion detection systems, such as [22, 24], the compromised nodes can be detected and a list of compromised nodes, R is sent to the base station. Let us say, $R = (n_1, n_2, \dots, n_r)$ are the ID's of the compromised nodes in the network. The base station then initiates the key revocation process as follows:

1. It generates a new random $(\lambda + 1) \times (\lambda + 1)$ symmetric matrix \bar{D} over $GF(q)$, and computes an $N \times (\lambda + 1)$ matrix $\bar{A} = (\bar{D} \cdot G)^T$, where $(\bar{D} \cdot G)^T$ is the transpose of $(\bar{D} \cdot G)$.
2. It then generates a bit vector X of length N , where N is the number of nodes in network, such that $X[i] = 0$ if the node ID i is in R , 1 otherwise.
3. Then it computes the $N \times (\lambda + 1)$ matrix $H = \bar{A} + A$, and performs $Row_i(H) \cdot X[i]$, where $Row_i(H)$ are the elements of the i th row of H .
4. For each non-revoked node i , the base station computes:

$$M_i = i || Ek(K_{ii}, Row_i(H)) || MAC(K_{ii}, i || Ek(K_{ii}, Row_i(H))) \tag{2}$$

where $||$ is the message concatenation and K_{ii} is computed as $Col_i(G) \cdot Row_i(A)$. Ek is any suitable symmetric encryption algorithm for WSNs using K_{ii} as key.

5. Finally, the BS computes $M = M || M_i$, where $i \notin R$ and broadcasts $R || M$. The base station then updates D as \bar{D} for future revocation.
6. When a non-revoked node t receives the revocation broadcast, it extracts its own share, $t || Ek(K_{tt}, Row_t(H)) || MAC(K_{tt}, t || Ek(K_{tt}, Row_t(H)))$ from the broadcast message. It then computes:

$$V = MAC(K_{tt}, t || Ek(K_{tt}, Row_t(H))) \tag{3}$$

where K_{tt} is determined by the node's private share as: $Col_t(G) \cdot Row_t(A)$. If V matches with the received MAC , then it means the broadcast have come from the base station node. After this, the node decrypts the encrypted part, $Ek(K_{tt}, Row_t(H))$ using K_{tt} and updates its private share as $Row_t(A) = Row_t(H) - Row_t(A)$. Otherwise, it discards the broadcast message. It is easy to see that after the update is done, the non-revoked node stores $Row_t(\bar{A})$ whereas the revoked node will not be able to update its key share and therefore will be unable to take part in future communications with non-revoked nodes. Note that if an attacker captures the broadcast message of the base station and then replays it at a later time, the calculation of $V = MAC(K_{tt}, Row_t(H))$, will not match with the replayed MAC , since the node has already updated its share to a new authentic share by updating the K_{tt} value. As a result the node would reject any replayed broadcast and thus prevent replay attack.

The key refresh is a special case of key revocation with R being an empty list. If for any reason the broadcast message is corrupted or a non-revoked node cannot establish a key

with other node then that non-revoked node would send request to the base station node and the base station node would resend the share update for that node as a separate message.

3.1.4 New Node Addition

To add a new node, the base station pre-loads the seed value (i.e., node ID) and private secret key shares to the new nodes memory. For the sake of simplicity, let us assume that the new node ID is $N + 1$, and then the private secret shares for node $N + 1$ can be computed as follows:

1. Extend the matrix G by adding a new column using $N + 1$ as the seed.
2. for $i = 1$ to λ
 for $j = 1$ to λ
 $nr[i] = nr[i] + D[i][j] \cdot G[i][N + 1]$
 where nr is the private secret share of node $N + 1$.

An illustration of the new node addition process is shown in Fig. 3.

3.2 Dynamic Group Key Management

Many WSNs applications such as network and environmental monitoring, air quality monitoring, forest fire detection, etc., require that the data gathered at individual nodes can be pre-processed in the network, e.g., correlating related data, compressing, or filtering. These kinds of applications may require temporary group of nodes to cooperate with each other by forming a dynamic group. To facilitate such requirement our protocol can be used to form a dynamic secure group.

3.2.1 Group Key Generation

In our protocol, the base station is responsible for initiating a dynamic group creation. Let us assume that the BS wants to initiate a group G_m with the group members

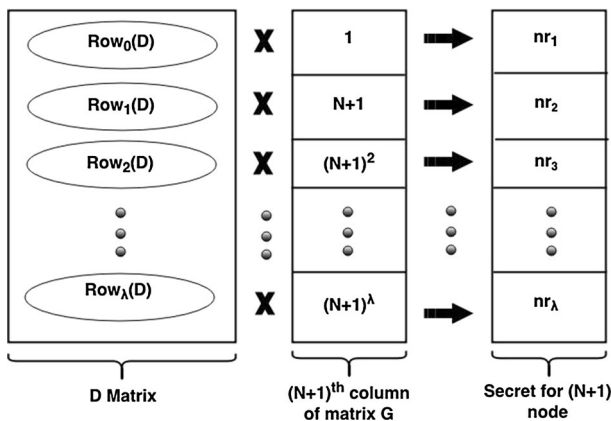


Fig. 3 New node addition mechanism

$\{n_1, n_2, \dots, n_m\}$, where $n_i: i = 1, \dots, m$, is the node ID. BS then performs the following steps to create and distribute the group key G_k for group G_m :

1. The BS randomly selects a group key: G_k .
2. Then for each member node n_i , it computes:

$$Y_i = n_i \parallel Ek(K_{ii}, G_k) \parallel TS \parallel MAC(K_{ii}, (n_i \parallel Ek(K_{ii}, G_k)) \parallel TS),$$
 where n_i is the ID of the i th group member node, K_{ii} is computed as $Col_i(G) \cdot Row_i(A)$, and TS is the current timestamp added to protect against reply attack.
3. The BS then broadcast $M_g = (n_1, Y_1) \parallel (n_2, Y_2) \parallel \dots \parallel (n_m, Y_m)$.
4. Receiving the broadcast message M_g , the group member node n_t , first extracts its share Y_t and verifies the integrity of the broadcast by computing $V = MAC(n_t \parallel Ek(K_{tt}, G_k) \parallel TS)$ and comparing it with the received MAC . Note that node n_t can compute K_{tt} using its private share as $Col_t(G) \cdot Row_t(A)$. If V matches with the received MAC , then the broadcast is authentic. Next, it checks the received timestamp value and accept the broadcast message if the timestamp is within a reasonable threshold value. The threshold value can be pre-determined or agreed upon by synchronizing the clock. For example BS can periodically broadcast the time on its clock together with a MAC . Once this step is passed, the node then decrypts the group key G_k using the key K_{tt} from the encrypted portion of Y_t .

3.2.2 Group Key Update

Any group key management scheme should satisfy the following basic security requirements, namely, *group confidentiality*: only group member nodes will be able to take part in the group communication, *forward secrecy*: ensuring that a group member node will not be able to derive any future session keys from its known key once it leaves the network, and *backward secrecy*: ensuring that if a new node joins the network it will not be able to derive the previous session keys from its current key. Therefore, group keys should be updated if a node's key is revoked or a new member in the group is added. In our protocol, a group key can be easily updated by selecting another new group key and broadcasting it to the group members.

4 Security Analysis

We proposed an integrated pairwise and group-wise key management protocol. In this section we present the security analysis of our scheme. We also present how our protocol can withstand some common security threats on WSNs.

4.1 Pairwise Key Distribution

Since we use the Blom's scheme as our central mechanism for pairwise key management, it follows from the Blom's scheme that our scheme is unconditionally secure as long as no more than λ nodes are compromised [2, 3]. In other words, since the pairwise key is different for each pair of nodes, the discovery of less than λ captured keys does not reveal any information about the keys of non-captured nodes and therefore the whole network is not compromised. To compromise the entire network an adversary need to capture λ nodes and get their private information from the memory. If λ is sufficiently large, then it is very

difficult and in some cases impossible for the adversary to capture all the λ nodes and collect their private information from the memory. Moreover, the key refresh mechanism, proposed in our scheme, puts a time bound on the adversary and makes the whole network more secure. This is due to the fact that after the key refresh phase, the pair-wise keys are refreshed to generate completely new keys. Thus, an adversary now has to capture λ nodes between two key refresh phases to breach the security of the entire network, which is difficult.

The key refresh or revocation process is also very secure because only the base station node is allowed to initiate the key refresh/revocation process. Note that the base station is assumed to be trusted and powerful in our system and only base station nodes possess the knowledge of all private keys of all the nodes in the network. Any broadcast message from base stations is encrypted with the current private key of each non-revoked node so that intruders cannot gain any knowledge about previous or new values of the private key share.

Apart from general security discussion, in the following sections we present how our protocol can withstand some specific security threats.

