Efficient Key Agreement for Merging Clusters in Ad-Hoc Networking Environments*

Sooyeon Shin and Taekyoung Kwon

Information Security and Computer Networks Lab.,
Sejong University, Seoul 143-747, Korea
shinsy80@sju.ac.kr, tkwon@sejong.ac.kr

Abstract. In this paper, we study a simple scheme that can effectively deal with merging different adjacent clusters in ad-hoc networks. When nodes of each cluster have already agreed on their own group keys and intend to merge themselves for further secure communications, our scheme can be used in an efficient and secure way.

1 Introduction

Ad-hoc networks consist of mobile nodes without any underlying infrastructure and are also referred to as MANETs (Mobile Ad-hoc NETworks) [4]. Each node should perform a function of router to transmit data to each other in the absence of infrastructure. Additionally, mobile nodes are limited by a range of radio coverage and have an irregular source of power because of using a battery. MANET Working group in IETF (Internet Engineering Task Force) works for standardization of such ad-hoc networks and mainly decides standards of routing protocols. In ad-hoc networks, there could be more than one group of nodes, while the group is occasionally regarded as a cluster.

Contributions of Our Scheme. In this paper, we propose a simple key agreement scheme for merging clusters. We consider that two different groups intend to merge themselves under their respective group keys. Our study can trivially be extended to n groups. In other words, two groups having different shared secrets such as passwords (e.g. PW_A and PW_B) and group keys (e.g. K_A and K_B) want to communicate together by merging clusters. For the purpose, two groups should agree on a new group key or new session key because they have different keys at merging themselves. In general, two ways can be considered for this; 1) to agree on a new group key, and 2) to reuse the established group key which is used before merging. The first may be expensive with regard to computation and communication costs depending on a group key agreement protocol. For example, performing arithmetic operations for group key agreement will be increased exponentially according to the increasing number of nodes. The second is more

* This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 996–1005, 2005.

efficient than the first one but special care must be taken with respect to security concerns. Therefore, we focus on the second way in order for an efficient method of group key agreement for merging two different groups in ad-hoc networks. We divide our protocol into two parts; GKA (Group Key Agreement) protocol in each cluster and MCP (Merging Cluster Protocol) between two clusters.

The rest of this paper is organized as follows. In Section 2, we describe two essential tools for our protocol; the basic and essential routing protocol, and Group Diffie Hellman Key Agreement. In Section 3, we describes the overview of GKA protocol and MCP and in Section 4, we describes analysis of the proposal protocol. In Section 5, we present our conclusions.

2 Related Work

We need two basic tools for designing our protocol; ad-hoc routing protocols, and group Diffie-Hellman key exchange. Ad-hoc routing protocols are basic and necessary for our protocol since our protocol should be executed in ad-hoc networks. Moreover, the group Diffie-Hellman is suitable for group key agreement. Especially, we use the Bresson's scheme et al. [3] because it is provably secure in the random oracle model. In the following subsections, we describe these basic tools.

2.1 Ad-Hoc Routing Protocols

There are three different types of routing protocols in ad-hoc networks; table-based routing protocol, on-demand routing protocol and hybrid routing protocol. Proactive or Table-based routing protocol always updates and keeps tables, storing all of the path setting information from nodes to nodes by performing a constant path search work. DSDV (Destination Sequenced Distance Vector) [11] and OLSR (Optimized Link State Routing) [8] are typical examples of Table-based routing protocol. Reactive or On-Demand routing protocol updates path setting information when there is a request from a start node. The examples of On-Demand routing protocol are DSR (Dynamic Source Routing) [10] and AODV (Ad-hoc On-demand Distance Vector routing) [12]. Last but not least, Hybrid routing protocol combines the advantages of both table-based and on-demand routing protocol. For example, there are ZRP (Zone Routing Protocol) [6] and CBRP (Cluster-Based Routing Protocol) [9]. Moreover, various protocols are provided and studied for improving performance of routing protocols. There is no guarantee that a path between two nodes would be free of malicious nodes because of the absence of infrastructure and the consequent absence of authorization facilities. Thus, messages transmitted over such path can come under various attacks such as eavesdropping, altering, impersonation and routing disruption, and secure routing is necessary. Most above routing protocols do not have considered security despite that ad-hoc networks are more susceptible to routing attacks. Secure routing has been studied in various research. The representative ones are SAR (Security-Aware Routing protocol) [13] and Ariadne [7]. According to applications, secure routing protocols can be applied to our scheme.

2.2 Group Diffie-Hellman Key Agreement

In this section, we describe an essential tool, a Diffie-Hellman protocol. We use an extended Diffie-Hellman protocol for GKA in each cluster. This extended version is a password-based group Diffie-Hellman key exchange protocol presented by Bresson et al. [3].

Group Diffie-Hellman. Diffie and Hellman provided the first practical scheme of public-key cryptography in 1976. The scheme provided a method whereby two principals communicating over an insecure network can agree on a secret key in [5]. The basic concept of Diffie-Hellman protocol is that two principals pick at random values x_1, x_2 and exchange the values g^{x_1}, g^{x_2} in a finite cyclic group over a network, and each principal computes Diffie-Hellman secret value $g^{x_1 x_2}$ using g^{x_2} (respectively, g^{x_1}) received from the other principal.

A Password-Based Group Diffie-Hellman Key Exchange. Several 2-party Diffie-Hellman key exchange protocols are aimed to distribute a session key among two principals when the principals share a password. Bellare et al. [1] presented a formal model for this problem. Then, Bresson et al. extended the famous EKE (Encrypted Key Exchange) to multi-party setting and proved its security [3]. This scheme will be referred to as the Bresson's scheme in the rest of this paper. We use the Bresson's scheme for Group Key Agreement (GKA) protocol because it is a provably secure password-based group Diffie-Hellman key exchange protocol. The followings are summaries of bases and assumptions of EKE protocol in the Bresson's scheme. In the Bresson's scheme, security parameters, l_1 and l_2 are defined and the arithmetic assumed is in a finite cyclic group $\mathbb{G} =< g >$ of order a l_1-bit prime number q. We then use a hash function $\mathcal{H}$ from $\{0,1\}^*$ to $\{0,1\}^{l_2}$ and consider several block ciphers, depending on the size of the input. For each integer $i \geq 2$, we define two families $\mathcal{E}^i = \{\mathcal{E}_k^i\}$ and $\mathcal{E}'^i = \{\mathcal{E}_k'^i\}$ where $k \in$Password. The inverse of $\mathcal{E}_k^i$ (respectively, $\mathcal{E}_k'^i$) is denoted $\mathcal{D}_k^i$ (respectively, $\mathcal{D}_k'^i$). This Password is a small dictionary of size N and nodes of the inner cluster share a low-entropy secret pw taken from this dictionary. Such encryption schemes can be instantiated with CBC mode so that each part of the plaintext depends on the entire ciphertext. Operators Φ and Φ' hide away exponent parts and are used for more secure group key exchange. They are defined formally in [3].

3 Merging Clusters Using Group Key Agreement in Ad-Hoc Networks

Mobile nodes are divided into several duplicated or separated clusters according to certain radio coverage for making ad-hoc groups from various nodes in ad-hoc networks. Figure 1(a) shows how nodes of each cluster in ad-hoc networks are divided into two groups, cluster A and B. We postulate the mobile nodes in each cluster share the same password pre-loaded from the cluster-head. There could be several ways for loading the shared password to each cluster under the control

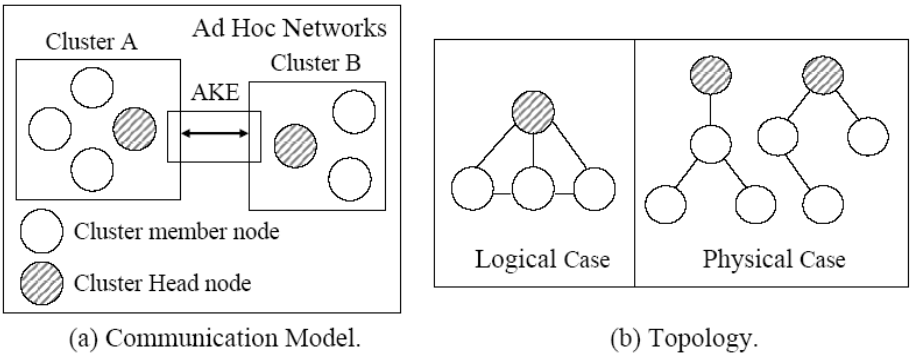

Fig. 1. Basic Concept

of the cluster-head or possible authority. This issue is out of scope in this paper. We assume that nodes of each cluster have already agreed on their own group key through GKA protocol and thus, two different group keys are maintained in ad-hoc networks. These group keys will be referred to as the established group key in the rest of paper. In this model, finally we assume two clusters intend to merge themselves.

3.1 Assumptions

Topology of the Inner Cluster. We divide a connection between nodes into a logical case and physical case since nodes have various topologies in ad-hoc networks. Figure 1(b) shows a logical case and physical case of topology. In a physical case, we assume that values and messages are transmitted via an adjacent different node (a neighbor node) during generating the inner group key and transmitting messages.

Assumptions for GKA Protocol. Nodes share a password PW_A (respectively, PW_B) for cluster A (respectively, cluster B). We also assume that a cluster-head is a victim node since it consumes more power than others. It is possible to play a role of cluster-head among nodes by turns or voluntarily since the nodes trust each other in this stage, but we do not focus on this issue. We rather assume that an adjacent node, the closest node to a different cluster, becomes a cluster-head. Additionally, we assume that two ad-hoc groups use the same hash function, such as SHA 1 or MD 5, used to compute session keys and MAC (Message Authentication Code) for secure routing.

Assumptions for MCP. We define a pseudo-random function PRF. A PRF is a deterministic function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ which is efficient and takes two inputs $x, k \in \{0,1\}^n$. Now, we only consider x to be a variable and let k be a hidden random seed and function index, $f(x,k) = f_k(x) = \mathcal{F}$. We assume that it is difficult to distinguish $g^{\mathcal{F}(x)}$ from g^x in a cyclic group of prime order q under

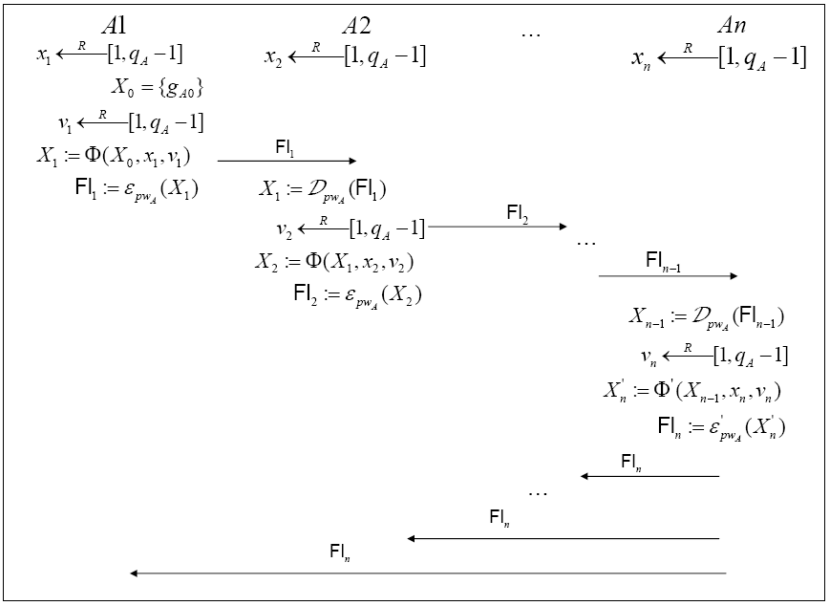

Fig. 2. GKA Protocol of Cluster A: It is based on the Bresson's scheme et al

randomness of $\mathcal{F}$. In order for merging clusters, some authentication methods may be necessary between two adjacent cluster-heads for AKE (Authentication Key Exchange) in Figure 1(a). There can be at least three methods. First, they can use PKI if they are connected with an external gateway. Secondly, they can share the secret key through physical meeting each other. Thirdly, it is possible for them to be located in the same radio coverage(e.g. Cluster C). Thus, they can share the same password.

3.2 Group Key Agreement Protocol (GKA) in Each Cluster

In this paper, GKA protocol will perform more than twice; for cluster A, cluster B and merging clusters.

GKA of Cluster A. Figure 2 describes GKA protocol of cluster A and the flows (Fl) are encrypted under password PW_A. Nodes are $\mathcal{U} = \{A1, \cdots, An\}$. The session key space SK associated to this protocol is equipped with a uniform distribution. This protocol divides into two stages; Flow 1 ($\rightarrow$) is encrypted by $\mathcal{E}$ and decrypted by $\mathcal{D}$, and Flow 2 ($\leftarrow$) is encrypted by $\mathcal{E}'$ and decrypted by $\mathcal{D}'$.

1. Flow 1.
   (a) We assume that a node, which wants to agree on a group key at the beginning, is $A1$ without considering topology of nodes in ad-hoc networks. We then assume that a node An, that receives the flow finally, is

a cluster-head. $A1$ just receives Fl_0 ="start", and builds $X_o = \{g_{A_0}\}$, where g_0 is a random element in $\mathbb{G}$

(b) A node Ai $(1 \leq i \leq n)$ decrypts ciphertext $Fl_{i-1} \in \bar{\mathbb{G}}$ received from a previous node using $\mathcal{D}_{PW_A}$ and puts it into the plaintext $X_{i-1} \in \bar{\mathbb{G}}^i$. .

(c) Then, a node Ai picks at random two (private) values (x_i, ν_i) in $\mathbb{Z}_q^*$ and gets $X_i := \Phi(X_{i-1}, x_i, \nu_i) \in \bar{\mathbb{G}}^{i+1}$ from the plaintext X_{i-1} according to the operator Φ.

(d) Finally, a node Ai encrypts the value X_i using $\mathcal{E}_{PW_A}$ and transmits ciphertext Fl_i to the next node in cluster A.

2. Flow 2: It starts when the last node An (a cluster-head) receive the last flow $Fl_{n-1} \in \bar{\mathbb{G}}^n$.

(a) The node An decrypts the last flow received from a previous node $A(n-1)$ using $\mathcal{D}_{PW_A}$ and put it into the plaintext $X_{n-1} \in \bar{\mathbb{G}}^n$.

(b) The node An then picks at random two (private) values (x_n, ν_n) in $\mathbb{Z}_q^*$ and gets $X_n^{'} := \Phi^{'}(X_{n-1}, x_n, \nu_n) \in \bar{\mathbb{G}}^n$ from the plaintext X_{n-1} according to the operator $\Phi^{'}$.

(c) Finally, the node An encrypts value $X_n^{'}$ using $\mathcal{E}_{PW_A}^{'}$ and broadcasts the ciphertext Fl_n.

After receiving the flow Fl_n, each node decrypts Fl_n received from An (a cluster-head) using $\mathcal{D}_{PW_A}^{'}$ and put it into the plaintext $X_n^{'} : X_n^{'} := \mathcal{D}_{PW_A}^{'}(Fl_n) = \{\alpha_1, \cdots, \alpha_n\}$. Finally, each node can compute the group key K_A which is shared between nodes in cluster A and generates the session key sk_A using a hash function.

$$K_A = (\alpha_i)^{x_i} = g_n^{x_1 \cdots x_n} \quad (g_n = g_{A0}^{\nu_1 \cdots \nu_n}) \tag{1}$$

$$sk_A = \mathcal{H}(\mathcal{U}||Fl_n||K_A) \tag{2}$$

GKA of Cluster B. GKA protocol of cluster B is the same as GKA protocol of cluster A except that a generator g is different $(g_{B0} \neq g_{A0})$ since a group $\mathbb{G}$, which is used by nodes of cluster B, is different. Also, nodes of cluster B share a different password from a password of cluster A. $(PW_B \neq PW_A)$ Conclusively, cluster B shares a different group key (K_B) from cluster A. Also, cluster B can generate the session key sk_B using K_B and a hash function.

3.3 Merging Clusters Between Cluster A and B

If the Bresson's scheme [3] is used for merging clusters, performing arithmetic operations will be increased exponentially according to the increasing number of nodes. Therefore, we provide an efficient way to merge clusters in this section.

MCP. In Section 3.1, we assumed that AKE is necessary. Thus, all messages transmitted in step 2, 3, and 4 are secure because of AKE and secure routing. We also assume that all values transmitted in cluster A or cluster B, are secure since they are encrypted by the established group key as the new session key.

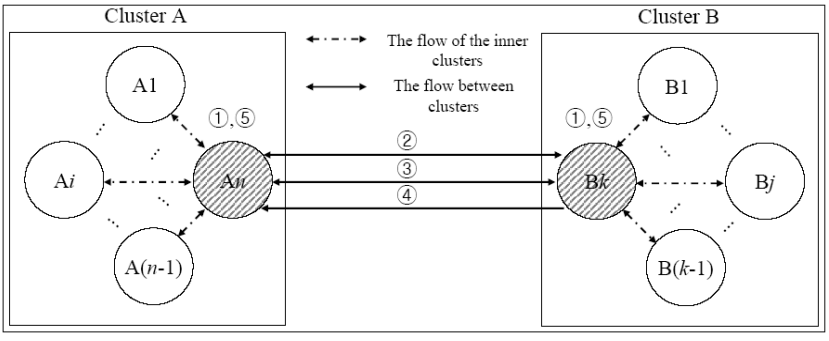

Fig. 3. Protocol for merging clusters

Figure 3 shows flows between cluster A and cluster B for MCP. We assume that the number of nodes in cluster A is n and the number of nodes in cluster B is k. Both $n \neq k$ and $n = k$ are possible. Before merging clusters, mutual agreement steps of domain parameters (e.g. g_{A0}, p_A and q_A) are needed in order to generate a new group key. A mutual agreement of domain parameters is performed by choosing majority in the next Section. They are step 1, step 2 and step 3.

1. Each cluster-head sends the number of nodes in own cluster to another cluster-head.
2. Each cluster-head randomly requests routing information to nodes in another cluster in order to determine which cluster has more nodes than another. Each cluster-head then computes the number of nodes in another cluster through routing information in step 1 and reply from nodes in another cluster and compares it with own number of nodes.
3. A cluster-head, which has more nodes than another cluster-head, sends domain parameters that was used when they generated the established group key in own cluster. (g_{A0}, p_A, q_A or g_{B0}, p_B, q_B)
4. Each cluster-head picks a random variable x (respectively, variable y) which is variable for PRF. It then computes a pseudo-random value $\mathcal{F}_A$ (respectively, $\mathcal{F}_B$)by using the established group key K_A (respectively, K_B) which is generated in own cluster and a random variable x (respectively, variable y): for cluster A, $f(x, K_A) = f_{K_A}(x) = \mathcal{F}_A$ and for cluster B, $f(y, K_B) = f_{K_B}(y) = \mathcal{F}_B$. They exchange $\mathcal{F}_A$ and $\mathcal{F}_B$.
5. Each cluster-head broadcasts a value ($[\mathcal{F}_A, x, g_{A0}]_{K_A}$, or $[\mathcal{F}_B, y, g_{B0}]_{K_B}$) is received from another cluster-head, to member nodes.

