Simulation of Distributed Key Management Protocols in Clustered Wireless Sensor Networks

Jibi Abraham and K S Ramanatha
Department of Computer Science & Engg.
M S Ramaiah Institute of Technology
MSRIT Post, Bangalore - 560 054, Karnataka, India
Email: jibiabraham@msrit.edu, ksramanatha@msrit.edu

Abstract

In Wireless Sensor Network security has to be mainly provided to two types of communications: one-to-all and one-to-one. This paper proposes a complete set of low complexity protocols to initially generate and distribute two types of secret keys, periodically renew the keys and change the keys based on cluster dynamics. They are designed for a clustered hierarchical network suitable for data aggregation and all the communications in the protocols are properly authenticated using Elliptic Curve Digital Signature scheme. The security analysis shows that the protocols are strong against several possible attacks. The protocols are implemented for TinyOS using NesC, simulated under TOSSIM and are viable for implementation in resource-constrained platforms like MICA. A major outcome of our simulation is the observation that most of the time rekeying is done because of cluster dynamics. It is therefore suggested that implementing proper authentication protocols and batch rekeying protocols reduce the number of times rekeying is performed, resulting in longer life of the network.

1. Introduction

Wireless Sensor network (*WSN*) comprises of large number of sensor nodes distributed over an area to be monitored. These sensor nodes are low cost devices and have very constrained resources. With their restricted computation and communication facilities, they perform collaborative processing between themselves to gather information and disseminate the task. There is a wide range of applications for sensor networks like surveillance applications to analyze the motion of tornado or fire detection in a forest. An observer can disseminate a query to collect data from the network. Sensor nodes detect the phenomenon and the measurements are routed to a Base Station, which analyzes them and gives the formatted result to the observer.

When we have to provide security to the data communications in sensor networks, we have to consider two types of application data exchanges: *one-to-all* communications and *one-to-one* communications. In ***One-to-all*** communication, the Base Station sends a query that will reach all the nodes in the network. When a node detects a phenomenon, the observation can be routed to the Base Station using ***one-to-one*** communication between neighboring sensor nodes. The security system should have key management protocols to take care of establishing and renewing keys for these two types of communication whenever required.

Pair-wise keys between Base Station and source sensor nodes can be used for providing security to *one-to-all* communication. Then the Base Station has to perform as many encryptions as the number of source nodes in the network and every source node has to perform one decryption. Instead if the Base Station has a network-wide key shared with all the nodes in the network, a query message needs to be encrypted only once. In such a system immediately after the self-configuration of the network [15], an efficient protocol can generate and distribute the network-wide key to all the nodes in the network.

End-to-end and *hop-by-hop* are the two methods to provide security to the *one-to-one* communications. In *end-to-end* security, the data get encrypted at the source node and decrypted at the Base Station. Since sensor nodes are resource constrained, the data transfer protocols are to be efficient enough to reduce the communication in the network. Because there are chances of spatial correlation in the data sensed by neighboring nodes, applying aggregation techniques [3] to the data at the intermediates nodes on the data traversal path can reduce the number of bytes transmitted through the network. This requires formation of clusters in the network, selection of a Cluster Head for each cluster and proper schemes to provide *hop-by-hop* security to the data. In the *hop-by-hop* security, the data can be encrypted at the cluster member using the pair-wise key shared between the member and the Cluster Head and decrypted at the Cluster Head. But to perform decryptions at the Cluster Head, it has to search and find out the appropriate pair-wise key for the respective sensor node, which is both time consuming and computation intensive. The decryption effort can be reduced if every cluster head shares a Cluster Key with all of its members. So to secure *one-to-one* communications efficiently, the system requires protocols for generation and distribution of Cluster Keys on the aggregation tree path.

There are two reasons for changing a cryptographic key to new one. If a given key is used for a long duration, a cryptanalyst can collect all the encrypted data, analyze them and break the key. According to [1], if the encryption key used is of length k bits then the key has to be changed after $2^{2k/3}$ number of encryptions. The ***Rekey Interval*** can be defined as the period with which encryption key of a node is changed to a new one. If the network topology does not change within the rekey interval, the new key can be easily distributed to all the nodes by encrypting it with the existing key. Each node decrypts the new key using the existing key and designates the new key as its current key. This way of rekeying is called ***Periodic Rekeying***.

