2013 International Conference on Parallel and Distributed Computing, Applications and Technologies

A New Efficient Random Key Revocation Protocol for Wireless Sensor Networks

Yi Jiang, Ruonan Zhang
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
Email: jiangyiv88@nwpu.edu.cn, rzhang@nwpu.edu.cn

Xiaojiang Du
College of Science and Technology
Temple University
Philadelphia, USA
Email: dxj@ieee.org

***Abstract*—In recent years, several random key pre-distribution schemes have been proposed for wireless sensor networks. However, the problem of key and node revocation has received fewer attentions. In this paper, we present a novel random key revocation protocol, which is suitable for large scale networks and removes compromised information efficiently. The proposed revocation protocol can guarantee network security and has less memory consumption and communication overhead. With the combination of centralized and distributed revocations, the protocol achieves both timeliness and accuracy for revocation. The simulation results show that our protocol has better performance than existing protocols in terms of increasing revocation validity and revocation velocity, and prolonging the network lifetime.**

***Keywords*-wireless sensor networks; key revocation;**

I. Introduction

Because sensor nodes have many limitations [1] in the capacity of communication, symmetric key cryptosystems and one-way function instead of asymmetric key cryptosystems are used in wireless sensor networks (WSNs). To ensure security for information sent among nodes, key management in WSNs is an important issue.

Key management includes two primary aspects: key distribution and key revocation. To generate keys for encryption successfully, several key pre-distribution schemes [2]-[3] have been proposed. However, key revocation protocol which refers to securely removing the keys and nodes that have been compromised has received fewer attentions. Eschenauer and Gligor [2] proposed a centralized revocation scheme which needs a base station to broadcast a revocation message to all of the sensor nodes. The scheme induces low speed revocation and large communication load for the base station. A distributed revocation scheme is put forward by Chan $et\ al.$ [3], which is suitable only for the random pairwise scheme. Eldefrawy $et\ al.$ [4] described a node revocation scheme based on PKC (Public Key Cryptography), which has some resource limits to be applied in WSNs. The group and session key revocation protocols are proposed in [5]-[7], which are unsuitable for random key pre-distribution schemes, because the communication key between two nodes is selected from the key pools randomly in random key pre-distribution schemes.

In this paper, we present a novel random key revocation protocol based on a random key pre-distribution scheme with node clustering in WSNs [8]. The revocation protocol is suitable for large scale networks and removes compromised information efficiently. Because the network topology has two-level hierarchy, we design the revocation protocol from two aspects: intra-cluster and inter-cluster. For intra-cluster, the revocation protocol is the combination of the centralized and distributed revocation with the advantages of timeliness and accuracy. For inter-cluster, the proposed revocation protocol can guarantee network security by using less memory consumption and communication load. In addition, we also propose a special method to set up secure links for any two clusters, which is another main contribution of the paper. The simulation results show that the proposed protocol can achieve better performance than the previous protocols in terms of revocation validity, revocation delay, and average energy consumption.

The remainder of the paper is organized as follows. Section II discusses our assumptions and the threat model. In Section III, we describe the proposed protocol from two aspects: intra-cluster and inter-cluster. Section IV analyzes the performance of our protocol. Finally, our concluding remark is exposed in section V.

II. Assumptions and Threat Model

In this paper, our assumptions are given in the following:

1) : Each node in the sub-region possesses u_i nodes which have the right to vote against it. Suppose only w nodes among the u_i nodes are needed to remove a node from the network. In order to transmit related information correctly, each node and its voting members share public key.

2) : Each node must store u_i root hash values, the hash values of $u_i \log_2 u_i$ brother path nodes, and the u_i votes, because it probably becomes a voting member to the u_i nodes. The contents above are used to verify the truth of vote.

3) : In order to authenticate other voting members, which have the voting right to the same node, each node must also store the IDs of the other voting members.

978-1-4799-2419-6/13 $31.00 © 2013 IEEE
DOI 10.1109/PDCAT.2013.43

4) : The cluster heads are special nodes, which usually have larger computation and communication capabilities than the other nodes within the cluster. It remembers the IDs and key rings of all nodes in the cluster.

5) : Each cluster head shares a private-key with KC (Key Center), denoted by k_{nx}, remembers all path node keys of the key management tree including their key chains and manages them by the root node key.

Our threat model is given below:

1) The node in a cluster is compromised:

- The attacker will obtain all the voting information of the compromised node.
- The attacker has the ability to distort the communication information sending to the other neighbor nodes.
- The attacker can vote against a valid node by pretending to be its voting member.

2) The cluster head node is compromised:

- Using the connection with other cluster head nodes, the attacker can transmit false information to other clusters and the nodes in its cluster.
- The corresponding path nodes keys of the key management tree will be compromised, which can threaten the connection between the other two clusters.

III. Cluster-Based Random Key Revocation Protocol

The centralized approach and the distributed approach are the two most popular key revocation approaches in recent years. We propose a new key revocation protocol, which combines the advantages of the two approaches and is suitable for large scale networks with hierarchical structure. Because the network topology is two-level hierarchical, the protocol is composed of two parts: intra-cluster and inter-cluster protocols.

A. Revocation Protocol in a Cluster

1) Preparation to Revocation: Before deployed in the sub-region $Z_{i(\eta)}$, every node has selected m_i keys to store in its key ring from the sub-key pool $S_{i(\eta)}$ [8]. To revoke nodes and keys which are compromised, a node must store some additive information which is described as follows.

a): Each node must store IDs of u_i nodes which have the right to vote against it. When two neighboring nodes set up a secure link with each other, they exchange their IDs. If the ID of one node belongs to the set of IDs stored in other node, it becomes a voting member of that node with valid voting right.

b): Each vote is a special code. The u_i votes which vote against the same node are calculated by a hash function, the outcome of which generates a Merkle tree [9]. Each of the u_i nodes uses the root hash value and the hash values of the $\log_2 u_i$ brother path nodes from the root to a leaf to authenticate a vote corresponding to the leaf node.

c): The memory consumption to every node for voting can be calculated by (1), and the space complexity is O($u_i \log_2 u_i$).

$$S_{mer} = \{u_i^2(ID) + u_i(root) + u_i \log_2 u_i(path) + u_i(vote)\} \quad (1)$$

2) Vote Mechanism: To describe the vote mechanism, we use Fig.1 as an example.

a): To prevent widespread release of the revocation keys by the compromised nodes, we require that only the voting members of node A have the right to vote against A. Each vote is encrypted by A using k_A, and can be used only by the decrypting of A, which inhsures a direct connection with A.

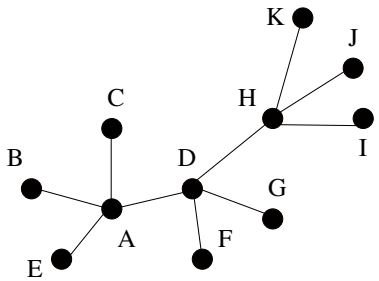

Figure 1. Voting relationship among nodes

As shown in Fig. 1, after the shared-key discovery phase, nodes B, C, D and E become voting members of node A, and they have the right to vote. When the vote begins, A sends k_A to B, C, D and E to decrypt their votes. Without loss of generality, we suppose that node D wants to vote against A, D broadcasts its vote k_D and path hash values after encrypted by k_A, as depicted in Fig.2.

b): When the voting members of A receive the message, they will calculate the information stored from the message, and compare the results with the root hash value stored in their memory. If they are equal, the voting members update flags to indicate that A is voted once. If a flag indicates w votes, the voting member will send this information to its cluster head node.

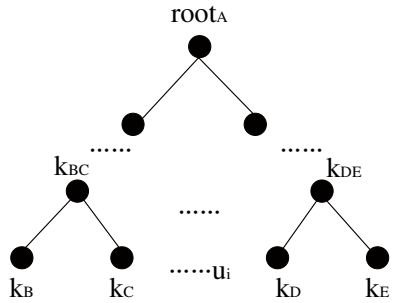

Figure 2. Voting Merkle tree

For example, as shown in Fig. 2, B, C and E receive the message from D. They find out that D is a voting member of

A like themselves; then, they compute the message to prove validity of the vote and update their flags. The computation process is denoted by (2), where $H(\ )$ is a hash function, which is used to compute $root_A$ by vote k_D and path hash values from D. $|$ is the connection symbol, which is used to adjoin two neighbor path hash values.

$$root_A = H(H(\cdots H(H(k_D)|H(k_E))))(2)$$

c): A voting member will broadcast a vote message every Δt time to ensure that the vote message can be sent to the destination. If other voting members receive the vote message, they will reply to it, so the broadcast will be over.

We prescribe that the revocation decision and execution occur within a bounded time period $\Delta d(\Delta d >> \Delta t)$. The whole life of the network is divided into a large number of time periods. Within every period Δd, if there is insufficient number of sensor nodes agreeing that a node is to be revoked, then the revocation decision returns to a negative result, which avoids the erroneous votes that can accumulate over the networks lifetime and result in the revocation of a legitimate node.

3) Revocation Mechanism: The revocation mechanism is the last step of our scheme to revoke nodes in a cluster. If a voting member finds that its vote flag of node A comes up to w, it will cut off the link with A and inform the cluster head.

If the cluster head confirms node A as being compromised, it will broadcast a single revocation message containing a signed list of m_i key IDs of the key ring to be revoked and the compromised node ID to the nodes in seven hexagonal sub-regions related to node A. Each node that can decrypt the message using the public cluster key, removes the link with node A, and checks whether its key ring includes the revoked keys. If the revoked keys are included, the compromised keys are removed.

4) Performance Analysis: In a cluster, the revocation protocol combines centralized revocation and distributed revocation. At first, nodes distribute their votes and deal with the compromised links by themselves which has the advantage of good timeliness. Then the cluster head node can remove captured keys if necessary, which has good accuracy and can prevent adversary from extending invasion.

B. Revocation Protocol between Clusters

1) Setting up Communication between Any Two Cluster Heads: To connect all cluster heads efficiently, we set up a hierarchical structure of symmetric keys, called a $key\ management\ tree$. As depicted in Fig.3, the tree has 2 dimensions. However, the dimension of the tree can be changed at will.

Each leaf in the tree is a cluster head which is managed by its father node. The father node is managed by its father, so a key management tree is established. All father nodes are called $path\ nodes$, except the root node, and denote the path node keys and the root node key by $k_{\mathrm{path}x}$ and k_{root}, respectively.

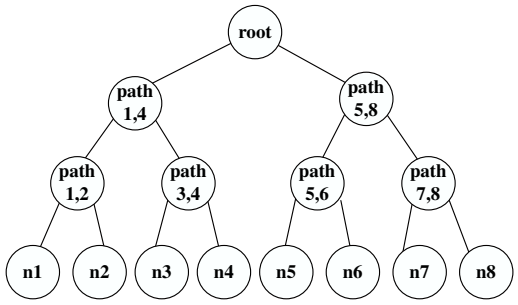

Figure 3. Key management tree

The key management tree can reduce the communication load of KC. All cluster head nodes can set up communication links by themselves. Each leaf node stores a set of keys of the path nodes which lie on the path from the root to itself, and all keys of the path nodes possess their own ID. When the communication setup phase begins, each cluster head broadcasts the IDs of the stored keys which are called $ID\text{-}Ring$ to the other nodes. The process of negotiating communication key between two neighboring heads can be realized in the following steps.

a): If two neighboring nodes have the same stair father node, they can communicate with each other. They will encrypt a link key by the stair father node key and exchange it within the scope of the transmission radius.

b): There are some neighboring nodes with different stair fathers which have not set up links with each other. We will search the same secondary father node for themselves. They will encrypt a link key by the secondary level father node key and exchange it within the radius of transmission to communicate with each other.

c): This method can be used similarly to the root node, until all neighboring nodes are connected with others.

