Asian Journal of Information Technology 6 (11): 1181-1186, 2007
ISSN: 1682-3915
© Medwell Journals, 2007

A Password-Based Group Key Agreement Protocol for Wireless Ad-Hoc Networks

[1]Rony Hasinur Rahman and [2]M. Lutfar Rahman
[1]Department of Computer Science and Engineering, Eastern University,
House 15/2, Road 3, Dhanmondi R/A, Dhaka-1205, Bangladesh
[2]Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh

Abstract: Ad-hoc networks offer communication over a shared wireless channel without any pre-existing infrastructure. Forming security association among a group of nodes in ad-hoc networks is more challenging than in conventional networks due to the lack of central authority. With that view in mind, group key management plays an important building block of any secure group communication. The main contribution of this study, is a low complexity key agreement scheme that is suitable for fully self-organized ad-hoc networks. The proposed protocol is based on the multi-party version of the famous Diffie-Hellman key exchange protocol. The protocol is also password authenticated, making it resilient against active attacks. Unlike other existing key agreement protocols, ours make no assumption about the structure of the underlying wireless network, making it suitable for truly ad-hoc networks. Finally, we will also show that our protocol minimizes the computation and communication burden on individual computers (nodes) for key establishment by comparing its complexity with some popular and well-known key agreement protocols.

Key words: Ad-hoc networks, group key management, key distribution protocols, password authentication

INTRODUCTION

Let us assume that a small group of people at a conference has come together in a room for an ad hoc meeting. They would like to set up a wireless network session with their laptop computers for the duration of the meeting. They want to share information securely so that no one outside the room can eavesdrop and learn about the contents of the meeting. The people physically present in the room know and trust one another. However, they do not have any a priori means of digitally identifying and authenticating each other, such as shared secrets or public key certificate authority or access to trusted third party key distribution centers. An attacker can monitor and modify all traffic on the wireless communication channel and may also attempt to impersonate as a valid member of the group. There is no secure communication channel to connect the computers. The problem is: how can the group set up a secure session among their computers under these circumstances? The network in the scenario described above is an example of an Ad-hoc network in which entities construct a communication network with little or no infrastructural support. In recent years, mobile ad-hoc networks have received a great deal of attention in both academia and industry because they provide anytime-anywhere networking services. Ad-hoc networks have overwhelming influence on military warfare where troops can be deployed anywhere in the world and in any hostile environment. Moreover, they need to establish a secure communication channel among themselves quickly and also they have to maintain the security of that channel in case of group detachment and re-attachment. As wireless networks are being rapidly deployed, secure wireless environment will be mandatory.

In this study, we are going to propose an efficient group key agreement protocol which is based on multi-party Diffie-Hellman group key exchange and which is also password-authenticated.

Key distribution in ad hoc networks is divided into three main classes.

Centralized group key management protocols: A single entity called the Key Distribution Center (KDC) is employed for controlling the whole group.

Decentralized architectures: The management of a large group is divided among subgroup managers, trying to minimize the problem of concentrating the work in a single place.

Distributed key management protocols: There is no explicit KDC and all members participate in the generation of the group key and each member contributes to a portion of the key.

Corresponding Author: Rony Hasinur Rahman, Department of Computer Science and Engineering, