Energy and Communication Efficient Group Key Management Protocol for Hierarchical Sensor Networks

Biswajit Panja, Sanjay Kumar Madria
Department of Computer Science
University of Missouri-Rolla, Mo 65401
bptfc,madrias@umr.edu

Bharat Bhargava
Department of Computer Science
Purdue University, West Lafayette, IN 47907
bb@cs.purdue.edu

Abstract

In this paper, we describe group key management protocosl for hierarchical sensor networks where instead of using pre-deployed keys, each sensor node generates a partial key dynamically using a function. The function takes partial keys of its children as input. The design of the protocol is motivated by the fact that traditional cryptographic techniques are impractical in sensor networks because of high energy and computational overheads. The group key management protocol supports the establishment of two types of group keys; one for the sensor nodes within a group, and the other in a group of cluster heads. The protocol handles freshness of the group key dynamically, and eliminates the involvement of a trusted third party (TTP). We have experimentally evaluated the time and energy consumption in broadcasting partial keys and group key under two sensor routing protocols (Tiny-AODV and Tiny-Diffusion) by varying the number of nodes and key sizes. The performance study provides the optimum number of partial keys needed for computing the group key to balance the available security and power consumption. The experimental study also concludes that the energy consumption in SPIN [9] increases rapidly as the number of group members increases in comparison to our protocol.

1. Introduction

Sensor networks [1] have become an important area of research because of their applications in military and disaster relief. The most limiting factors of a sensor node are its battery capacity and available memory. Thus, the energy and storage conservation are two important issues at the node level and at network level.

Security is one of the most important issues in distributed ad hoc sensor networks. For example, a wireless sensor network uses a radio frequency (RF) channel [10], which is not a secure channel. It is also difficult to prevent an adversary sensor node from compromising the security of sensor networks because of untraceable sensor nodes and less physical protection [2,3,5]. To control information access in a hierarchical sensor environment, only authorized sensors should have the cryptographic keys by which they can decode the disseminated information. Thus, a group key management is required for such a hierarchical environment as it can implement different access control policies at each level and provide mechanisms for secure group communication by eliminating compromised nodes. In the traditional cryptographic techniques for security, every sensor node would need a {private, public} key pair which is impractical because of high energy consumption and scalability.

In this paper, we propose a group key management protocol using a hierarchical architecture consisting of different groups with a unique group key. Using this approach, multi-level security can be achieved to secure the group of sensors at different levels. There are two types of group keys: intra-cluster, and inter-cluster. The intra-cluster group key is used for encryption/decryption of messages inside a sensor network group, whereas the inter-cluster group key is used for groups of cluster heads.

The two most important advantages of dynamic partial keys over pre-deployed keys are that (1) sensor nodes need not store too many keys and (2) the dynamic key may not be compromised because it changes frequently. In our proposed scheme, every sensor node in a group generates a partial key dynamically, and used it for computing the group key in a bottom up fashion. Once the sensor network is deployed, it is organized in a hierarchical fashion. After that, a cluster head (leader) gets messages from all its group members to know their level and location in the sub-tree, and in response it sends a message. It also requests the leaf nodes (initiator) to compute their partial keys. In this protocol, the partial keys are computed using the function associated with each node which uses partial keys of their descendents as arguments. The key computation starts by leaf nodes generating random numbers as their partial keys, because they have no descendents. The cluster head computes the group key using an optimum number of partial keys. The decision for choosing the number of

This research is partially supported by NSF grant EIA-0323630.

partial keys to be used is based on the security and energy consumption.

In this paper, we modified the Tree Based Group Diffie-Hellman (TGDH) protocol [7] for group key management using a general tree structure. In addition, Diffie-Hellman protocol [9] is used for computing the group key. Using the modified TDGH, a new group key management scheme is presented. The experiments, with a sensor network environment created in NS-2 and TinyOS are performed using two sensor routing protocols (Tiny-AODV and Tiny-Diffusion) by varying nodes, key sizes and energy consumption. It is observed that Tiny-AODV is slower than Tiny-Diffusion in terms of broadcasting the partial keys and the group key. Also, the Tiny-AODV takes more time to re-establish a route and re-send the partial keys. The experiments for energy computation and delivery of partial keys helped in selecting the optimum partial key and group key sizes. It is observed that a 300-bits group key size would be reasonable considering memory and communication overheads. In experiments, the optimum (energy, security) group key size is 300 bits, which can be computed from 15 partial keys of 20 bits each. It is known that decrypting 300-bits group key needs 2^{300} micro seconds, with the decryption rate of 1 bit per micro second [10].

We performed experiments to compute energy consumption and concluded that the protocol consumes very small amount of energy (approximately 0.245 joule) in the process of computing 15 partial keys, broadcasting them, computing and broadcasting the group key. The energy consumption is very small compared to the total available energy of 4,61700 joule (for 15 nodes with two batteries each having 15,390 joule per battery). The proposed protocol conserves energy and communication with respect to SPIN [6] and pre-deployed key protocol. To save communication cost and energy, re-keying of the group is done by the cluster head by sending a message to the sensor nodes, which contains information for adding or removing certain partial keys to generate the new group key. To guarantee that all the nodes in a group received the information, they send a reply (REP) message. If the cluster head does not get the REP from every node, it re-broadcasts the message.

2. Related work

Kim et al. [7] proposed a group key agreement protocol called Tree Based Group Diffie-Hellman protocol [TGDH], which is based on the Diffie-Hellman key exchange. In this protocol, a distributed key agreement has been considered rather than a centralized group key agreement. Different group agreement protocols have been proposed. In centralized group key distribution, one key server generates keys and distributes them to the group, while in decentralized approaches the key is computed dynamically. The basic requirements for group key agreement protocols [1, 5] are key freshness, group key secrecy (forward and backward) and key independence.

In SPIN [6] two security concepts are used: SNEP (Secure network encryption protocol) and micro TESLA (Time, efficient, streaming, loss-tolerant authentication protocol). The advantages of SNEP are low communication overhead and semantic security. A DES-CBC chaining algorithm is used to maintain data confidentiality in SPIN, and a MAC is used to keep messages unaltered. A special counter is used to maintain the sequence of messages. The counter value will never be the same, so the encrypted message is different for the same data. The disadvantages of SPIN are that (1) it is based on one-to-one communication but nodes in sensor networks work in groups and (2) it does not consider the security in hierarchical structure and clustering which are important for sensor network applications such as in military.

Eschenauer et al [4] presented a key management scheme which has selective distribution and revocation of keys in sensor nodes. Their scheme is based on the probabilistic distribution of the key, which guarantees that two neighboring nodes will have at least one common key in their key ring. This key is used by the neighboring nodes to encrypt/decrypt messages. The disadvantage of this approach is that, pre-deployed keys are comparatively easy to forge than a dynamic key. Also, each node needs more memory to store many keys, and therefore, it is impractical

3. Hierarchical sensor network model

In this section, the architecture of a hierarchical sensor network with multiple levels consisting of sensor nodes, cluster heads, and relay nodes is described. There are two types of sensor groups; one is a group of sensor nodes lead by a cluster head, and the other is a group of cluster heads with one cluster head as head of that group. Figure 1 shows the architecture. In this model, each sensor group collects data from a particular geographical area and sends the data to the nearest sensor nodes. If the neighboring nodes are relay nodes, they forward those data using the appropriate routing path. Finally, the cluster head aggregates the data and forwards that to its upper level cluster head.

Three different types of identifications are used in this model: a unique identification (ID) for each sensor node, each cluster head and each group of clusters. The assignments of the IDs are done in the following ways:

- The IDs of sensor nodes are given by the cluster head of that particular group.
- The IDs of cluster heads are given by the Head of Cluster Heads [HCH] of a particular geographical area. HCH is the cluster head, which is responsible for leading the group of cluster heads.

- Also, the IDs of the group of sensor nodes are given by HCH.

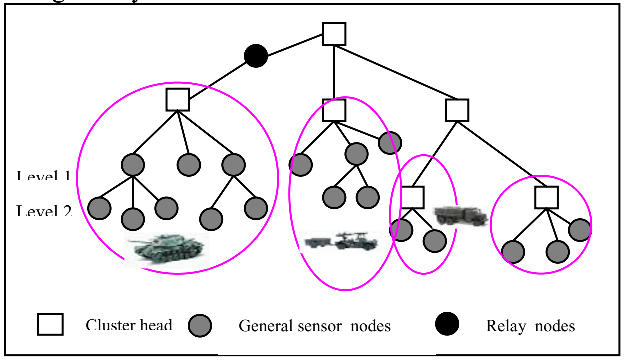

Figure 1. Hierarchical architecture

4. Partial key computation

In this section, we describe the scheme for computing the partial key in each sensor node. We assume that after the organization, the cluster head of each group knows the position (*Pos*$_{<l,v>}$) (level in the sub-tree) of its group members. It send a message *Msg*(*Pos*$_{<l,v>}$) to let them know their position. Next, the cluster head sends a message (*Rep*$_{<init,node>}$) to the leaf nodes to compute the partial keys. The function *f(partial key of child1, partial key of child2)* is used to compute the partial keys in each node. As the leaf nodes do not have any decedents, they generate random numbers as their partial keys. Then, it uses a simple approach to compute the partial keys of non-leaf nodes. The parents of the leaf nodes compute their partial keys using a function *f()*. The function is $f(k_1,k_2)=\alpha^{k_1 \oplus k_2} \bmod p$, where p is the prime number, α is the primitive root of P and k_1, k_2 are keys ($k_1,k_2<p$). The arguments of the function are the partial keys of their children. Using a bottom up approach, all non-leaf sensor nodes can compute their partial keys.

If the tree is binary, then the function for computing the partial key K is $f(K_{<l+1,\,2v>}\ K_{<l+1,\,2v+1>})$, where l is the level in the tree and v is the position of the node from left. For example in Figure 2, to compute the $K_{<2,\,0>}$ we need the function $f(K_{<3,\,0>}\ K_{<3,\,1>})$. The computation of $K_{<2,\,1>}$ is not possible using $f(K_{<3,\,2>}\ K_{<3,\,3>})$ as children now have different parents. If the function $f(K_{<l+1,\,2v+1>}\ K_{<l+1,\,2v+2>})$ is used then we are able to compute $K_{<2,\,1>}$. From this we observe that the levels l do not change, but the position v changes depending on the number of children. In a non-binary tree, the algorithm needs to count the siblings in the left part of the sub-tree for calculating a key for the parent node. It then computes m where *m = siblings - 2* so the function would be $f(K_{<l+1,\,2v+m>}\ K_{<l+1,\,2v+1+m>})$.

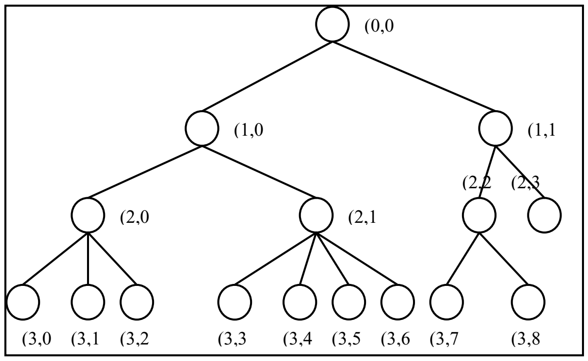

Figure 2. Dynamic partial key computation

The advantage of this function is that it is difficult to decode. When the nodes know the function ($f(k_1,k_2)=\alpha^{k_1 \oplus k_2} \bmod p$) then they do not need to analyze the known function. The complexity of the protocol is *O(logn+mn),* because the partial keys and the group key are computed in a tree structure. For *n* nodes, it takes *O(logn)* time, and computing the *m* partial keys from n nodes takes *O(mn)* time.

A one time symmetric key is used for generation and verification of the MAC first time. This key is also used for encryption/decryption of partial keys, intermediate keys and group key. Once the group key is computed, the symmetric key is discarded. Figure 3 provides the algorithm for computing the partial keys in a tree which is based on the position of the nodes in that group.

```
Algorithm:
1. if (tree is binary) then
2. {   $f(K_{<l+1,\,2v>}\ K_{<l+1,\,2v+1>})$   }
3. else if (it is not binary), then
4.     {   $f(K_{<l+1,\,2v+m>}\ K_{<l+1,\,2v+1+m>})$
   m = [ If there is sub-tree left side of node, left
   siblings – (number of sub-tree)x 2 – 2 ] }
5. if (no sub-tree in the left side of the node) then
6. {   $f(K_{<l+1,\,2v>}\ K_{<l+1,\,2v+1>})$   }
7. else if (sub-tree is there in the left side of the node),
   then
8.     {   $f(K_{<l+1,\,2v+m>}\ K_{<l+1,\,2v+1+m>})$   }
```

Figure 3. Algorithm for dynamic partial key computation

5. Group key computation

We have used the multi-party Diffie-Hellman and TGDH protocol [7] to propose a new group key computation method for sensor networks. To accomplish this proposition, the leaf nodes work as the initiators and the cluster head as the leader. Starting from the initiator sensor nodes, every sensor node contributes its partial key for computing the group key. The leader node accumulates all partial keys for computation of the group key. This is a bottom up approach, as partial keys are accumulated from leaf nodes to the parent nodes. In the following subsection, we show the group key computation with and without using the blind factor. The blind factor is a unique number generated by a sensor node.