Taking Fig.3 for instance, we depict the concrete process of setting up communication links as follows.

a): If n_3 and n_4 are neighbors: To connect with each other, they first broadcast their $ID\text{-}Ring$ within the range of transmission, which are

$$\mathrm{ID-Ring}_3 = \{n_3, \mathrm{ID}_{\mathrm{path3,4}}, \mathrm{ID}_{\mathrm{path1,4}}, \mathrm{ID}_{\mathrm{root}}\},$$

$$\mathrm{ID-Ring}_4 = \{n_4, \mathrm{ID}_{\mathrm{path3,4}}, \mathrm{ID}_{\mathrm{path1,4}}, \mathrm{ID}_{\mathrm{root}}\}.$$

We can see that n_3 and n_4 have the same stair father node $\mathrm{ID}_{\mathrm{path3,4}}$, so n_3 can use $k_{\mathrm{path3,4}}$ to encrypt their communication key $k_{3,4}$ and transmit it to n_4.Then they can use $k_{3,4}$ to encrypt the following data.

b): If n_1 and n_4 are neighbors: To connect with each other, they broadcast their $ID\text{-}Ring$ within the range of transmission, which are

$$\mathrm{ID-Ring}_1 = \{n_1, \mathrm{ID}_{\mathrm{path1,2}}, \mathrm{ID}_{\mathrm{path1,4}}, \mathrm{ID}_{\mathrm{root}}\},$$

$$\mathrm{ID}-\mathrm{Ring}_4=\{n_4,\mathrm{ID}_{\mathrm{path3,4}},\mathrm{ID}_{\mathrm{path1,4}},\mathrm{ID}_{\mathrm{root}}\}.$$

We can see that n_1 and n_4 have the same secondary father node $\mathrm{ID}_{\mathrm{path1,4}}$, so n_1 can use $k_{\mathrm{path1,4}}$ to encrypt their communication key $k_{1,4}$ and transmit it to n_4. Then they can use $k_{1,4}$ to encrypt the following data.

c): If n_2 and n_5 are neighbors: To connect with each other, they broadcast their $ID\text{-}Ring$ within the range of transmission, which are

$$\mathrm{ID}-\mathrm{Ring}_2=\{n_2,\mathrm{ID}_{\mathrm{path1,2}},\mathrm{ID}_{\mathrm{path1,4}},\mathrm{ID}_{\mathrm{root}}\},$$

$$\mathrm{ID}-\mathrm{Ring}_5=\{n_5,\mathrm{ID}_{\mathrm{path5,6}},\mathrm{ID}_{\mathrm{path5,8}},\mathrm{ID}_{\mathrm{root}}\}.$$

We can see that n_2 and n_5 have different father nodes except for the root, so n_2 can use k_{root} to encrypt their communication key $k_{2,5}$ and transmit it to n_5. Then they can use $k_{2,5}$ to encrypt the following data.

All the neighboring nodes can establish connections with each other by themselves. Although this method increases the memory consumption and communication load for each node, it can reduce the burden of KC greatly and can be used to renew the compromised keys as described in the next section.

2) Renewing Keys Stored in the Cluster Head: In this section, we will discuss how to insure security when some heads are captured.

a) Key chain: To avoid being captured by adversaries, we assume that every node key is composed of a key chain. We must change the compromised keys after the capture action has happened. The nodes in the tree are able to authenticate a renew message and use it for their new node key. We use a hash function $H(\ )$ to form a node key chain, which is expressed in (3). The received key is the correct one if the result of the key computed by $H(\ )$ is equal to the former one.

$$k_i=H(k_{i+1}),\qquad 0\le i\le n-1 \quad (3)$$

b) Renewing compromised keys: Every cluster head stores a set of keys of the current path nodes from the root to itself, called $Set\text{-}Key$. If a head is compromised, other nodes connected with it will remove the links. All keys belonging to the heads $Set\text{-}Key$ are captured, so we must renew them to the other nodes which can use the keys to set up new links. These keys must be securely sent to the related nodes, except for the compromised node. We assume that $patha$ is a path node lying on the path from the root to the compromised node, and $pathb$ is its child node. The principle of renewing keys is as follows:

- If $pathb$ is a leaf node, KC will use a private key with $pathb$ to encrypt the next key of $patha$, and then send it to $pathb$. The method is the same for the other child nodes of $patha$ except for the compromised node.
- If $pathb$ is an internal tree node and belongs to the compromised path, KC will use the next key of $pathb$ to encrypt the next key of $patha$, and then send it to the corresponding child leaf nodes of $pathb$ except for the compromised node.
- If $pathb$ is an internal tree node but dose not belong to the compromised path, KC will use a current key of $pathb$ to encrypt the next key of $patha$, and then send it to the corresponding child leaf nodes of $pathb$.

Taking Fig. 3 for instance, we depict the detailed process of renewing compromised keys as follows. If n_5 has been captured, its ID-Ring includes $ID_{\mathrm{path5,6}}$, $ID_{\mathrm{path5,8}}$ and ID_{root}, so KC must renew the keys of these path nodes for each corresponding head node.

- For $path5,6$, its child node is leaf n_5 and leaf n_6. n_5 is compromised, so KC will send the next key of $path5,6$ to $n_6(E_{k_6}\{k_{\mathrm{next5,6}}\})$.
- For $path5,8$, its child node is $path5,6$ and $path7,8$. $path5,6$ accords with (2), so KC will send the message $(E_{k_{\mathrm{next5,6}}}\{k_{\mathrm{next5,8}}\})$ to $path5,6$'s child node n_6. $path7,8$ accords with (3), so KC will send the message $(E_{k_{cur7,8}}\{k_{next5,8}\})$ to $path7,8$'s child nodes n_7 and n_8.
- For root, its child node is $path1,4$ and $path5,8$. $path1,4$ accords with (3), so KC will send the message $(E_{k_{\mathrm{cur1,4}}}\{k_{\mathrm{root(next)}}\})$ to $path1,4$'s child leaf nodes $n_1 \sim n_4$. $path5,8$ accords with (2), so KC will send the message $(E_{k_{\mathrm{next5,8}}}\{k_{\mathrm{root(next)}}\})$ to $path5,8$'s child leaf nodes $n_6 \sim n_8$.

Every leaf node can use its corresponding key to decrypt the received message, and use $H(\ )$ to authenticate whether the new key is the correct one for the key chain.

c) Revoking communication to compromised nodes: When a compromised node is removed from the whole tree, all head nodes previously connected with it must detach links with it. Because revocation can induce some nodes to have no affiliation to their neighboring nodes, the setting up communication phase will restart. They use new path keys to establish new communication links, and generate the new link-keys. This process is accomplished automatically, unrelated to KC.

3) Performance Analysis: Among clusters, all of the heads can establish connections by storing a set of keys of path nodes, independent from KC. Since remembering all node keys to set up connections is avoided, the storage consumption is low. The scheme of renewing keys stored in the cluster node head can improve performance by reducing the complexity and the number of re-keying messages. For the complexity of re-keying messages, we use the symmetry encryption algorithm which has lower complexity than a digital signature, and our key management tree has better secure performance than the dissymmetrical encryption algorithm. As to the number of re-keying messages, KC only needs to broadcast $\sum_{i=1}^{\log_{\dim}\gamma}(\dim^i-1)$ messages (where dim is the

dimension of tree, γ is the number of clusters), instead of $\gamma \log_{\dim} \gamma$ messages, to set up a connected network, which can reduce the communication load greatly.

IV. Simulation and Analysis

In this section, we compare our revocation protocol with the basic revocation protocol [2] and the pairwise key revocation protocol [3] in terms of increasing revocation validity and velocity and prolonging the lifetime of the network. The performance of these protocols is simulated in NS2 for the region with 1000×1000 square units.