Finally, all nodes can compute own pseudo-random value and generate a new group key using it and another one which is received from a cluster-head.

$$K_{AB} = g^{\mathcal{F}_A \mathcal{F}_B} \quad \text{where} \quad g = g_{A0} \quad \text{or} \quad g = g_{B0}. \tag{3}$$

Two clusters can create new session using a new group key K_{AB}. Consequently, two clusters merge.

Application of Majority. The application of majority is in order to offer fairness by choosing majority during merging clusters. Two clusters get routing information of nodes in another cluster for getting the number of nodes in the other cluster. In step 1 and 2 of Figure 3, majority is determined. Since our protocol depends on secure routing protocols, both cluster-heads can get routing information and know majority. Except that all nodes in another cluster try to deceive the number of nodes in own cluster, it is impossible for cluster-head to cheat another cluster-head because of secure routing which provides integrity. We note that the number of nodes in both clusters are the same. We assume that majority is determined by coin flipping in this case.

4 Analysis

In this section, we discuss an analysis of our protocol. In Section 4.1, we present security requirements and how our protocol can provide security. In Section 4.2, we describe efficiency of our protocol.

4.1 Security

Ad-hoc networks are more vulnerable than typical networks because of some reasons. Attackers can easily intercept messages transmitted over insecure channels since there is no fixed infrastructure and mobile nodes are wireless. Since these nodes use a power of low battery, they can not use strong cryptographic techniques. Therefore, security is important in ad-hoc networks and also for our protocol. We describe essential security requirements; confidentiality, authentication, integrity and forward secrecy and how our protocol provide these security in the followings.

Confidentiality. Confidentiality is the process of keeping the information sent unreadable to unauthorized nodes. To protect information transmitted over insecure channels in ad-hoc networks is necessary since information is more available not only to its intended nodes but also to eavesdroppers. One way to reach confidentiality is the use of cryptographic techniques. Encryption is a simple cryptographic technique in order to provide confidentiality. In both GKA protocol and MCP, all messages are encrypted by keys. Therefore, our protocol assures confidentiality. Semantic security, which guarantees that an adversary can not distinguish a secret value from random value. Our protocol also provides semantic security since a basic scheme of key agreement is based on Group Diffie-Hellman assumptions which have already proved that they are secure.

Authentication. Authentication is the process of verifying messages that are generated and transmitted from whether the claimed nodes and authorized nodes or not. It is important for ad-hoc networks to provide authentication because it is possible for malicious node or adversary to impersonate authorized nodes or legitimate nodes. In GKA protocol, we provide authentication through secure routing. AKE of MCP also guarantees authentication.

Integrity. When information is transmitted over insecure channels in ad-hoc networks, there is a risk that attackers see a message and change some important

data and resend it. The integrity is the ability of the secure system to guarantee that the received message is the real one that has not been altered. Also, some form of replay attacks might threaten the integrity attribute. We provide integrity in GKA protocol through secure routing. Especially, integrity is of great importance for majority of MCP. Therefore, we provide it through both secure routing and AKE.

Forward Secrecy.

1. Forward secrecy for GKA protocol: We assumed that cluster A (respectively, cluster B) agreed on the established group key K_A (respectively, K_B) before merging clusters. Therefore, a communication before merging clusters should be protected from a communication after merging clusters. We emphasize that our protocol use the established group keys (K_A or K_B) which were used in each cluster. However, that is not just using the established group key but using pseudo-random values ($\mathcal{F}(x)$ or $\mathcal{F}(y)$) of established group key. It is possible to protect the established group key before merging clusters from objects of the other cluster by PRF. Therefore, K_A or K_B cannot be recovered from $\mathcal{F}(x)$ or $\mathcal{F}(y)$.
2. Forward secrecy for MCP: Also, a new group key of both cluster A and cluster B is $K_{AB} = g^{\mathcal{F}(x)\mathcal{F}(y)}$. We use a Diffie-Hellman arithmetic since a pseudo-random function has large randomness. Thus, $\mathcal{F}(x)$ and $\mathcal{F}(y)$ cannot be recovered from a new group key K_{AB}. Therefore, a new group key can be protected from external objects.

