Key Agreement in Ad Hoc Networks

Ren-Junn Hwang, Rong-Chi Chang and Kai-Jun Lin
Department of Computer Science and Information Engineering
Tamkang University, Tamshi, Taipei Hsien 251, Taiwan
Email: {victor, roger}@mail.tku.edu.tw

Abstract

This paper proposes a new key agreement protocol based on a shared conference password. With this protocol, it provides an efficient algorithm and takes less computation cost to construct a secret communication channel. Besides, the honest participants can use password to authenticate themselves. The proposed scheme also provides an efficient protocol to reconstruct new session key when some members join or leave the conference.

Keywords: key agreement protocol, session key, Ad hoc networks, security

WORKSHOP: Workshop on Cryptology and information Security

Corresponding author:

Rong-Chi Chang

Post address: P.O. Box 1-130 Tamshui, Taipei Country, 251, Taiwan, R.O.C.

E-mail: roger@mail.tku.edu.tw

TEL: +886-2-26251652, +886-919-553561

FAX:+886-2-26251564

Key Agreement in Ad Hoc Networks

Ren-Junn Hwang, Rong-Chi Chang and Kai-Jun Lin
Department of Computer Science and Information Engineering
Tamkang University, Tamshi, Taipei Hsien 251, Taiwan
Email: {victor, roger}@mail.tku.edu.tw

Abstract

This paper proposes a new key agreement protocol based on a shared conference password. With this protocol, it provides an efficient algorithm and takes less computation cost to construct a secret communication channel. Besides, the honest participants can use password to authenticate themselves. The proposed scheme also provides a efficient protocol to reconstruct new session key when some members join or leave the conference.

Keywords: key agreement protocol, session key, Ad hoc networks, security

1. Introduction

With fast growth of the Internet and the shift of communication services to the network, group communication becomes increasingly important. Modern group-oriented applications include IP-telephony, video-conferencing and collaborative workspaces etc… Simultaneously, security and privacy become necessary. The security requirement of these applications can be addressed by building upon a secret key.

Group key agreement means that several parties want to create a common secret to be used in exchanging information covertly. For example, a group of people that is coming together in a closed meeting and wants to from a private wireless network with their laptop computers for duration of the ad hoc meeting. They want to share information security so that no one outside of the room can eavesdrop during their communication.

Ad hoc networks are dynamic, peer-to-peer network with little or no supporting infrastructure. The members of ad hoc networks may be PDA, mobile phone or notebook and so forth. These equipments are hardware-limited lack of storage devices and due to the security problems caused by ad hoc network, we consider a small group in a closed meeting. Members in this group know each other but can not digitally identifying and authenticating each another. Group members cannot provide or access third party key management service. They need a group shared key establishment protocol to construct a secure communication channel.

In general group key management protocols come in two different flavors: contributory key agreement protocols for small groups and centralized, server-based key distribution protocols for large groups. Becker and Wille [5] analyze the minimal communication complexity of group key distribution protocol and propose two protocols: *hypercube* and *octopus*. They proposed a method using Diffie-Hellman Key exchange protocol to construct a common group key. This protocol handles join and merge operations efficiently, but it is inefficient when the group member leave. Becker and Wille [5] proposed the *hypercube* protocol for the number of group member is just equal to the exponents of 2; otherwise, the efficiency to decrease. Steiner et al. [2] address dynamic membership issues of group key agreement based on the two-party Diffie-Hellman Key exchange [12]. The method named Group Diffie Hellman (GDH) protocols. GDH provides contributory authenticated key agreement and key independence. It requires one broadcast message at the end of each protocol run. The GDH protocol should be implemented on linear chain network topology where the last node has broadcast capabilities. The scheme uses a group controller and need n protocol rounds to establish a common key in a group of n members.

In this paper, we develop a key agreement protocol based on XOR operation [14]. The group members share a conference password. Each group member contributes its share to derive a common session key in a general ad hoc network environment without making additional assumptions about the availability of any support infrastructure. By the proposed method, the member generate group shared key more efficient then the previous methods.

The rest of this paper is organized as follows. In Section 2 introduces our key agreement protocols, along with novel security properties. Section 3 introduces membership events of group key agreement protocol. The protocol security discussion and complexity analysis are shown in Section 4. Finally, we make conclusions in Section 5.

2. Key agreement protocol

This section introduces our key agreement protocol. Subsection 2.1 describes a key tree structure that we construct based on the member numbers. The proposed protocol based this tree structure will be introduced in Subsection 2.2.

2.1. The key tree of the key agreement protocol

We assume that there are n members, M_1, M_2,…,M_n, want to hold a closed conference base on ad hoc network without network infrastructure. Each member of this group keeps a unique number over [1, n]. These members cooperate based on a *complete binary tree*. The complete binary tree is constructed based on each member unique number. Figure 1 shows an example of a key tree with 14 members. In the key tree, its root is located at level l=0 and its height is h, where h=4. Since the structure is a complete binary tree, every node is either a leaf or a parent of one or two children nodes. The nodes are denoted with each member's unique number. In this group, we assign the member M_n be "Checker". The checker is just a group member, but with an additional role to confirm the session key correctness. The member M_{n-1} is named "Candidate", who arranges replacement of member number after the member leave the conference meeting. For example, there are 14 members in the key tree shown Figure 1. Member number 1 is the root node. Member number 13 is the candidate and member number 14 is the checker. Besides, to simplify our subsequent protocol description, we introduce the term *key-path*, denoted as T_i, which is the tree path from root node to member M_i. In other words, every member M_i (except member M_1 and M_n) along his parent node to root node can build a key-path T_i. For example, the key-path T_5 of member M_5 in Figure 1 is the tree path $M_5 \rightarrow M_2 \rightarrow M_1$.

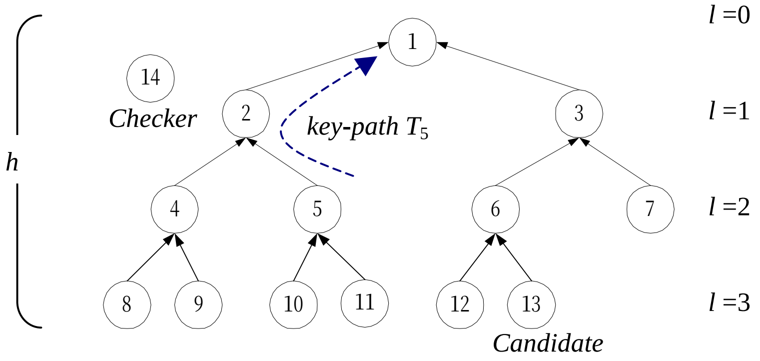

Figure 1: An example of the key tree structure

2.2. Two phases of the proposed protocol

This subsection introduces our key agreement protocol based on XOR operation. In our scenario, there are n members sharing a password P. Our goal is that at the end of the protocol all members who know P will get a shared session key $K = S_1 \oplus S_2 \oplus \ldots \oplus S_n$, where S_i is contributed by M_i. M_i selects S_i randomly. The protocol is divided into two phases. In the first phases, M_1, $M_2, \ldots, M_{n\text{-}1}$ cooperate to construct a subkey $\pi = S_1 \oplus S_2 \oplus \ldots \oplus S_{n-1}$ secretly. In the second phases, each M_i (i = 1, 2, …,n-1) engages in a separate exchange with M_n, all members have sufficient information to compute the session key K. He also verifies that the other members generated the same session key K. We introduce our method in detail as the following two phases:

Phase 1:

Each member M_i chooses a random quantity S_i, i is the node number that M_i located in the key tree. If the member M_i locates at leaf node (i.e. $2i > n$) of the key tree, he assigns his intermediate key K'_i as S_i. He sends intermediate key K'_i and verification message, F_i ($=f(P||K'_i)$, where $f(\bullet)$ is a public one-way hash function) to his parent node. The parent concatenates K'_i with P and generates a verification message F_i' by hash function $f(\bullet)$. If $F=F'$, the parent node authenticates the child note's identity and his S_i because they share the same P. The parent node records children's intermediate keys. If the member M_i locates at internal node (i.e. $2i \leq n$), he authenticates the children nodes' identities and their intermediate keys (e.g. K'_{2i} and K'_{2i+1}) by using verification messages F_{2i}($=f(P||K'_{2i})$ and F_{2i+1} ($=f(P||K'_{2i+1})$) separately. The M_i randomly selects a number S_i and generates intermediate key $K'_i = S_i \oplus K'_{2i} \oplus K'_{2i+1}$, where "$\oplus$" denotes the XOR operation. He also generate the verification message F_i ($=f(P||K'_i)$). Furthermore, he sends the intermediate key and verification message to his parent node. If the member is the root node (i.e. i = 1), who has to collect his children nodes' intermediate keys and use his random number S_1 to compute the subkey π ($=K'_1=S_1 \oplus K'_2 \oplus K'_3$). Note that the members perform the previous simultaneously when they locate on the same level of the key tree. The key agreement algorithm is presented below:

Algorithm 1: [phase 1 of key agreement protocol]

For each level *from* the level 0 to the last shallowest level of the key tree:

/* The members in the same level of the key tree perform the following steps simultaneously*/

Case 1 ($2i < n$-1): M_i verifies K'_{2i} and K'_{2i+1} by $F_{2i}=f(P||K'_{2i})$ and $F_{2i+1}=f(P||K'_{2i+1})$. He selects a random number S_i and computes $K'_i=K'_{2i} \oplus K'_{2i+1} \oplus S_i$. He also generates the verification message $F_i=f(P||K'_i)$ for K'_i. He sends F_i and K'_i to his parent node.

/* K'_{2i} and K'_{2i+1} are provided by M_{2i} and M_{2i+1} respectively. M_{2i} and M_{2i+1} are children of M_i */

Case 2 ($2i = n$-1): M_i verifies K'_{2i} by $F_{2i}=f(P||K'_{2i})$. He selects a random number S_i and computes $K'_i=K'_{2i} \oplus S_i$. He also generates the verification message $F_i=f(P||K'_i)$ for K'_i. He sends F_i and K'_i to his parent node.

/* K'_{2i} is provided by M_{2i}. M_{2i} is the child of M_i */

Case 3 ($2i > n$-1): M_i selects a random number S_i and assigns $K'_i=S_i$. He also generates the verification message $F_i=f(P||K'_i)$ for K'_i. He sends F_i and K'_i to his parent node.

/* M_i is a leaf node */

Case 4 (i=1): M_i verifies K'_2 and K'_3 by $F_2=f(P||K'_2)$ and $F_3=f(P||K'_3)$. He selects a random number S_1 and computes subkey $\pi=K'_1=K'_2\oplus K'_3\oplus S_1$.

/* K'_2 and K'_3 are provided by M_2 and M_3 respectively. M_2 and M_3 are children of M_1. M_1 is the root of key tree. */

Phase 2:

At the end of Phase 1, the member M_1 generates a subkey π (= $S_1\oplus S_2\oplus\ldots\oplus S_{n-1}$). In Step1 of this phase, the member M_1 broadcasts subkey π to each member, except the member M_n. In Step2, each member M_i (i = 1,2,…,n-1) removes its contribution from π and inserts a randomly chosen blinding factor S'_i. The resulting quantity, C_i, is equal to $\pi\oplus S_i\oplus S'_i$. Each member M_i (i = 1,2,…,n-1) sends C_i and the verification message $f(P||C_i)$ to member M_n. M_n verifies the message sent by each member. In Step3, M_n computes and sends $E_{P\oplus C_i}(C_i\oplus S_n)$ to each member M_i. He encrypted the message $C_i\oplus S_n$ by using the symmetric encryption function with key $P\oplus C_i$. The legal member decrypts the received messages to extract S_n. A this point, M_i (i = 1,2,…,n-1) unbinds the quantity received from M_n and constructs a session key $K_i=\pi\oplus S_n$. In Step4, each member M_i (for i=1, 2, …, n-1) sends the key confirmation message of K_i as $E_{P\oplus S_n}(K_i)$ to member M_n, where $E_{P\oplus S_n}(K_i)$ denotes encrypting K_i with a symmetric encryption function and key $P\oplus S_n$. In Step5, the member M_n verifies that each member generated the same session key K ($=K_1=K_2=\ldots=K_{n-1}$). M_n notifies all members the conference that the session key is established successfully. The algorithm key agreement protocol is shown as following:

Algorithm 2: [phase 2 of key agreement protocol]

Step1: $M_1\rightarrow M_i$: π, $f(P\|\pi)$; for i=2,3,…,n-1 and $\pi = S_1\oplus S_2\oplus\ldots\oplus S_{n-1}$

Step2: $M_i\rightarrow M_n$: C_i, $f(P\|C_i)$; for i=1,2,…,n-1, and $C_i=\pi\oplus S_i\oplus S'_i$, S'_i is a blinding factor that is randomly chosen by M_i

Step3: $M_n\rightarrow M_i$: $E_{P\oplus C_i}(C_i\oplus S_n)$; for i =1,2,…,n-1

Step4: $M_i\rightarrow M_n$: M_i, $E_{P\oplus S_n}(K_i)$; for i =1,2,…,n-1 and $K_i=(\pi\oplus S_n)$

Step5: M_n check session key K

3. Membership events

In our scenario, the conference members are not always fixed. Some times there are new members joint the conference, after the session key is generated. This new member does not authorize to know the messages of this conference before he joins this conference. The conference should change their session key and the shared password. Some times there are some members leave. They do not authorize to get the messages after they leave. This conference should change the session key and the shared password, too. This section introduces two protocols to generate the new session key when the group member is adapting.

3.1. Joining protocol

We assume that the group has n members: M_1, M_2,…,M_n. The new member M_{n+1} wants to join the conference. The new member M_{n+1} initiate the protocol by sending a joining request message that contains his member number. The new member will be allowed to join the conference when the members in the conference receive this message and permit it. Furthermore, the members of the conference have to change the password from P to P' and reconstruct the session key. We describe the reconstructing protocol in the following paragraph.

Firstly, M_{n+1} sends random quantity S_{n+1} encrypted with new password P' to old members M_1, M_2,…,M_n. In this conference, the old members receive and decrypt the message to extract S_{n+1}. Each members M_i (i =1,2,…,n) computes a session key $\hat{K}_i$ $=K\oplus P\oplus S_{n+1}$. Secondly, the M_n sends a quantity $K\oplus P$ encrypted with new password P' (i.e. $E_{P'}(K\oplus P)$) to member M_{n+1}. Note that M_{n+1} computes the session key $\hat{K}_{n+1}$ by $\hat{K}_{n+1}=S_{n+1}\oplus(K\oplus P)$. Thirdly, M_{n+1} encrypts the session key $\hat{K}_{n+1}$ with key $P'\oplus S_{n+1}$ and sends it to M_1. M_1 verifies the session key $\hat{K}$ that member M_{n+1} generated. Figure 2 shows the joining protocol in detail:

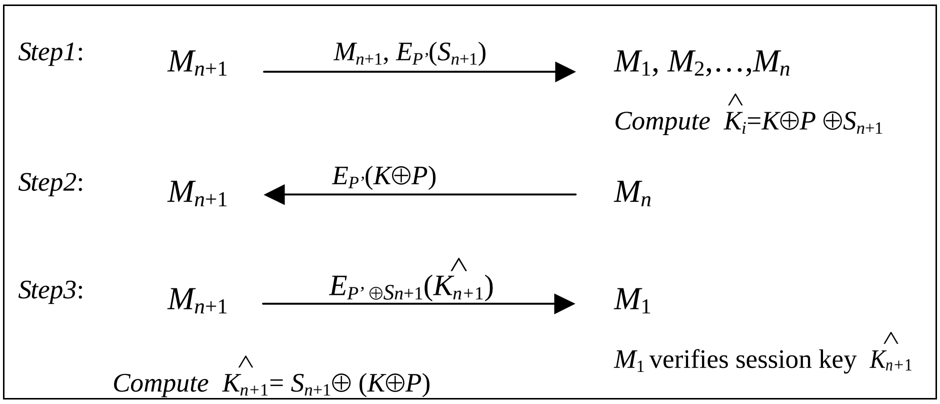

Figure 2: key agreement protocol of a new member joining the conference

3.2. Leaving protocol

Assume that there are n members in the conference room and the member M_d wants to leave the conference. When a member leaved the conference, the session key should be changed and the key tree structure is altered. In order to maintain the security of conference meeting, the shared password should be changed to *P'*, too. The leaving member initiates the leave protocol by sending a leave message to all conference members. The candidate member changes his member number as the member number of leaving member. The candidate picks a new random number and sends it to his new parent node.