A. Revocation Validity

When nodes are compromised, we can use the revocation protocol to ensure the normal operation of the network. The revocation validity is measured by packet loss rate. The serviceability and extendibility of the three revocation protocols are discussed. The number of nodes in the network is from 1000 to 5000, and the 802.11 MAC protocol with 1Mbps is used. If there are fifty compromised nodes which are located randomly in different places and can not forward the normal packets, we will execute revocation protocols. By sending 1000 data packages, the packet loss rate is simulated and Fig.4 shows the simulation results of the average of 100 trials.

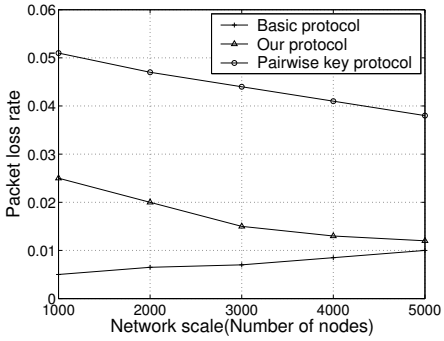

Figure 4. Packet loss rate with the variety of network scale

As shown in Fig.4, we can see that the packet loss rate of the basic protocol is the smallest. This is because the basic protocol is a centralized revocation method based on the control node. It has the strengths of timeliness and veracity. With the increased network scale, we can see that the curve rises slowly. This is because more nodes need to be controlled, so the revocation effect is slightly low. The packet loss rate of the pairwise key protocol which belongs to the distributed revocation method is higher than that of the basic protocol. Despite halfway revocation, this protocol has good revocation effect and speed in partial region. The compromised nodes are arranged more sparsely with the increased network scale, which has little effect on packets, and thus the loss rate declines. Our protocol integrates the advantages of the centralized method and the distributed method, referring to the issue of key update to the cluster node, and its loss rate performance is in the middle. For large network scale, our methods efficiency is close to that of the basic protocol.

B. Revocation Delay

The revocation delay is defined as the time interval from the moment when the revocation starts to the time when the uncompromised correlative nodes receive revocation messages. The number of nodes in the network is from 1000 to 5000 and the 802.11 MAC protocol with 1Mbps data rate is used. Suppose there is a compromised node whose position is uncertain. We simulate and compare the revocation delay of the three revocation protocols. As show in Fig.5, the simulation results are the average of 100 trials.

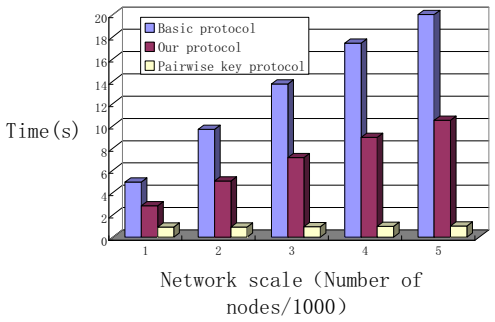

Figure 5. Revocation delay with the variety of network scale

We can observe from Fig.5 that the revocation delay of the basic protocol is the longest and gets longer with increasing network scale. This is because the basic protocol is a centralized revocation method in which the control node broadcasts revocation messages to every node belonging to its scope of management, so the revocation speed is slow, and the revocation delay is longer with increased node number. The pairwise key protocol adopts the distributed revocation method which has better revocation effect in partial region. The impact of network scale on it is small. Because the revocation is done by the nodes themselves, the revocation speed is fast. Our protocol uses the distributed revocation to the partial region and the centralized revocation to the whole region. Based on a random key pre-distribution scheme which sets up sub-key pools, the revocation is performed to the correlation region only, and the broadcast scope of revocation messages and delay are smaller than those for the basic protocol, but are higher than the pairwise key protocol.