4.2 Efficiency

There are two cases for getting efficiency of our protocol; Generating of a new group key under the Bresson's scheme [3] as GKA, and MCP which uses pseudo-random values of the established group keys. We compare the frequency of exponentiations, the frequency of flow transmissions according to two cases in Table 1. For this evaluation, we assume the followings. We assume that both generators and values p. We assume that the number of nodes in cluster A is n, the number of nodes in cluster B is k and the number of nodes in cluster A is more than the number of nodes in cluster B ($n > k$). Also, we use the Bresson's scheme [3] in the case of generating a new group key between cluster A and cluster B. the Bresson's scheme uses an encryption scheme and operators, but we consider values raising generators to random values power which are chosen by nodes. We assume that the order of nodes in Flow 1 is $A1 \rightarrow \cdots \rightarrow An \rightarrow Bk \rightarrow \cdots \rightarrow B1$. ($An$: cluster A-head, Bn: cluster B-head) Consequently, our protocol is more efficient than generating a new group key under the Bresson's scheme according to Table 1.

Table 1. Efficiency of our protocol

	Under the Bresson's scheme	Our protocol
exponentiation	$2(n+k)$times	$(n+k)$times
transmissions	$3n+3k-4$ times	$(n-1)+(k-1)$ times

5 Conclusion

In this paper, we have provided the group key agreement protocol for merging clusters in ad-hoc networks by not generating new group key but using the established key. Namely, we have provided the efficient group key agreement protocol in order to reduce overheads for generating new group key. In addition, we evaluated that our protocol is more efficient than generating a new group key. We have also demonstrated that our protocol provide security since we assume that secure routings and AKE.

In the future study, we will implement and simulate our protocol, and consider separating clusters. Moreover, we will implement secure routing suitable for applying our scheme to ad-hoc sensor networks.

References

1. M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated Key Exchange Secure Against Dictionary Attacks," In B. Preneel, editor, Proc. of Eurocrypt'00, LNCS 1807, pages 139-155, 2000.
2. E. Bresson, O. Chevassut, and D. Pointcheval, "The Group Diffie-Hellman Problems," In H. Heys and K. Nyberg, editors, Proc. of SAC'2002, LNCS, 2002.
3. E. Bresson, O. Chevassut, and D. Pointcheval, "Group Diffie-hellman key exchange secure against dictionary attacks," In Y. Zheng, editor, Proc. of Asiacrypt'2002, 2002. Full Version available at http://www.di.ens.fr/users/pointche.
4. M.S. Corson and J.P. Macker, "Mobile Ad-hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," IETF RFC 2501, 1999.
5. W. Diffie and M. E. Hellman, "New directions in cryptography," Transactions on Information Theory, IT-22(6):644-654, 1976.
6. Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," draft-ietf-manet-zone-zrp-04.txt, 2002.
7. Yih-Chen Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," In Proceedings of MOBICOM'02, 2002.
8. P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot and T. Clausen "Optimized Link State Routing Protocol," Internet Draft, RFC 3636.
9. M. Jiang, J. Li, and Y. Chiang Tay, "Cluster Based Routing Protocol (CBRP) Functional Specification," draft-ieft-manet-cbrp-spec-00.txt, 1998.
10. D. Johnson, D. Maltz, Y-C. Hu and J. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad-hoc Networks," Internet Draft, draft-ietf-manet-dsr-09.txt, 2003.
11. C. E. Perkins, P. Bhagwat, "Higly Dynamic Destination-Sequenced Distance Vector (DSDV) for Mobile Computers," Proc. of the SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications, pp 234-244, 1994.
12. C. Perkins, E.Royer and S. Das, "Ad-hoc On-demand Distance Vector (AODV) Routing," Internet Draft, draft-ietf-manet-aodv-11.txt, work in progress, 2002.
13. S. Yi, P. Naldurg, and R. Kravets, "A Security-Aware Routing Protocol for Wireless Ad Hoc Networks," The 6th World Multi-Conference on Systemics, Cybernetics and Informatics, 2002.