The reconstructing protocols are different based on the M_d's location in the key tree. Generally, we divide two cases to introduce the leaving protocol.

Case 1: M_d is a leaf node

Figure 3 shows an example of this case clearly. If M_{11} leaves the conference, the candidate, M_{13}, (the right-most leaf node) alters his member number to 11. Moreover, the checker M_{14} changes its member number to 13. M_{12} is the new candidate of this conference.

The key tree is reorganized. Figure 3 (b) is the new key tree structure. Firstly, each of the members, M_i, except the root M_1 on the key-path, $M_{11} \rightarrow M_5 \rightarrow M_2 \rightarrow M_1$, should select a new random number S''_i. The first node of this key-path, M_{11}, sends his new random number to his parent as $E_{p'}(S''_{11})$. Secondly, the node, M_i, except the leaf node compute the new intermediate key $K''_i = K''_{2i} \oplus K''_{2i+1} \oplus S''_i$, where K''_{2i} and K''_{2i+1} are intermediate keys transformed by his children. If M_i's child M_{2i} does not in the key-path, then $K''_{2i} = K'_{2i}$. M_i sends intermediate key K''_i to his parent. The key-path that each member in it should joint previous process is from the leaving member's location to the root. Finally, M_1 performs the algorithm 2 in Section 2 to reconstruct and verify a new session key.

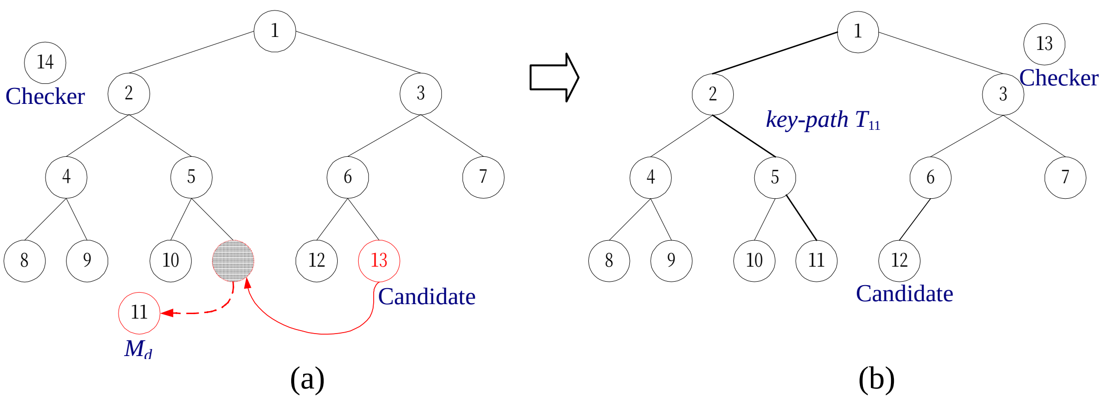

Figure 3: Tree updating in leave operation- M_d is leaf node

Case 2: M_d is an internal node

Figure 4 shows an example of this case. The leaving member, M_5, is an internal node of this key tree. The candidate member, M_{13}, change his number to 5. Moreover, the checker M_{14} changes its member number to 13, and the new candidate is M_{12}.

The key tree is altered. Figure 4 (b) is the new key tree structure. The conference has to change its session key by security considerations. Firstly, each member, M_i, except the root member in the key-path, $M_5 \rightarrow M_2 \rightarrow M_1$, and the two children of M_5 select a new random number S''_i. Secondly, each of the two children, M_i, of the first node of the key-path, sends the new random number to his parent as $E_{p'}(S''_i)$. Thirdly, each member M_j of the key-path compute a new intermediate key as $K''_j = K''_{2j} \oplus K''_{2j+1} \oplus S''_j$, where K''_{2j} is a intermediate key sent by the child node M_{2j}. If the child M_{2j} does not in the key-path then $K''_{2j} = K'_{2j}$. The special key-path is from the altered node to the root. Finally, M_1 perform the algorithm 2 of Section 2 to reconstruct a new session key.