C. Average Energy Consumption

To evaluate the lifetime of the network, the average energy consumption to revoke a single node using revocation protocols is examined. Suppose that the energy of every node is $1J$, and the energy consumption of sending or receiving

one packet is $0.0001J$. The simulation results are shown in Fig.6, where we can see that the energy consumption in the pairwise key protocol is the smallest. This is because it only revokes partially and does not broadcast the revocation massages to the whole network. With the increased network scale, the number of nodes rises, and it leads to the decline in average energy consumption. Therefore, the average energy consumption of the basic protocol is the biggest and that of our protocol, which combines the centralized and the distributed revocation, is in the middle. Because centralized KC revocation is closely related to the corresponding region instead of the larger scope. The average energy consumption of our protocol declines with the increased network scale.

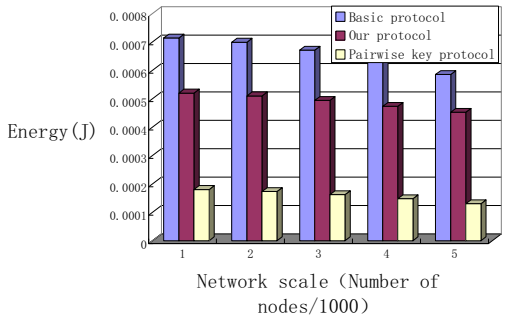

Figure 6. Average energy consumption with the variety of network scale

V. Conclusion

We describe a novel random key revocation protocol, which is suitable for large scale networks and useful to remove the compromised secret information. According to the analysis of performance, for intra-cluster, the revocation protocol is a combination of the centralized and the distributed revocation with the advantages of timeliness and veracity, and for inter-cluster, the revocation protocol can guarantee network security with less storage consumption and communication load. Furthermore, a method to set up a secure link between any two cluster nodes using a key management tree is also proposed. The simulation results show that our protocol can achieve better performance than previous protocols in revocation validity, faster revocation speed and better balanced ability to node energy consumption. Based on the comprehensive analysis and comparison, our protocol is an optimum one.

Acknowledgment

The authors acknowledge support from the Nature Science Foundation of China (61301092, 61202394) and the Nature Science Foundation of Shanxi Province of China (2012JQ8005). The authors would also like to thank all of the anonymous reviewers for their comments and suggestions.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, *A Survey on Sensor Network*, IEEE Communications Magazine. vol. 40, no. 8, August 2002, pp. 102-114.

[2] L. Eschenauer and V.D. Gligor, *A Key-management Scheme for Distributed Sensor Networks*, in Proc. of the 9th ACM Conference on Computer and Communications Security. Washington, DC, USA, November18-22 2002, pp. 41-47.

[3] H. Chan, A. Perrig and D. Song, *Random Key Predistribution Schemes for Sensor Networks*, in Proc. of the IEEE Symposium on Security and Privacy. Berkeley, California, May 11-14 2003, pp. 197-213.

[4] M.H. Eldefrawy, M.K. Khan and K. Alghathbar, *A Key Agreement Algorithm With Rekeying for Wireless Sensor Networks Using Public Key Cryptography*, in Proc. of the International Conference Anti-Counterfeiting Security and Identification in Communication (ASID). 2010, pp: 1-6.

[5] M. Rahman, S. Sampalli and S. Hussain, *A Robust Pair-wise and Group Key Management Protocol for Wireless Sensor Network*, in Proc. of the 2010 GLOBECOM Workshops. pp. 1528-1532.

[6] Q. Wang, *Constant Storage Self-Healing Group Key Distribution for Large-Scale Wireless Multimedia Sensor Networks*, in Proc. of the 2010 International Conference on Multimedia Technology (ICMT). pp. 1-4.

[7] C. Park, Y. Zhang, I. Kim and M. Park, *DLS: Dynamic Level Session Key Revocation Protocol for Wireless Sensor Networks*, Information Science and Applications (ICISA). 2010, pp. 1-8.

[8] Y. Jiang and H. Shi, *A Key Pre-distribution Scheme for Wireless Sensor Networks Using Hexagonal Deployment Knowledge*, Chinese Journal of Electronics. 2008, 17(3:520-525.

[9] R. Merkle, *Protocols for Public Key Cryptosystems*, in Proc. of the IEEE Symposium on Research in Security and Privacy. Apr. 1980.