5.1 Group key computation without blind factor

We use the following approach for group key computation without the blind factor [1]. As the leaf nodes act as the initiators, they first broadcast their partial keys. The parent nodes of the leaf nodes get the partial keys and then add their own partial keys and rebroadcast it. As it is a bottom up approach, the cluster head will have all the partial keys, and it will compute the group key using its partial key contribution. After that, the cluster head broadcasts the group key.

Initially, a pre-deployed one time symmetric key is used to encrypt and decrypt the partial keys and the group key. There after, only the group key is used for this purpose. The identification of the nodes is attached with the encrypted partial keys. The sensor nodes check the identification before decrypting the partial keys, as the parent nodes only need the partial keys of their children. In this way, they can have early rejection of packets, which saves communication and computation overheads. The other group members cannot compute the group key because they cannot get the partial key of the cluster head, since it does not broadcast its partial key.

In Figure 4, the leaf nodes are $M_1, M_2, \ldots, M_9$. To start, M_1 computes the partial key g^{S1} and broadcasts it. The parent node M_{10} gets the partial keys from M_1 and other children. Here, g is a generator of the multiplicative group Z_P* (i.e. the set $\{1, 2\ldots p\text{-}1\}$, p is the prime) and $S1$ is a randomly chosen secret number for member M_1. Likewise, member M_2 computes g^{S2} and broadcasts it, and the parent M_{10} gets the partial keys. In this way, member M_{10} receives g^{S1S2S3}, and raises the power by S_{10} to get the intermediate key *(IK).* Here, g^{S10} is the partial key contribution of M_{10}.
In the following paragraphs, we discuss two types of group keys: the intra-cluster and the inter-cluster.

The intermediate keys in M_{10}, M_{11}, and M_{12} are $IK1 = g^{S1\ S2\ S3\ S10}$, $IK2 = g^{S4\ S5\ S6\ S11}$, *and* $IK3 = g^{S7\ S8\ S9\ S12}$, respectively. The intermediate keys are encrypted using a one time symmetric key, as explained earlier. The cluster head computes the group key K, using *IK1*, *IK2*, and *IK3* and its contribution g^{s13}.

$$K = g^{S1\ S2\ S3\ S10\ S4\ S5\ S6\ S11\ S7\ S8\ S9\ S12s13} \quad \textit{[Intra-cluster group key]}$$

Then, it encrypts the group key using the symmetric key. The authentic nodes, which have the symmetric key, can decrypt the group key. The cluster head broadcasts the group key to its group members, so that every sensor node gets the group key. This group key is called the intra-cluster group key, and is used for encryption/decryption of messages inside the group of sensor nodes.

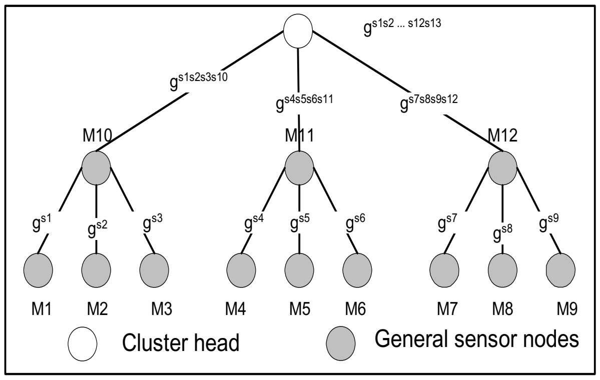

Figure 4. Intra-Cluster key computation

For inter-cluster encryption/decryption a different group key is computed. The inter-cluster group key is not known to the general sensor nodes. Figure 5 shows the computation of the inter-cluster group key. The intermediate key in C_7, C_8, C_9 are $IK1_{inter} = g^{C1C7}$, $IK2_{inter} = g^{C2C3C8}$, and $IK3_{inter} = g^{C4C5C6C9}$

The head of the cluster heads (HCH) computes the inter-cluster group key C using intermediate keys $IK1_{inter}$, $IK2_{inter}$, $IK3_{inter}$, and its contribution g^{c10}.

$$C = g^{C1C7C2C3C8C4C5C6C9c10} \quad \textit{[Inter-cluster group key]}$$

The HCH broadcasts the inter-cluster group key to the cluster heads. The cluster heads use this group key for encryption/decryption of messages among the cluster heads.