The second reason to change a key in usage is because of the network topology change. This change happens within the rekey interval due to joining of new nodes or leaving of existing nodes. The rekeying method applied at this time is called ***Rekeying with Cluster Dynamics***. Normally, when a new node wishes to join a group, it is necessary to update the current group key to a new key to maintain ***Backward Secrecy*** to prevent the new node with the current key from going backwards in time to decipher previous contents encrypted with prior keys. Likewise, when a node leaves, it is necessary to update the key to maintain ***Forward Secrecy*** to prevent the node that left the group from using an old key to continue decrypting new contents.

The remainder of the paper is organized as follows: Section 2 provides the details about network architecture considered to implement the key management protocols. The set of key management protocols for the clustered sensor network is presented in Section 3. We analyze the security of the protocols in Section 4 and then we show the implementation details and results in Section 5. The related works are added to Section 6. In section 7, we address the simulation observations.

2. Sensor Network architecture

The network architecture we consider is a static sensor network in which thousands of nodes are distributed in an application area and the sensor nodes do not move from their deployment position at any later time. All the nodes in the network except the Base Station are with equal resources and the Base Station is with high-end computing and communication facilities. A typical sensor can be *MICA* sensor [22] with *TinyOS* [16] operating system, 4K *RAM*, 128K flash memory, 8bit 4MHz Atmel Atmega processor and 2 *AA* batteries.

The multi-hop protocol [6], part of *TinyOS* helps the network to self-organize and forms the network topology as a hierarchical tree. Hybrid, Energy-Efficient, Distributed clustering algorithm known as *HEED* [5] is used in our implementations to group the network into clusters. The Cluster Heads are uniformly distributed in the network and they are elected based on the clustering function, whose value is decided by residual energy and communication cost values among the neighborhood. After the formation of clusters and election of a Cluster Head in each cluster, each member finds out its Cluster Head with the highest clustering function value and sends the application specific data to the Base Station through its Cluster Head.

Residual energy of all nodes gets reduced upon performing the computations and communications. So cluster groups and memberships need to be changed at regular clustering intervals based on the value of the clustering function. At every clustering interval, a node continues with the same role of being a Cluster Head or a cluster member or change the role from Cluster Head to member or vice versa. This re-clustering helps to avoid the chances of Cluster Heads' battery charges getting drained quickly so that all the nodes in the network have almost same lifetime. If any node changes its role or a node chooses another Cluster Head, then it is a case of cluster dynamics.

3. Proposed Key Management Protocols

The key management protocols for the *one-to-all* communication require formation of a network wide key shared by all members of the network. The key management for *one-to-one* communication facilitating aggregation requires formation of pair-wise keys shared between every Cluster Head and cluster members and a Cluster Key shared by all members in a cluster. We assume a static sensor network with one or more clusters. Each cluster is having a Cluster Head decided by the *HEED* algorithm. Initially each Cluster Head establishes a pair-wise key with each of its cluster members using the *Pair-wise Key Distribution Protocol*. Then each Cluster Head generates and distributes the initial Cluster Key by using *Initial Cluster Key Distribution protocol*. Once all the Cluster Heads complete the Cluster Key distribution, the Base Station initiates the *Initial Network Key Distribution protocol*. These key management protocols are explained in detail in the coming sections. The notations used in our protocols are given in Table 1.

3.1 Initial Key Distribution Protocols

This includes generation and distribution of the pair-wise key, the initial Network Key and the initial Cluster Key.

3.1.1 Pair-wise Key Distribution Protocol: The Cluster Head has to share a pair-wise key with each of its cluster members. To establish a pair-wise key between the Cluster Head and a member, we have used the method proposed in our earlier work [8]. During the initial cluster formation phase of *HEED*, each sensor node authenticates to the Base Station and gets assigned a Cluster Head. The Base Station also helps to establish a pair-wise key to each sensor node and another pair-wise key between each sensor node and its Cluster Head. The authentication, Cluster Head assignment and pair-wise keys formation are done in a single step with very low channel usage, less memory requirement and usage of only symmetric cryptographic primitives. These pair-wise keys are useful for distribution of Cluster Keys and also during rekeying.