4.1.1 Spoofed or Reply Attacks

In this type of attack, an attacker sniffs the wireless communication channel to capture data and at a later point of time replays the captured message to gain unauthorized access or make the system de-synchronized. In our scheme, only knowing each other's ID and without any message exchange, communicating parties can generate a secret symmetric key. Therefore, an attacker cannot capture and replay messages. However, in the key refresh phase, the base station node sends key update materials to all the nodes in the network. A malicious attacker may capture this update message and replay it later to launch a de-synchronization attack. To prevent such an attack, note that the base station nodes, compute the message digest of the new share update data using the current private key of each node and attach that digest along with the encrypted new share update data. Upon receiving the broadcast message, every node decrypts its own private share update data and then checks the integrity by computing the message digest of that data using its current private key. If the computed message digest does not match with the received message digest then that node discards the broadcast as rouge and hence prevents replay attack.

In the case of group creation, we prevent reply attack by including different timestamps with different group creation broadcasts. Both the timestamp and the broadcast are authenticated by computing a MAC using the members private keys, which makes it immune from unauthorized modifications. Before accepting any group creation broadcast, the member node verifies that the timestamp is within a predetermined threshold or not. If the threshold value is small (few hundred milliseconds), it is very difficult for an attacker to copy and reply the message within the short time span.

4.1.2 De-synchronization Attacks

A de-synchronization attack is launched with an intention to make communicating parties reach at a different state in their protocol, where they cannot find common credentials for the communication. In the proposed system, an attacker may prevent key refresh/update broadcast message to be reached to a particular node. As a result, all nodes will update their key materials, except that particular node. Consequently, the other nodes will not be

able to establish a shared key with it. This would lead to a de-synchronized state in the system.

To overcome such a situation, when a non-revoked node is unable to establish a shared key with other non-revoked node, it sends a request to the BS node to resend the key update/refresh materials for both nodes. For instance, assume node i wants to send an encrypted message to node j . Node i uses its currently held key K_{ij} for the encryption and computes *MAC* of the message. Then, it sends the encrypted message and the *MAC* to node j . When node j receives this message, it first decrypts the message and computes *MAC* of the received message. To verify the integrity of the message, node j then compares the computed *MAC* with the received *MAC*. If they do not match, then there might be two cases; (1) the key materials used for computing K_{ij} or K_{ji} are different on either side, or (2) some intruder may have modified the message in transit. In both cases, the receiving node, (i.e., node j) sends a request to the BS node to resend the key material update/refresh for both nodes.

Upon receiving such request, the BS node, first keeps a record of the request and queries both nodes for their current key material values. Once the current key material values are identified, BS node resends the key update/refresh materials to both nodes separately. If the BS node receives two such requests within a certain period from the same set of nodes, then it initiates intrusion detection system for that part of the network to determine if the network is under attack. Thus, even though we cannot completely prevent the de-synchronization attacks, we can detect and recover from it.

4.1.3 Denial-of-Service Attacks

Denial-of-Service (DoS) attack is very difficult to prevent in any system. It is done with the intention of disrupting the availability of legitimate services. Many DoS attacks require man-in-the-middle (MiM), where an intruder can actively capture and modify/drop the data packets. In our scheme, all the broadcast and unicast message transfers are sent encrypted and a *MAC* is attached to identify any modifications. If an attacker deliberately changes the broadcast message to something meaningless with an intention to launch a denial-of-service attack, then our system can overcome this situation after detecting a mismatched message digest or being unable to establish a key with other non-revoked node. In that case, it can request the base station to resend the share update. If multiple phenomena are detected, where integrity check fails, nodes can request BS node to trigger intrusion detection system to identify any potential threats.

4.1.4 Physical Node Capture Attacks

As WSNs are deployed in unattended mode, capturing and reading the memory contents of a node is very easy for an attacker. Without any physical protection on the nodes, this kind of attack is challenging to prevent. However, our scheme provides better resilience to node capture attack, since we only store partial key materials in the memory of a node. If an intruder captures and reads the memory contents of a node, it can only compromise messages that are transferred between that node with other nodes. However, the attacker will not be able to compromise the communication of any other set of nodes. To compromise the entire network, the attacker needs to capture and read the memory of λ nodes within a short period of time, which makes it hard in the context of WSNs.

4.1.5 Forward and Backward Secrecy

In the case of group creation, the group key is randomly chosen. Any non-group member node cannot decrypt the group key from the group creation broadcast message, since the pairwise key that is used for encrypting the group key would not match with the non-member's key. Therefore, non-group member node cannot take part in the group communication. In addition, the pairwise key between any two nodes is different; hence the non-group member nodes have no information about the pair-wise keys of group member nodes.

Our scheme provides perfect forward and backward secrecy. On the one hand, the group key, G_k , which is unique and random, is updated once a node is added or deleted to or from the group. As a result, the knowledge of the previous key does not help in deriving the new group key and hence a deleted node cannot decrypt any future group communications. On the other hand, a newly added node cannot derive previous group keys from the new key since there are no practical relationships among them. Therefore, a newly added node cannot decrypt any previous group communications.

4.1.6 Black Hole and Worm Hole Attacks

In a black hole attack, an intruder captures some nodes and reprograms them not to forward any packet to other nodes. Thus any packets that passes through the captured nodes are not delivered to the destination node, creating a black hole region in the network. The worm hole attack is also similar to black hole attack. In this attack, an attacker receives packets at one point in the network, *tunnels* them to another point in the network, and then delivers the received packets into the network from that point. Both these attacks are easy to launch, but difficult to prevent. Essentially these attacks are another from of DoS attack effective on network layer, that can affect routing protocols, aggregation of data and location based services.

Our scheme provides a key management service that can be used in different layers of the protocol stack including network layer to help prevent black hole or worm hole attacks. For example, many well known techniques for preventing black hole and worm hole attacks in WSN such as watchdog and pathrater [18], packet leash [10], multipath hop-count analysis [13], time-base mechanism [7] etc. can use our scheme to provide a secure authenticated communication channel between neighbor nodes. This in turn will strengthen these protocols to provide better protection against black hole and worm hole attacks.

5 Performance Analysis

In this section we present the performance analysis of our scheme in terms of memory usage, communication cost, energy and computational overhead. We implemented our scheme on TelosB motes, the latest research oriented mote developed by UC Berkeley [26]. TelosB has MSP430 microcontroller which incorporates an 8 MHz, 16-bit RISC CPU, 48K bytes flash memory (ROM) and 10K RAM. The RF transceiver on TelosB is IEEE 802.15.4/ZigBee compliant, and can have upto 250 kbps data rate. We have used NesC programming language to implement our protocol on TinyOS [14]. The key size $GF(q)$ is chosen as 64-bits, where q is a large prime number that require 64-bits to store. The 64-bits key size was chosen because TinySec [12], which is a fully implemented

protocol for link-layer cryptography in sensor networks, currently uses a 64-bit key. The results were obtained for various cases of λ . To improve the accuracy of our results, we repeated the simulations at least 50 times. Table 1 shows a summary of the overheads of our proposed scheme.

5.1 Parameter Setting

Our proposed scheme has a setup tasks that BS node performs at the network setup phase. To simplify the experiments, we have simulated the setup phase and preloaded the private key shares in the code, so that we can use more RAM for other variables. In our experiments, we kept the code to the bare minimum, key computation and turning on the LEDs at the start and end of the process. When the program runs, it lights up an LED, computes the pairwise key, and lights up another LED on completion.

5.2 Memory Overhead

The most memory overhead in our proposed protocol comes from the fact that every node need to store $(\lambda + 1)$ 64-bits numbers as pre-loaded secret. With $\lambda = 100$, each sensor node needs to store $(101 \times 8 \text{ bytes}) = 808$ bytes of memory which is less than 10

5.3 Energy Overhead

We also estimated the energy consumed in the key computation process. In general, the energy consumption E can be calculated by

$$E = U.I.t \quad (4)$$

Here U is the voltage, I is the current drawn during an operation and t is the time duration of an operation. Several energy models for measuring energy consumptions on WSNs are proposed in literature [9, 20, 21, 25]. All of these models use the modular approach, which requires the decomposition of power consumption into elementary factors contributing to the overall energy consumption. It is relatively easy to find the current drawn by the complete node on a commercially available nodes, however, measuring the power of individual components is generally difficult and often requires special experimental platform [1]. The authors in [21], conducted a module-based energy measurement on a TelosB platform and extended the current consumption chart of the individual components present

Table 1 Overheads on our scheme

Overhead	Memory	Communication	Computation
Pair-wise key generation	$(\lambda + 1) \cdot q$	Zero	Evaluation of λ -degree polynomial
Key revocation/refresh	$(\lambda + 1) \cdot q$	One broadcast, size: $N \cdot (\lambda + 1) \cdot q$	$1H, (\lambda + 1) \cdot M$
New node addition	$(\lambda + 1) \cdot q$	Zero	Zero
Group key generation	q	One broadcast, size: $(GS \cdot q) + c$	$1H, 1\text{decryption}$

M Multiplication, H Hash computation, c Constant value, GS Group size, q size of key in bits

in a TelosB mote. To estimate the energy consumption of our protocol we used the energy consumed by different individual components on TelosB from [21].

TelosB motes are powered by two AA batteries, so U is approximately equal to 3 V. The current-supply value (I) varies in different operations as shown in Table 2 abstracted from [26]. In a TelosB mote, apart from the full active mode, the micro controller can be configured to operate in five different states [21]. However, the authors in [21] show that, the highest current is drawn when the micro controller is in the full active mode. In addition, the rest of the states result in current draw reduction that could provide only marginal difference in a typical WSN application. Therefore, in our calculations, we have considered the upper bound, which is the full active mode for the micro controller with 4 MHz CPU clock cycle. To calculate the energy of different phases of our protocol, first we compute the processing time of each phase, then use the above equation to calculate energy consumption. If no data transmission is required, then we use the current draw metric as normal “*Micro Controller Unit (MCU) on, Radio off*” mode of data processing with 4 MHz CPU clock. For example, to compute a 64-bit 32-degree polynomial on a TelosB mote, with the “*MCU on, Radio off*” mode, it takes about 16.8 ms, therefore, the power consumption for this operation is calculated as: $E = 3 \times 1.8 \times 16.8 = 0.09$ mJ. As the table 2 shows that the most energy on a sensor mote is consumed during the data transmission phase and our protocol has very minimal data communication, therefore, the power consumption of our protocol is also minimal.

5.4 Communication Overhead

As discussed earlier, the radio transceiver related power consumption is the most critical component of the overall system-wide power consumption on a mote. More specifically receiving a frame consumes more energy than sending a frame. Considering that, assuming the motes are expected to transmit/receive at full power, for which current demands are, approximately the same, only two different current draw thresholds are actually distinguished in Table 2. Based on these thresholds, the power consumption contributed by the radio operation can be calculated considering the time duration percentages regarding the radio “Transmitting” and “Receiving” states with MCU on.

Our protocol is very efficient in terms of communication overhead, because nodes need not exchange any data for the key agreement phase which take place prior to actual data transfer. For example, in the case of pairwise key generation, a node does not need to go through any key agreement phase, hence no message exchange is required. Instead, a sender node can generate the key for the receiver node by knowing only the receiving node’s ID. For the key refresh/revocation and group initiation, it only requires one broadcast message to be sent. Besides, the new node addition process does not require any communication at all. Table 1 shows the estimation for the communication overhead of our protocol.

Our protocol supports different communication architectures including multi-hop flat and hierarchical architectures. In the case of a flat multi-hop communication, any two

Table 2 Current drawn for different operations on TelosB

Operation	Normal (mA)	Max (mA)
MCU on, radio off	1.8	2.4
MCU on, radio receiving	21.8	23
MCU on, radio transmitting	19.5	21

neighbouring nodes can create a secure communication path with each other without prior communications. Similarly, in the case of hierarchical architecture, any node can create a secure communication with the cluster head node. In addition, our protocol also supports group communication, in case a cluster of nodes want to communicate with each other.

5.5 Computational Overhead

Each node in our protocol only needs to perform a polynomial evaluation of $(\lambda + 1)$ terms, which involves $(\lambda + 1)$ multiplications and additions. In general, we are interested in measuring the cost of the λ -degree polynomial computation in sensor mote. In our implementation, we have chosen q to be 64-bits. Therefore, $\lambda - (64 - bit \times 64 - bit)$ modular multiplications are required to compute the polynomial. On the 16-bit CPU of TelosB, each $64 - bit \times 64 - bit$ multiplication costs 16-word multiplications. To further reduce the computational cost, we adopted a simplification by selecting the IDs of the node as 16-bit integer. Therefore, the modular multiplication in polynomial is always performed between a 64-bit integer and 16-bit integer. As the result, the cost of multiplication is reduced by four times. Table 3 shows the summary of the computational overheads for the pairwise key generation process. In key refresh phase, the BS broadcasts a list with updated key materials to all nodes in the network. Table 4 provides a summary outlining the individual operations that nodes have to do at this stage and their associated computational and energy consumption costs.

6 Discussions and Comparisons

In this section, we present a brief comparison of our scheme with other existing schemes in some performance metrics important in WSNs, i.e., computational time, energy, memory and communication, as shown in the following graph (Fig. 4), in addition a summary is also presented in Table 5.

Since the public key cryptography provides mechanisms for key agreements and authentications, we compare our schemes with both public key and symmetric key cryptosystems. In the public key cryptography, require two parties to exchange their public keys from which to derive a common secret. To authenticate each other, public keys need to be signed by a trusted authority and verified by both parties. Liu et al. [15] implemented TinyECC in MICAz mote that had both key computation and authentication using ECDH and ECDSA. With all optimizations enabled they found that the execution times were: ECDSA initialization 3393 ms, verification 2436 ms, and for ECDH initialization 1839 ms, and key computation 2117 ms. Hence, key computation and signature verification would take 4.5 s after initialization taking about 5.2 s. In another study, Wang et al.

Table 3 Computational overheads of λ -degree polynomial evaluation in TelosB sensor node

λ value	32	64	128
Memory (bytes)	282	538	1050
Time (ms)	16.8	34.94	74.07
Power (mJ)	0.09	0.18	0.40

Table 4 Key refresh/update with $\lambda = 128$

Operations	Time (ms)	Energy (mJ)
Step 1: One λ -degree polynomial evaluation	74.07	399.98
Step 2: One MAC verification [12]	4.66	25.17
Step 3: One TinySec-AE decryption [12]	0.38	2.054
Step 4: (λ) 64-bit subtractions	0.128	0.69
Total	79.23	427.90

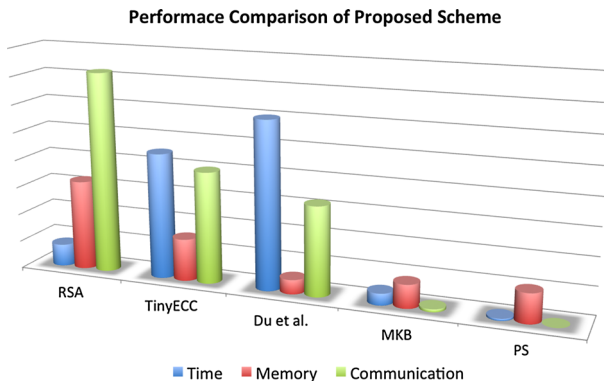


Fig. 4 Comparison of different schemes with the proposed scheme

Table 5 Comparisons of different schemes

	Method	Time (s)	Memory (kB)	Data exchange (bits)
RSA [28]	RSA-1024	0.79	3.224	1152
TinyECC [15]	ECC-160	4.5	1.51	640
Du et al. [6]	Blom	6	0.521	512
MKB scheme [30]	Blom	0.417	0.838	16
Our proposed scheme	Blom	0.074	1.05	0

[28] compared the computation time of RSA and ECDSA signature verification. Using RSA-1024 bits signatures; it is shown to be 0.79 s compared to 2.85 for ECDSA signature verification [28]. The memory requirements for TinyECC, with all optimizations enabled were 1510 bytes and 1774 bytes for ECDSA and ECDH respectively. Similarly, the RSA scheme required 3224 bytes of memory [28].

In contrast to PKC, Du et al. [6] scheme and MKB [30, 31] schemes are based on symmetric key cryptosystems using the Blom’s scheme. In [4], they implemented the Du’s scheme in MicaZ mote and found that with $\lambda = 20$ and high probability of connectivity, it took about 6 s and 521 bytes of memory to compute a pairwise key. Since Du’s scheme uses multiple-space key distribution, prior to establishing a pairwise key, it needs to

broadcast a message of 512 bits containing IDs and spaces to figure out if two neighbouring nodes share a common space. In MKB scheme [30], for the $\lambda = 128$, the key agreement time and average memory usage were 417 ms and 838 bytes respectively. In addition, they have 16-bit data exchange prior to key agreement phase.

Figure 4 clearly shows that our scheme is more efficient among all the above mentioned schemes. Although our scheme uses slightly more memory than Du et al. and MKB schemes, however, in terms of computation and communication overheads, our scheme performs better than these schemes. In our protocol, the computational time depends on the value of λ (see Table 3), it took around 74 ms with the value of $\lambda = 128$. Since the energy cost is directly related to the execution time and number of communications, it is obvious that our scheme outperforms all the above discussed schemes with respect to computation, communication and energy consumption overheads.

None of the above mentioned schemes discuss mechanisms for key refresh/update, new node addition techniques and group creation mechanisms; therefore we could not compare the performance of our scheme with them in terms of key update/refresh and new node addition techniques and group creation.

7 Conclusion

We have presented a lightweight, robust protocol for both pairwise and group key management for WSNs. We optimized the Blom's scheme for wireless sensor networks, while retaining the advantages of the original scheme. It is able to achieve efficient and secure pairwise and group key generation and requires little energy and computational resources. We implemented our protocol in TelosB mote and the results are impressive in terms of speed, energy, communication overhead and memory usage. We have also compared the performance of our scheme with other schemes for WSNs.

References

1. Agarwal, R., Martinez-Catala, R. V., Harte, S., Segard, C., & O'Flynn, B. (2008). Modeling power in multi-functionality sensor network applications. In *Second international conference on sensor technologies and applications, 2008. SENSORCOMM '08* (pp. 507–512).
2. Blom, R. (1985). An optimal class of symmetric key generation systems. In *Proceedings of the EUROCRYPT 84 workshop on advances in cryptology: Theory and application of cryptographic techniques* (pp. 335–338). New York, NY, USA: Springer-Verlag New York Inc.
3. Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., & Yung, M. (1993). Perfectly-secure key distribution for dynamic conferences. In *CRYPTO '92: Proceedings of the 12th annual international cryptology conference on advances in cryptology* (pp. 471–486). London, UK: Springer-Verlag.
4. Boujelben, M., Youssef, H., & Abid, M. (2008). An efficient scheme for key pre-distribution in wireless sensor networks. In *IEEE international conference on wireless and mobile computing, networking and communication* (pp. 532–537).
5. Chan, H., Perrig, A., & Song, D. (2003). Random key predistribution schemes for sensor networks. In *Proceedings of 2003 symposium on security and privacy* (pp. 197–213, 11–14).
6. Du, W., Deng, J., Han, Y. S., Chen, S., & Varshney, P. K. (2004). A key management scheme for wireless sensor networks using deployment knowledge. In *INFOCOM 2004. Twenty-third Annual Joint conference of the IEEE computer and communications societies* (Vol. 1, pp. -597).
7. Eriksson, J., Krishnamurthy, S. V., & Faloutsos, M. (2006). Truelink: A practical countermeasure to the wormhole attack in wireless networks. In *Proceedings of the 2006 14th IEEE international conference on network protocols, 2006. ICNP '06* (pp. 75–84).