5.2. Group key computation using blind factor

We can compute the group key using a blind factor. The advantage of using a blind factor is that an attacker will not be able to get the group key when the cluster head broadcasts the group key. In Figure 4, the intermediate keys are $IK1 = g^{S1\ S2\ S3\ S10}$, $IK2 = g^{S4\ S5\ S6\ S11}$ and $IK3 = g^{S7\ S8\ S9\ S12}$. After computation of *IK1, IK2,* and *IK3* the parent nodes M_{10}, M_{11}, M_{12} broadcast the intermediate keys. The children of M_{10}, M_{11}, and *M12* are interested in those keys, as they need to

remove their contribution from the *IK*. Then they insert a randomly chosen blind factor *B*. The keys, after inserting blind factor, are as follows.
$IKB1 = g^{\mathbf{B1}\ S2\ S3\ S10}$, $IKB2 = g^{S1\ B2\ S3\ S10}$, *and* $IKB9 = g^{S7\ S8\ B9\ S12}$. The cluster head gets the broadcasted keys *IKB1,…, IKB9*. The cluster head then computes the group key *K*, using *IKB1…IKB9* and its contribution g^{s13}.

$$K = g^{B1\ S2\ S3\ S10\ S4\ S5\ S6\ S11\ S7\ S8\ S9\ S12s13}$$

After the group key computation, the cluster head broadcasts the group key with blind factor. Now the authentic sensor node can recognize its blind factor. Each member remove its blind factor that it received from the cluster head. They reinsert their original contribution S_i $(i = 1...n)$ for getting the group key. The same method is used to compute the inter-cluster group key. A symmetric key is used for encryption and decryption of partial keys. The cluster head uses the same symmetric key for encryption of the group key.

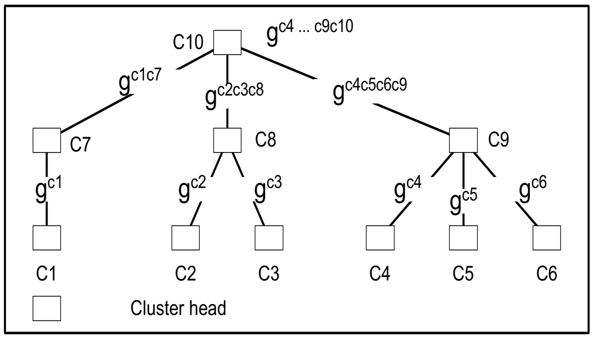

Figure 5. Inter-Cluster key computation

5.3. Updating a group key

Our key management protocol provides a scalable approach for updating group keys for large dynamic groups. Section 3 shows that large dynamic groups, re-keying the group on each membership change becomes unsustainable. One of the approaches to keep the key fresh is by re-keying the group at fixed intervals; this approach is computationally expensive as the partial keys and the group key will be computed again. Another approach for updating the key would be to send a message from the cluster head to its group members consists of instructions to remove or add a certain partial key from the group key in order to get the new group key. The group key *K,* which is described in Section 5.1 is:

$$K = g^{S1\ S2\ S3\ S10\ S4\ S5\ S6\ S11\ S7\ S8\ S9\ S12s13} \ \textit{[Old group key]}$$

For example, the cluster head sends an encrypted message to its group member for removing *S10* from their group key. The new group key *K* would be:

$$K = g^{S1\ S2\ S3\ S4\ S5\ S6\ S11\ S7\ S8\ S9\ S12s13} \ \textit{[New group key]}$$

To guarantee that all the sensor nodes received the message to update the part of the group key, nodes send an acknowledgment *(ACK)* message to the cluster head. If the cluster head receives the reply *(REP)* from all the nodes, then it sends the next message that "the new group key is in effect now". If the cluster head does not receive the reply message *REP* from all its group members, then it resends (broadcast) the message until it can get a reply message or it can come to know events, such as a particular sensor node does not have battery power left for communication. To keep track of all the partial keys, the cluster head can use a primary index by sorting the IDs of its group member nodes.

5.4. Analysis: Energy and security level with respect to key size

This subsection explains the analysis of balancing the energy consumption verses security level by choosing the appropriate key sizes in the proposed protocol.