Table 1. Notations used in the Protocols

Notation	Description
BS	Base Station
C_i	i^{th} Cluster Head
S_j	j^{th} sensor node
CK_i	Cluster key generated at i^{th} Cluster Head
NK	Network key generated at the BS
K_{AB}	Pair-wise key shared between A and B
$E_K(M)$	Symmetric encryption of message M by using the key K
$Child(A, B)$	True if B is a child of A
$Genpubkey()$	Generates the private key and public key for Elliptic curve Cryptosystem
Pr_A	Private key of A
Pu_A	Public key of A
$DSAG(\mathrm{M}, Pr_A)$	Digital signature generated for message M using Private key of A
$DSAV(M, Pu_A)$	True if signature is verified for M using Public key of A
$Genkey()$	Generates a random pair-wise key
$A \parallel B$	A is concatenated to B

3.1.2 Initial Cluster Key distribution protocol: When the network is set up and organized into clusters, each Cluster Head C_i generates a Cluster Key CK_{Ci}. The initial Cluster Key distribution starts with the Base Station because it is also a Cluster Head.

$$CK_{Ci} = Genkey\ ()$$

The Cluster Head encrypts the key with each member's pair-wise key and sends to each member S_k. If public key mechanism is used for authenticating the rekeying messages, the Cluster Head generates the public key-private key pair so that this public key can also be distributed along with the initial Cluster Key.

```
$(Pr_{Ci}, Pu_{Ci}) = Genpubkey\ ()$
$\forall S_k$, if (Child $(C_i, S_k)$) then
    $C_i \rightarrow S_k$: $E_{KCiSk}(CK_{Ci} \parallel Pu_{Ci})$
```

Each member S_k receiving the message from its Cluster Head decrypts the message and stores the keys. If this node in turn is a Cluster Head to another cluster, the same procedure is repeated in a recursive manner.

At any later time, the Network Key may need to be changed either because of periodic rekeying or due to cluster dynamics. If it is because of cluster dynamics, that Cluster Head is required to send an authenticated request for network rekeying to the Base Station. We prefer to use the Elliptic curve digital signatures for validating the authenticity of the request. So every Cluster Head sends its public key to the Base Station encrypted with the pair-wise key shared with the Base Station.

$C_i \rightarrow BS$: E_{KBsCi} (Pu_{Ci})

The Base Station receiving the public key of the Cluster Head saves the key appropriately.

3.1.3 Initial Network Key distribution protocol: As the entire network is divided into many clusters, our protocol utilizes the Cluster Keys to efficiently distribute the Network Key. The protocol begins by generation of the Network Key by the Base Station.

$NK = Genkey$ ()

The public key-private key of the Base Station is already generated during the initial Cluster Key generation phase, because Base Station is also a Cluster Head. In each cluster in the network, the digital signature $DSAG$ (NK, Pr_{Ci}) is calculated; the Network Key and Base Station's public key are encrypted by using that cluster's Cluster Key CK_{Ci} and are sent to its members.

```
∀$S_k$, if (Child ($C_i$, $S_k$)) then
        $C_i \rightarrow S_k$: $E_{CKCi}$ ($NK \| Pu_{Bs}$) || $DSAG$ ($NK$, $Pr_{Ci}$)
```

Each member decrypts the message, verifies the digital signature by evaluating $DSAV$ (NK, Pu_{Ci}) and if success, stores the keys NK and Pu_{Bs}. If this member in turn is a Cluster Head of another cluster, the algorithm continues recursively.

In the direct method of distributing a Network Key, the Base Station needs to encrypt the Network Key with the pair-wise key shared with each node in the network. Our recursive secure Network Key distribution method reduces the number of encryptions need to be performed at the Base Station to one operation.

3.2 Periodic Rekeying Protocols

In a network like WSN, the duration of the application is life long. There is no session generation and termination. So periodic rekeying of Network Key and Cluster Key are required to avoid cryptanalysis on the collected ciphers. Periodic rekeying has to be performed if any node finishes $2^{2k/3}$ number of encryptions using the same key, where k is the number of bits in the key.