8. Eschenauer, L., & Gligor, V. D. (2002). A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on computer and communications security* (pp. 41–47), New York, NY, USA, ACM.
9. Heinzelman, W. R., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd annual Hawaii international conference on system sciences*, 2000 (Vol. 2, p. 10).
10. Hu, Y.-C., Perrig, A., & Johnson, D. B. (2003). Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM 2003. Twenty-second annual joint conference of the IEEE computer and communications. IEEE Societies* (Vol. 3, pp. 1976–1986).
11. Hussain, S., Kausar, F., & Masood, A. (2007). An efficient key distribution scheme for heterogeneous sensor networks. In *Proceedings of the 2007 international conference on wireless communications and mobile computing, IWCMC '07*, (pp. 388–392), New York, NY, USA, 2007. ACM.
12. Karlof, C., Sastry, N., & Wagner, D. (2004). Tinysec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on embedded networked sensor systems, SenSys '04*, (pp. 162–175), New York, NY, USA, 2004. ACM.
13. Karlsson, J., Dooley, L. S., & Pulkkis, G. (2011). A new manet wormhole detection algorithm based on traversal time and hop count analysis. *Sensors*, *11*(12), 11122–11140.
14. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Woo, A., Gay, D., et al. (2004). Tinyos: An operating system for sensor networks. In *in ambient intelligence*. Berlin: Springer.
15. Liu, A., & Ning, P. (2008). Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *International conference on information processing in sensor networks, 2008. IPSN '08* (pp. 245–256).
16. Liu, D., & Ning, P. (2003). Location-based pairwise key establishments for static sensor networks. In *Proceedings of the 1st ACM workshop on security of ad hoc and sensor networks, SASN '03* (pp. 72–82). New York, NY, USA, 2003. ACM.
17. MacWilliams, F. J., & Sloane, N. J. A. (1988). *The theory of error-correcting codes* (North-Holland Mathematical Library). North Holland Publishing Company.
18. Marti, S., Giuli, T. J., Lai, K., & Baker, M. (2000). Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on mobile computing and networking, MobiCom '00* (pp. 255–265). New York, NY, USA, 2000. ACM.
19. Panja, B., Madria, S. K., & Bhargava, B. (2006). Energy and communication efficient group key management protocol for hierarchical sensor networks. In *IEEE international conference on sensor networks, ubiquitous, and trustworthy computing, 2006* (Vol. 1, p. 8).
20. Polastre, J., Hill, J., & Culler, D. (2004). Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on embedded networked sensor systems, SenSys '04* (pp. 95–107). New York, NY, USA, 2004. ACM.
21. Prayati, A., Antonopoulos, Ch., Stoyanova, T., Koulamas, C., & Papadopoulos, G. (2010). A modeling approach on the telosb wsn platform power consumption. *Journal of Systems and Software*, *83*(8), 1355–1363.
22. Puttini, Ricardo, Hanashiro, Mara, Miziara, Fbio, de Sousa, Rafael, Javier Garca-Villalba, L., & Barenco, C. J. (2006). On the anomaly intrusion-detection in mobile ad hoc network environments. In Pedro Cuenca & Luiz Orozco-Barbosa (Eds.), *Personal wireless communications* (Vol. 4217, pp. 182–193), Lecture notes in computer science Berlin: Springer.
23. Riaz, R., Naureen, A., Akram, A., Akbar, A. H., Kim, K.-H., & Ahmed, H. F. (2008). A unified security framework with three key management schemes for wireless sensor networks. *Computer Communications*, *31*(18), 4269–4280. (Secure multi-mode systems and their applications for pervasive computing).
24. Sen, S., & Clark, J. A. (2011). Evolutionary computation techniques for intrusion detection in mobile ad hoc networks. *Computer Networks*, *55*(15), 3441–3457.
25. Shih, E., Cho, S.-H., Ickes, N., Min, R., Sinha, A., Wang, A., & Chandrakasan, A. (2001). Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *Proceedings of the 7th annual international conference on mobile computing and networking, MobiCom '01* (pp. 272–287). New York, NY, USA, ACM.
26. TelosB data sheet. (2014). Crossbow technology Inc. Accessed September 20, 2014, <http://www.xbow.com/>
27. Traynor, P., Kumar, R., Bin Saad, H., Cao, G., & La Porta, T. (2006). Liger: Implementing efficient hybrid security mechanisms for heterogeneous sensor networks. In P. Gunningberg, L. Larzon, M. Satyanarayanan & N. Davies (Eds), *ACM/USENIX fourth international conference on mobile systems applications and services (MobiSys)* (pp. 15–27). ACM press.

28. Wang, H., & Li, Q. (2006). Efficient implementation of public key cryptosystems on mote sensors (short paper). In *Proceedings of the 8th international conference on information and communications security, ICICS'06* (pp. 519–528). Berlin, Heidelberg, Springer.
29. Wen, M., Zheng, Y.-F., Ye, W.-J., Chen, K.-F., & Qiu, W.-D. (2009). A key management protocol with robust continuity for sensor networks. *Computer Standards and Interfaces*, 31(4), 642–647.
30. Yang, M. L., Al-Anbuky, A., & Liu, W. (2012). A fast and efficient key agreement scheme for wireless sensor networks. In *ICWMC 2012, The eighth international conference on wireless and mobile communications* (pp. 231–237). Venice: ThinkMind.
31. Al-Anbuky, A., & Liu, W. (2014). An authenticated key agreement scheme for wireless sensor networks. *Journal of Sensor and Actuator Networks*, 3(3), 181–206.
32. Younis, M. F., Ghumman, K., & Eltoweissy, M. (2006). Location-aware combinatorial key management scheme for clustered sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(18), 865–882.
33. Zhu, S., Setia, S., & Jajodia, S. (2003). Leap: efficient security mechanisms for large-scale distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on computer and communications security* (pp. 62–72). New York, NY, USA, ACM.
34. Zhu, S., Xu, S., Setia, S., & Jajodia, S. (2003). Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In *Proceedings of the 11th IEEE international conference on network protocols, ICNP '03* (p. 326) Washington, DC, USA, IEEE Computer Society.



Musfiq Rahman is an Assistant Professor in the Department of Computing Science, Thompson Rivers University. His main research area is in security and key management protocols for wireless sensor networks. He has conducted many research projects in the areas of RFID, wireless ad hoc networks, smart phone app development, and NFC as a lead researcher.



Srinivas Sampalli is a Professor and 3M National Teaching Fellow in the Faculty of Computer Science, Dalhousie University. His research is mainly in the area of wireless security and applications. Specifically, he has investigated protocol vulnerabilities, security best practices, risk mitigation and analysis, design of intrusion detection and prevention systems, and applications of RFID systems and NFC-enabled smartphones.