Notations:

L_f	Leaf level nodes in the hierarchy
L_{vt}	Leaf level
K_{lf}	Partial keys in the leaf nodes
P_k	Pre-deployed key in the network, can be symmetric or asymmetric
K_p	Parent key of the sub-tree
P_n	Parent nodes of each sub-tree
L_v	Levels in the hierarchy as in figure 1
$f(\,)$	Function with the partial keys of children as arguments
E	Energy consumption (joule)
P_{total}	Total energy in the network (joule)
$P_{consumed}$	Consumed energy for group key computation
G	Group of sensors in the network
N_g	Number of groups in the network
S	Size of key
L_n	Number of levels in each group
C_n	Number of partial keys considered for the group key
V	Battery voltage (volt)
C	Capacitance (farad)
f	Frequency (Hz)

Let $L_f \in L_{vt}$ and L_v =1, 2, …….n [L_v = 0 is the root]. The parent key K_p is computed from the function $f(K_{lf}, K_{lf+1})$ where $f(k) = \alpha^k \bmod p$.
The energy remaining in each sensor node is computed using their levels in the architecture. It is known that the energy consumption in a circuit is : Power $P = V^2 .f. C$
If the power (energy) consumption at each level is P then the total power in all groups is:

$$\text{Total power} = \sum P \times L_v \times N_g$$

$$\text{Energy remaining} = P_{total} - \sum P \times L_v \times N_g$$

The energy required at each level will compute the total energy needed in the computation of the group key, and is given by $\int^n L_v = \sum P \times L_v \times L_n$
By considering the partial keys at each node level and at their parent level energy consumption would be:

$$\prod_{Lf}^{Pn} (L_v \le K_p \times L_n) \quad \exists K_p \in k_{lf} + P_n$$

If n partial keys are considered to form the group key then the energy consumption with respect to key size would be: $P_{consumed} = (K_{lf} + K_p) \times C_n \times E$
The energy consumption for the group key must be less than the total available energy. Thus, $(K_{lf} + K_p) \times C_n \times E \le P_{total} \le E \times N_g$
It is assumed that there exists more than one group in the network. If the partial key size varies from 20 to 100 bits then the energy consumption would be:

$$E = \sum_{S=20byte}^{100} (K_{lf} + K_p) \times C_n \times S$$

The partial key sizes are based on the group key size, the number of partial keys considered for the group key and the available energy.
The energy consumption for computing the partial keys and the group key proportional to the security requirement in terms of key sizes can be expressed as follows:

$$\forall C_n \in G,$$

$$\exists K_p, K_{lf} \mid C_n \times (K_p + K_{lf}) \times E \propto Sec$$

where Sec is the time to the decrypt the key at a decryption rate of 1 bit per micro second. It is observed that, as the key size increases the security and energy consumption would increase, and it is not possible to have $P_{consumed} \ge P_{total}$.

If $\dfrac{(L_f \times L_n \times E) \cong (L_{vt} \times L_n \times E)}{P_{consumed}} \le 1$,

then the cluster head can choose the partial keys from the leaf nodes. From the above, we can find the number of partial keys to be used for balancing the security requirement based on key size verses the energy consumption.

6. Performance evaluation and observation

For performance analysis, we have implemented the group key management protocol in TinyOS and NS-2. As NS-2 is not developed for sensor network, a sensor network environment is created in NS-2 by plugging in sensor agent, energy model, and multi-channel model developed by the United States Navy.

The cluster head computes the group using the method as described in Section 5. It chooses a certain number of partial keys in order to compute the group key. It also stores the identification of the nodes along with the partial keys so that it can save memory by discarding the partial keys received earlier.

In order to measure how fast the partial keys are delivered to the cluster head, we did experiments using two sensor routing protocols: Tiny-AODV and Tiny-Diffusion. The sensor nodes, which do not fall in a 10 meters range, generate partial keys so that the cluster head can get the partial keys from different regions in its group. Then, they establish a route to the cluster head using the routing protocol and send the partial keys using that route. The reasons for restricting the number of sensor nodes and broadcasting the partial keys are to reduce communication overhead, and energy consumption. Figure 6 shows the time taken by the network to send the first 25 partial keys to the cluster head using the Tiny-AODV and Tiny-Diffusion routing protocols. The algorithm for generating and broadcasting partial keys and computing the group key was kept the same in both cases.

Routing Protocols	Tiny-AODV, Tiny-Diffusion
Area	2000 x 2000 meter
Number of nodes per group	50
Channel	Single (wireless)
Simulation time	160 sec
Transmitting power	0.175 mW
Receiving power	0.175 mW
Idle	0.0 W
Initial energy	0.5 Joule
Sensing power	0.00000175 mW

Table 1. Parameters for simulation

We observed that the Tiny-AODV takes approximately 54 seconds to deliver the first 25 partial keys to the cluster head; whereas Tiny-Diffusion takes almost 15 seconds. This experiment is performed to determine the effectiveness of routing protocols in terms of how fast the partial keys can reach the destination. After that, the group key is computed by the cluster head. Then it is broadcasted to all its group members. We found that both routing protocols take a comparably small amount of time for broadcasting partial keys.

Figure 7 shows the time taken by the network to broadcast the group key generated based on the collection of partial keys using the Tiny-AODV and Tiny-Diffusion routing protocols. From experiments, it is observed that the group members whose IDs are 1 through 25 receive the group key in approximately 7.57 seconds if Tiny-AODV is used whereas if Tiny-Diffusion is used then it takes approximately 11.82 seconds. The reason is that, Tiny-Diffusion uses a routing table for creating routes and therefore, it takes more time to deliver the initial packets compared to Tiny-AODV because of its proactive features. There is a very small time difference in broadcasting the group key using Tiny-AODV and Tiny-Diffusion. Thus, delivering the partial keys (as shown in Figure 7) to the cluster head using Tiny-AODV takes more time than Tiny-Diffusion because of the partial key broadcast from every node, and then delivering it to the cluster head requires many-to-one communication.

In Figure 8, we present the results for time taken in broadcasting partial keys of size 20 and 30 bits. The group key in the cluster head is computed using the first 15 partial keys. The group key size would be 300 and 450 bits, respectively. We observed that the time taken to accumulate partial keys of size 300 and 450 bits remains almost same. Theoretically, the time required to decrypt 300 bits key size is 2^{300} micro seconds, at a rate of 1 bit decrypt per micro second. Therefore, the forgery time for size of 300 bits group key is quite high.

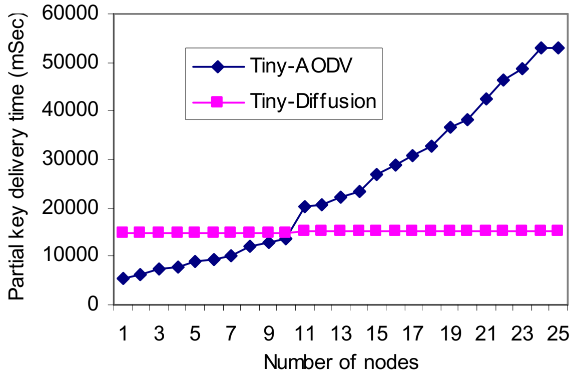

Figure 6. Partial key delivery to the cluster head

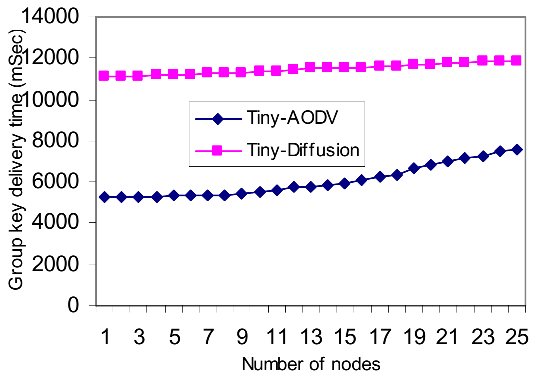

Figure 7. Broadcasting group key

We performed another experiment to analyze the energy consumption in the proposed model. As a reminder, the power consumed by the sensor nodes for transmission and reception is set at 175 mW, for sensing 0.00000175 mW, and the initial energy of the general sensor nodes is 0.5 joule. The cluster head consumes the same power for transmission and reception with its initial energy is set to 2.5 joules. Figure 9 shows the energy consumption graph for generating 20 partial keys and delivering those to the cluster head. This also includes the energy consumption for communication before delivering the partial keys. From this experiment, we observe that changing the number of senders has impact on the energy consumption, and this experiment helps to decide the number of partial keys that should be chosen in order to balance the energy consumption and security. The optimum number of partial keys is 15 with respect to the current configuration. Although the total power consumption for generating partial keys, delivering, computing the group key and broadcasting back is not shown in the figure, it is approximately 0.245 joule.

One way to protect the network from intruders is by updating (re-keying) the group keys frequently so that an intruder cannot get enough time to forge the group key. In our model, we update the group key using the technique described in Section 5.3. As explained earlier, in order to save communication and computation cost, instead of re-computing the partial keys and the group key, the cluster head sends a message to its group members for removing or adding certain partial keys from the group key to obtain the new group key.

In Figure 10, the energy consumption by SPIN and our key management protocol are compared. Though SPIN is used for one-to-one node communication, here it used for group communication. It is observed that SPIN takes more time for communicating within a group. We observed that the energy consumption of

SPIN increases exponentially with the increase of the number of nodes in a group.

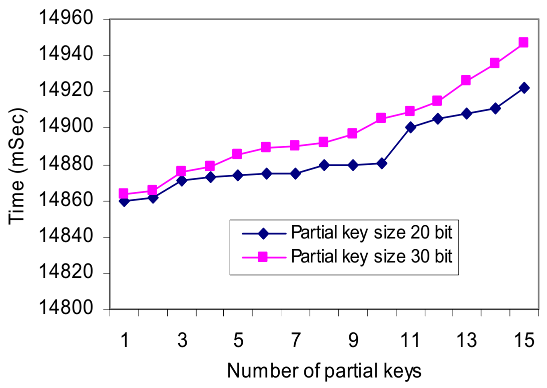

Figure 8. Time taken for different key sizes

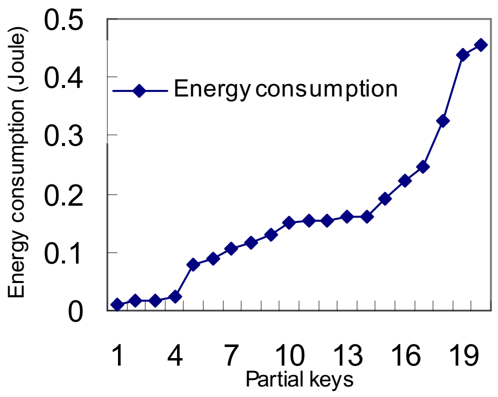

Figure 9. Energy consumption verses the number of partial keys

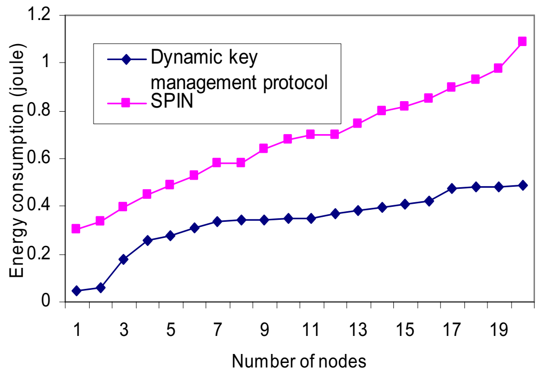

Figure 10. Comparison of dynamic key management protocol with SPIN

7. Conclusions

In this paper, we have described a group key management protocol where the partial keys are computed dynamically, instead of using pre-deployed keys. A group key is generated based on the partial keys of the group members. The dynamic partial keys have some advantages over pre-deployed keys, as many pre-deployed keys need to be stored to provide secure communication, which is not feasible in sensor nodes because of the limited memory. It is easy to re-compute the partial keys as each node uses a function with the partial keys of its children as arguments of the function to compute its partial key. Using a detailed simulation study it is shown that the proposed protocol is able to compute the partial keys and the group key within a very small time period. We observed that the energy consumption for generating the partial keys and the group key is very small compared to the total available energy. It is also sown by an experiment that the energy consumption of SPIN increases exponentially as the number of group nodes increase, but our proposed protocol consumes a very small amount of energy after the first group key computation.

8. References

[1] David W. Carman, Peter S. Kruus, and Brian J.Matt. Constraints and approaches for distributed sensor network security. *NAI Labs Technical Report #00-010*, September 2000.

[2] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks, in *IEEE Symposium on Security and Privacy*, Berkeley, California, May 11-14 2003.

[3] Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen and Pramod Varshney. A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge. *To appear in IEEE INFOCOM*, 2004.

[4] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks, in *Proceedings of the 9th ACM conferenceon Computer and communications security*, Washington, DC, USA, 2002.

[5] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod Varshney. A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, Washington DC, October 27-31, 2003.

[6] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar. SPINS: Security protocols for sensor networks. *In Proceedings of Mobicom*, 2001.

[7] Michael Steiner, Gene Tsudik, Michael Waidner. Key Agreement in Dynamic Peer Groups. *IEEE Transactions on Parallel and Distributed Systems 11(8): 769-780*, 2000.

[8] A. Shamir. How to Share a Secret *Communications of the ACM, 22(11):612–613*, 1979.

[9] Whitfield Diffie and Martin E. Hellman. Privacy and authentication. An introduction to cryptography. *Proceedings of the IEEE*, 67(3):397–427, March 1979.

[10] William Stallings. Network Security Essentials Applications and Standards.