3.2.1 Periodic cluster rekeying protocol: The protocol is initiated by the Cluster Head. It generates the new Cluster Key CK'_{Ci}, encrypts it with the old Cluster Key CK_{Ci}, calculates the digital signature $DSAG$ (CK'_{Ci}, Pr_{Ci}) and distributes to the members.

```
$CK'_{Ci}$ = Genkey ()
∀$S_k$, if (Child ($C_i$, $S_k$)) then
        $C_i \rightarrow S_k$: $E_{CKCi}$ ($CK'_{Ci}$) || $DSAG$ ($CK'_{Ci}$, $Pr_{Ci}$)
```

Each member S_k receiving the message from its Cluster Head decrypts the message. It verifies the digital signature by evaluating $DSAV\ (CK'_{Ci}, Pu_{Ci})$ and if success, stores the new Cluster Key CK'_{Ci}.

3.2.2 Periodic network rekeying protocol: The Base Station initiates this protocol by generating the new Network Key NK', encrypts it with the old Network Key NK, appends the digital signature $DSAG\ (NK', Pr_{Bs})$ and broadcasts in the network.

$NK' = Genkey\ ()$
$BS \rightarrow * : E_{NK}\ (NK') \parallel DSAG\ (NK', Pr_{Bs})$

Each node in the network receives the message, decrypts it, verifies the digital signature by $DSAV\ (NK', Pu_{Bs})$ and if success, stores the new Network Key NK'.

3.3. Rekeying Protocols with Cluster Dynamics

When an existing member leaves the group or a new member joins the group, the group key in current usage has to be changed to provide forward and backward secrecies.

3.3.1 Member_join_cluster rekeying protocol: The Cluster Head of the cluster to which the new member joins, generates the new Cluster Key CK'_{Ci} and the new key encrypted with the old Cluster Key CK_{Ci} along with the authentication information $DSAG\ (CK'_{\mathrm{Ci}}, Pr_{Ci})$ is sent to all the existing members. The new Cluster Key CK'_{Ci}, the public key of Cluster Head Pu_{Ci} and the public key of Base Station Pu_{Bs} are encrypted with the pair-wise key shared with the new member S_{new} and is sent to the new member.

```
$CK'_{Ci} = Genkey\ ()$
$\forall S_k$ other than the new member, if (Child $(C_i, S_k)$) then
    $C_i \rightarrow S_k: E_{CKCi}\ (CK'_{Ci}) \parallel DSAG\ (CK'_{Ci}, Pr_{Ci})$
$C_i \rightarrow S_{new}: E_{KCiSnew}\ (CK'_{Ci} \parallel Pu_{Ci} \parallel Pu_{Bs})$
```

The key install delay in a cluster includes the propagation delay for the message to reach the cluster member and the time required to install the key at the member node. After the key install delay, the Cluster Head requests the Base Station to initiate *Member_join_network* rekeying.

$C_i \rightarrow BS: Rekey\_req\_ID \parallel DSAG\ (Rekey\_req\_ID, Pr_{Ci})$

The Base Station receives the rekey request from C_i. After verifying the authenticity of the request through digital signature by $DSAV\ (Rekey\_req\_ID, Pu_{Ci})$, it initiates the *Member_join_network* rekeying protocol.

3.3.2 Member_join_network rekeying protocol: Base Station after verifying the authenticity of the rekeying request from the Cluster Head, initiates the network-wide rekeying. The procedure is the same as *periodic Network Rekeying*.

$NK' = Genkey\ ()$
$BS \rightarrow * : E_{NK}\ (NK') \parallel DSAG\ (NK', Pr_{Bs})$

Each node in the network receives the message, decrypts it and after verifying the digital signature by $DSAV\ (NK', Pu_{Bs})$, stores the new Network Key. The new member S_{new} also receives the message from the Base Station but it is not aware of the existing Network key.

Hence the message is useless to it. So the Cluster Head C_i of the new member separately sends the new Network key to it and preserves the backward secrecy property.

$C_i \rightarrow S_{new}$: $E_{KciSnew}$ (NK') || $DSAG$ (NK', Pr_{Ci})

The new member S_{new} decrypts the message using K_{CiSnew}, verifies the digital signature by $DSAV$ (NK', Pu_{Ci}) and stores the Network key NK'.