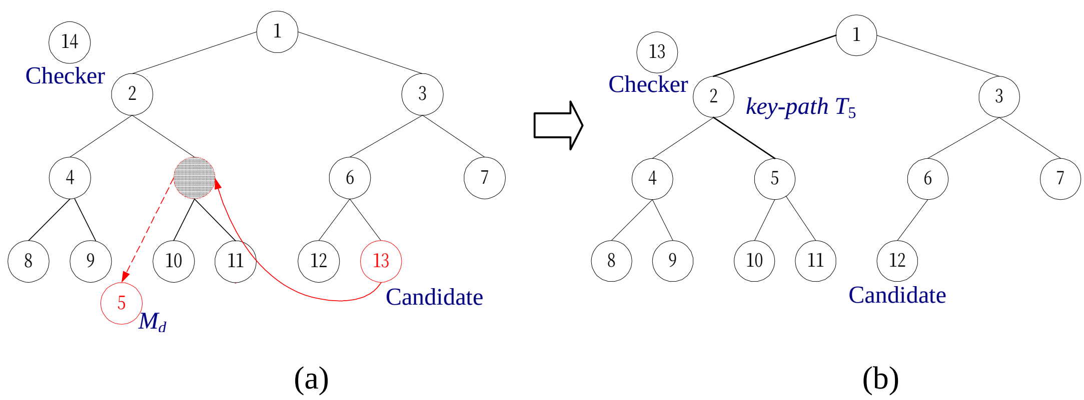

Figure 4: Tree updating in leave operation- M_d is internal node

4. Discussion

We discuss the security analysis and efficiency of the proposed protocols in this section. The subsection 4.1 discusses the forward and backward secrecy. We make the comparisons among the GDH.2 [2], hypercube [5], octopus protocols [5] and our method in Subsection 4.2.

4.1. Security analysis

The new member of the conference cannot derive the session key before he join this conference is named backward secrecy. It is important, because that new member is legal after he joins. The session key can be derived by knowing the random number.

In the member joining protocol, the joining member M_{n+1} picks a random quantity S_{n+1} and encrypted it by password P' and broadcasts it with its join request. The conference password P is changed to P'. M_n sends a quantity, $K \oplus P$, to M_{n+1} which is encrypted with P'. At this point, the M_{n+1} can compute the new session key $\hat{K}$, but he cannot derive old session key K, because the M_{n+1} does not know the old password P. The proposed protocol provides the backward security.

A member M_d leaving the conference cannot get the new session key after his leaving is named forward secrecy. In the leaving protocol, the candidate changes its share and member number to replace the leaved member location when M_d leaves the conference. The members in the key-path which construct from M_d to root change their shares. The leaving protocol reconstructs the session key based on these shares. The leaving member, M_d, cannot generate the new session key. The new session key is regenerated by new shared random number that M_d does not know. The proposed protocol provides forward secrecy.

4.2. Efficiency discussion

This subsection analyzes the communication and computation costs of the key agreement protocols. Table 1 shows the comparisons among GDH.2 [2], hypercube [5], octopus protocols [5] and our protocols. The *n* denotes the number of member in this conference. The second row of the Table 1 shows the numbers of *DH-Key exchanges* that are sent by two members. In the third row, the *simple round* means that every member can send or receive at most one message per round. [5] In the last row, the *broadcast* means one member sends a message to each member simultaneously.

By the Table 1, it is clearly that GDH.2 and our protocol need fewer numbers of communication messages. The Octopus protocol need fewer number of 2-party DH exchanges than ours, but in our protocol all member use the XOR operation to compute the session key. The XOR operation takes less computation cost than DH exchange operation. As Table 1 illustrates, our method need $\log_2 n+1$ times of the simple rounds. The number of simple rounds of our scheme is fewer than GDH.2 and

Octopus protocols'. In generally, Table 1 shows that our protocol is more efficient than the others.

Table 1: Protocols comparison

Methods / Items	GDH.2	Hypercube	Octopus	Our method
The number of messages send via the communication	n	$n\log_2 n$	$3n$-4	n
DH-Key Exchanges	n	$\frac{n\log_2 n}{2}$	$2n$-4	0
Simple Rounds	n	$\log_2 n$	$2\left\lceil \frac{n-4}{4} \right\rceil + 2$	$\log_2 n+1$
Broadcast	Yes	No	No	Yes

5. Conclusions

In this paper, we propose new protocols for password-based key agreement in ad hoc networks. The environment does not provide additional infrastructure and physically secures communication channels. In our protocol, the legal conference member use password to authenticate participants and lower computing operations for the session key generation. In addition, this protocol supports dynamic conference member events. The proposed protocol is more efficient than the others.

6. References

[1] N. Asokan and Philip Ginzboorg. “Key-agreement in ad hoc networks”, *Computer Communications*, Vol. 23, No. 17, Nov. 2000, pp. 1627-1637.

[2] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. “Authenticated group key agreement and friends”, In *Proc. 5th ACM Conference on Computer and Communications Security*, Nov. 1998, pp. 17-26.

[3] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. “New multiparty authentication services and key agreement protocols”, *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 4, Apr. 2001, pp. 628-640.

[4] Giuseppe Ateniese and Gene Tsudik. “Some open issues and new directions in group signatures”, In *Proc. 3rd International Conference on Financial Cryptography (FC’99)*, Vol. 1648 of *LNCS*, Feb. 1999, pp. 196-211.

[5] Klaus Becker and Uta Wille. “Communication complexity of group key distribution”, In Proc. *5th ACM Conference on Computer and Communications Security*, Nov. 1998, pp. 1-6.

[6] M. Bellare, D. Pointcheval and P. Rogaway, “Authenticated key exchange secure against dictionary attack”, *Proceeding of Advances in Cryptology – Eurocypt 2000*, pp. 139-155.

[7] S. Bellovin and M. Merritt, “Encrypted key exchange: password-based protocols secure against dictionary attack”, *IEEE Symposium on Research in Security and Privacy*, 1992, pp. 72-84.

[8] S. Bellovin and M. Merritt, “Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password-life compromise”, *ACM Conference on Computer and Communications Security*, 1993, pp. 244-250.

[9] M. Boyarsky, “Public-key cryptography and password protocols: the multi-user case”, *ACM Conference on Computer and Communications Security*, Sep.1999, pp. 63-72.

[10] V. Boyko, P. Mackenzie and S. Patel, “Provably secure password authenticated key exchange using Diffie-Hellman”, *Proceedings of Advances in Cryptology-Eurocrypt 2000*, 1998, pp. 156-171.

[11] Mike Burmester and Yvo Desmedt. “A secure and efficient conference key

distribution system", In *Advances in Cryptology - EUROCRYPT '94*, Vol. 950 of *LNCS*, May 1994, pp. 275-286.

[12] W. Diffie, and M.E. Hellman, "New directions in cryptography", *IEEE Trans. On Information Theory*, Vol. IT-22, No.6, 1976, pp. 644-654.

[13] D. Jablon, "Extended password key exchange protocols", *WETICE Workshop on Enterprise Security,* 1997.

[14] Sahar M. Ghanem, Hussein Abdel-Wahab, "A simple XOR-based technique for distributing group key in secure multicasting", In Proc. *The Fifth IEEE Symposium on Computers and Communications*, pp. 166-171, 2000.

[15] Ingemar Ingemarsson, Donald T. Tang, and C. K. Wong. "A conference key distribution system", *IEEE Transactions on Information Theory*, Vol. IT-28, No. 5, Sep. 1982, pp. 714-720.

[16] Y. Kim, A. Perrig, and G. Tsudik. "Simple and fault-tolerant key agreement for dynamic collaborative group", In *7th ACM conference on Computer and communications,* Nov. 2000, pp. 235-244.

[17] Silja Mäki, Maarit Hietalahti, and Tuomas Aura. "A survey of ad-hoc network security", Interim report of project 007- security of mobile agents and ad-hoc societies, *Helsinki University of Technology, Laboratory for Theoretical Computer Science,* Sep. 2000.

[18] Michael Steiner, Gene Tsudik, and Michael Waidner. "Diffie-Hellman key distribution extended to group communication", In *3rd ACM Conference on Computer and Communications Security*, Mar. 1996, pp. 31-37.

[19] Michael Steiner, Gene Tsudik, and Michael Waidner. "CLIQUES: a new approach to group key agreement", In *Proc. 18th International Conference on Distributed Computing Systems (ICDCS'98),* May 1998, pp. 380-387.

[20] Michael Steiner, Gene Tsudik, and Michael Waidner. "Key agreement in dynamic peer groups", *IEEE Transactions on Parallel and Distributed Systems,* Vol. 11, No. 8, Aug. 2000, pp. 769-780.