3.3.3 Member_leave_cluster rekeying protocol: A cluster member leaves the cluster when its battery power gets exhausted (abrupt leave) or when it does not satisfy clustering function (polite leave). Which ever be the case, the Cluster Head C_i should detect it and perform the rekeying. The Cluster Head generates the new Cluster Key CK'_{Ci} and for each of the existing members other than the node that left the cluster, the new Cluster Key is encrypted with the pair-wise key shared with members and sends to them.

```
$CK'_{Ci} = Genkey\ ()$
$\forall S_k$ other than the left member, if (Child $(C_i, S_k)$) then
        $C_i \rightarrow S_k$: $E_{KciSk}$ $(CK'_{Ci})$ || $DSAG$ $(CK'_{Ci}, Pr_{Ci})$
```

Each existing cluster member S_k decrypts the message using K_{CiSk}. After verifying the digital signature by $DSAV$ (CK'_{Ci}, Pu_{Ci}), cluster member stores the new Cluster Key CK'_{Ci}.

After the key install delay at the cluster member, the Cluster Head requests the Base Station to initiate a *Member_leave_network* rekeying protocol.

$C_i \rightarrow BS$: *Rekey_req_ID*, $DSAG$ (*Rekey_req_ID*, Pr_{Ci})

The Base Station receives the rekey request from C_i. After verifying the digital signature by $DSAV$ (*Rekey_req_ID*, Pu_{Ci}), it initiates the *Member_leave_network* rekeying protocol.

3.3.4 Member_leave_network rekeying protocol: The new Network Key cannot be distributed using the existing Network Key because if it does so, forward secrecy will be violated. Distribution is very similar to initial Network Key distribution except that the public key of Base Station is excluded in the message.

```
$NK' = Genkey\ ()$
If this node $C_i$ is a Cluster Head then
$\forall S_k$, if (Child $(C_i, S_k)$ then
        $C_i \rightarrow S_k$: $E_{CKCi}$ $(NK')$ || $DSAG$ $(NK', Pr_{Ci})$
```

Each cluster member S_k decrypts the message using CK_{Ci}. After verifying the digital signature by $DSAV$ (NK', Pu_{Ci}) stores the new Network Key NK'. If this member in turn is a Cluster Head of another cluster, the algorithm continues recursively.

3.4 Memory required for key storage

Each cluster member at the leaf of the tree needs to store a pair-wise key with its Cluster Head, a Cluster Key, public key of the Cluster Head, Network key and public key of the Base Station. Assume that the degree of the network aggregation tree is d and there is n number of Cluster Heads. The Base Station needs to store its own public key- private key pair, Network Key, Cluster Key, pair-wise keys of its own cluster members and public keys of n number of Cluster Heads in the network. Any intermediate Cluster Head needs to store its own public key- private key pair, Cluster Key shared with its members, pair-wise keys of its cluster members, Network Key, Base Station's public key, Cluster Key shared with its Cluster Head, pair-wise key shared with its Cluster Head and its Cluster Head's public key. If Elliptic Curve Cryptography (*ECC*) of 163 bits [9] is used for public key operations then each private key and each public key is 21 Bytes. If the pair-wise key is 16 Bytes, the memory required to store keys in each type of nodes in the network is given in Table 2. The key management schemes based on key pools [4] [2] [7], usually require high memory to store large number of keys. Comparing to them the memory requirement for our scheme is very less.

Table 2. Memory required for key storage

Storage requirement (Bytes)		
Base Station	Cluster Head	Cluster Member
16 d + 74 + 21 n	16 d + 148	90

4. Security Analysis

Sensor networks are prone to different types of security attacks like eavesdropping, node compromise attack, sybil attack, wormhole attack, various denial of service attacks etc. To prevent the chances of cryptanalysis by using the same key for prolonged time, our protocol includes periodic rekeying. If a node gets compromised, it is possible for the attacker to know all the keys stored at that node and can reveal the contents of the communication happening at that node. If we expect that the attacker requires a fixed amount of time to compromise the node, before the attacker could utilize the obtained keys, the Cluster Key and Network Key would have got changed into new ones. But if the rekeying could not happen before the usage of the keys, there is a possibility of attack on rekeying. A compromised cluster member can also act as a Custer Head, generate and distribute the new key by encrypting with the existing Cluster Key. But our protocol withstands this attack by attaching the digital signature for authentication along with the rekeying material. So every member receiving the new key from the compromised node fails to succeed in digital signature verification. Similarly because of the usage of digital signature, our protocol withstands any spoofing attacks and attacks on data integrity. But a compromised cluster member can always supply false sensor readings and prevention of this attack requires mechanisms like secure aggregation.

The request for network rekeying from any Cluster Head to Base Station when new members join or existing members leave are also authenticated. But a compromised Cluster Head can initiate false rekeying request and can cause repeated network rekeying and early exhaustion of power in the sensor nodes. To overcome this type of attack, additional security mechanisms like verifying the status of join/leave directly with the node that has joined/left are to be provided. Our protocol can also withstand wormhole attacks [6] on data routing because the data will be routed to the Base Station only via the path formed through the aggregation tree.

5. Implementation, testing and results

The proposed protocols are implemented for *TinyOS* using *NesC* [17] programming language. The simulations are done in *TOSSIM* [18] environment. The simulator itself decides the topology of the network. The protocols are simulated for various numbers of nodes from 5 to 250. The evaluation parameter chosen is the number of times rekeying is performed at the Cluster Head and at the Base Station. The results of rekeying are given in Figure 1. We have observed that most of the times rekeying is performed because of cluster dynamics and not because of periodic rekeying.

In order to provide evidence for the fact that various protocols performed correctly, we extracted information during simulation. For example, when the protocols are simulated for a network with 25 nodes, the *HEED* algorithm initially formed the network with 2 clusters. The initial Cluster Keys are distributed to these clusters and later the initial Network Key also got distributed. As simulation proceeded, based on the feasibility function of *HEED*, many scenarios occurred like: cluster member left existing cluster, current Cluster Head lost its position, new Cluster Head got elected, members shifted from one cluster to another and so on. On all these situations, the cluster rekeying as well as network-wide rekeying is performed to ensure the total network security. At the end the simulation, there were 5 clusters. For the span of the simulation period, a total of 61 times cluster rekeying was done among all the clusters and 58 times rekeying was done at the network level. These results correspond to the points (25, 61) and (25, 58) in the Figure 1.

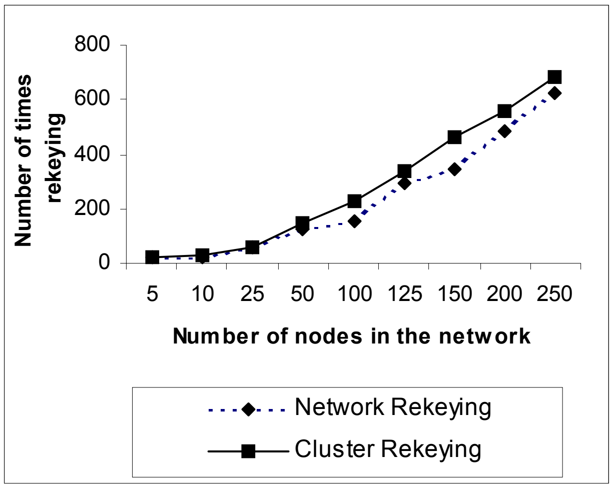

Figure 1. The results of rekeying

6. Related Works

The related works found in the literature are either incomplete or inefficient. Hash-chain method is used in [14] [10] [13] [19] for authentication of key distribution. The Cluster Head pre-computes the one-way sequence of keys $f(K_n) \rightarrow K_{n-1}$; $f(K_{n-1}) \rightarrow K_{n-2}$; …, $f(K_1) \rightarrow K_0$, where n is reasonably large. The last key in the chain will be given to each cluster member as the initial Cluster Key and for every periodic rekeying the next key in the chain is used. If every Cluster Head and member needs to have a hash chain of *n* keys, then storage of *kn* bytes is required, where k is the number of bytes in the shared key. The hash chain idea was introduced in [21] and the optimal value for *n* is said to be 1000. If the key length is 16Bytes, this requires 1000 * 16 bytes storage only for hash-chain, which is not feasible with the memory available with sensor nodes. Also if a new member joins the network or an existing member leaves the network, the key chain is not at all changed to a different chain, which violates the forward and backward secrecies.

The works given in [11] [20] [10] [12] are incomplete because none of them considered all the phases of a complete group key management protocol for both *one-to-all* communication and for *one-to-one* communication. Work proposed in [11] [20] considers only *one-to-one* communication, but does not consider periodic rekeying. When a member joins, the rekeying operation does not utilize the existing Cluster Key to distribute the new key to existing members. In the work [20], when a Cluster Head leaves its position, members join the cluster of their grandparent. But there is no guarantee that all cluster members will be in the communication range of their grandparent. Although work given in [10] uses a network wide master key for distribution of Cluster Keys to nodes, it does not utilize this master key for securing *one-to-all* communication from Base Station. Also when a new node joins, rekeying is not performed. When a node gets compromised, network master key and hash chain are not changed.

The above discussions clearly shows that key management protocols considered in the literature are neither complete nor efficient as our work in this paper.

7. Observations from simulations

During simulations we have observed that most of the times rekeying is performed because of cluster dynamics and not because of periodic rekeying. In many clusters, multiple times rekeying is performed within a short span of time. Similar observation is found in the case network rekeying. *HEED* clustering protocol groups the network in such a way that Cluster Heads are evenly distributed and depletion of their energy is not faster compared to cluster members. Based on the feasibility of the clustering function, cluster groups and memberships change at regular clustering intervals. During a re-clustering phase, the members of a cluster may leave this cluster and join another. Cluster dynamics is very high in this case as at the same time multiple nodes may leave and | or join a cluster. This leads to multiple times execution of *Member_Leave_Cluster* Protocol in the first Cluster Head and multiple times execution of *Member_Join_Cluster* Protocol in the second Cluster Head. And for every leave and join of a cluster member, execution of network rekeying also has to be done. So if we follow the strict rekeying rules upon every cluster member change, this will lead to early depletion of energy in all nodes of the network. In scenarios like these, it may be noted that only the existing network nodes are leaving their original clusters and joining some other clusters. In such cases there is no need for rekeying. However authentication of these nodes in the new clusters that they join is needed. The original Cluster Head also needs to be informed about the current status of its previous member.

The rekeying requests from many Cluster Heads may reach the Base Station in a short span of time. If the Base Station performs rekeying for every request, the current Network Key may get changed to new one even before using it. To avoid such unnecessary rekeying, a batch rekeying can be performed by the Base Station. In batch rekeying, the Base Station collects rekeying requests for a span of time and performs a single rekeying instead of multiple time rekeying.

So if we can frame proper authentication protocols and batch rekeying policies, the unnecessary rekeying can be avoided. This can provide longer life to the secure network.

8. Conclusion

We have implemented a complete set of protocols for securing *one-to-all* and *one-to-one* data communications in Wireless Sensor Networks. We have considered initial generation and distribution of keys as well as rekeying periodically and rekeying upon group dynamics. All the messages required for rekeying are properly authenticated using digital signature scheme. The protocols are designed not to violate the backward secrecy and forward secrecy policies required for any group management scheme. They are very simple, efficient and viable with respect to communication, computation and storage requirements. Thus they are suitable to provide perfect security for resource-constrained networks like Wireless Sensor Networks.

The observations during simulation show that if we follow the strict rekeying rules, they force to perform multiple times rekeying in the cluster and network level. These unnecessary rekeying can be avoided by the formation of proper authentication protocols and batch-rekeying protocols so that the energy constrained sensor networks can achieve long life.

9. References

[1] Michel Abdalla, Mihir Bellare, "Increasing the Lifetime of a Key: A Comparative Analysis of the Security of re-Keying Techniques", *Asiacrypt* 2000, LNCS 1976, pp. 546-559, Springer-Verlag.

[2] J. Hwang and Y. Kim, "Revisiting random Key Pre-distribution schemes for Wireless Sensor Networks*", 2nd ACM Workshop on security of Adhoc and Sensor Networks*, pp. 43-52, ACM Press, 2004.

[3] S. Madden, S. Szewczyk, M.J. Franklin and D. Culler, "Supporting Aggregate Queries Over Ad-Hoc Sensor Networks", *Workshop on Mobile Computing and Systems Application*, 2002

[4] Laurent Eschenauer and Virgil D. Gligor, "A Key Management Scheme for Distributed Sensor Networks", *ACM Conference on Computer and Communications Security*, 2002, pp. 41-47.

[5] Ossama Younis and Sonia Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad-hoc Sensor Networks," *IEEE Transactions on Mobile Computing*, volume 3, issue 4, pp. 366-379, 2004.

[6] Chris Karlof and David Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", *First IEEE international workshop on Sensor Network Protocols and applications*, May 2003, pp. 113-117.

[7] Donggang Liu, Peng Ning and Wenliang Du, "Group Based Key PreDistribution in Wireless Sensor Networks", *ACM WiSE*'05, 2005.

[8] Jibi Abraham and K S Ramanatha, "An Efficient Protocol for Authentication and Initial Shared Key Establishment in Clustered Wireless Sensor Networks", *Third IFIP/IEEE International Conference on Wireless and Optical Communications Networks*, April 11-13, 2006, India

[9] Jibi Abraham and K S Ramanatha, "Security Protocols for Wireless Sensor Networks Based on Tiny Diffusion and Elliptic Curves", *IASTED International Conference on Networks and Communication Systems*, pp. 35-40, March 29-31, 2006, Chiang Mai, Thailand, ACTA Press.

[10] Tassos Dimitriou, Ioannis Krontiris and F. Nikakis, "A Localized, Distributed Protocol for Secure Information Exchange in Sensor Networks", *IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks*, April 2005, pp. 240a

[11] Jyh-How Huang, Jason Buckingham and Richard Han "Level Key Infrastructure for Secure and Efficient Group Communication in Wireless Sensor Networks", *IEEE International conference on Security and Privacy for Emerging Areas in Communication*, September 2005, pp. 249-260.

[12] Leonardo B. Oliveira, Hao Chi Wong, Antonio A. Loureiro, "LHA-SP: Secure Protocols for Hierarchical Wireless Sensor Networks", *IFIP/IEEE International Symposium on Integrated Network Management*, 2005, pp. 31-44

[13] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks", *ACM CCS*, October 2003, pp. 62-72.

[14] Taejoon Park and Kang G Shin, "LiSP: A Lightweight Security Protocol for Wireless Sensor Networks", *ACM-Transaction on Embedded Computing Systems*, Vol 3 No. 3, 2004, pp 634-660.

[15] Jibi Abraham and K S Ramanatha, "Wireless Sensor Networks Security: Current Research – A Survey", *International Journal of Lateral Computing*, Vol 3. No. 1, pp. 149 – 167, August 2006, ISSN 0973- 208X, India.

[16] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, "System architecture directions for network sensors", *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93-104.

[17] David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer and David Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems", *Programming Language Design and Implementation*, 2003, pp. 1-11.

[18] Philip Levis, Nelson Lee, Matt Welsh, David Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", *SenSys*, 2003, pp. 126-137.

[19] Fei Hu, Neeraj K. Sharma, "Secure Wireless Sensor Networks: Problems and Solutions", *Journal on Systemics, Cybernetics and Informatics,* Vol.1, No.9, 2004.

[20] T. Dimitriou Dimitris Foteinakis, "Secure and Efficient In-Network Processing for Sensor Networks", *First Annual International Conference on Broadband Advanced Sensor Networks*, 2004.

[21] Leslie Lamport, "Password Authentication with Insecure Communication", *Communication of ACM*, Vol 24, No 11, November 1981, pp 770–772.

[22] M. Horton, et. al., "Mica: The Communication of Microsensor Motes", Sensors online Magazine, April 2002.

Authors

Jibi Abraham

Jibi Abraham completed her B.Tech in Computer Science & Engg from Govt. College of Engg. Trichur, University of Calicut, Kerala in 1993 and M. S. in Software Systems from BITS, Pilani, Rajastan in 1999. Now she is pursuing her PhD under Visveswaraiah Technological University, Karnataka in the area of Network Security. She has a total of 3 years of experience in Industry and 10 years in teaching. Currently she holds the position of Assistant Professor in the Dept. of Computer Science & Engg in M. S. Ramaiah Institute of Technology, Bangalore. Her areas of research interests include Network routing algorithms, Cryptography, Network Security of Wireless Sensor Networks and Algorithms Design.

K S Ramanatha

K.S. Ramanatha holds a Ph.D. degree from the Indian Institute of Science, and B.E. and M.Tech. degrees from Mysore University. His Ph.D. thesis won the Alumni Medal for the best thesis in the Electrical Sciences Division of IISc, during 1982 and the design of completely testable programmable logic arrays developed by him for the thesis was recognized as RAMANATHA TECHNIQUE in leading books and research papers. He was a Visiting Professor at the University of Manitoba, Canada, from 1985 to 1988. A former Principal of MSRIT, he now holds the position of Administrator and Professor of Computer Science & Engg. in that Institute. His current research interests are in the areas of design of MANETS, Security and Data Mining.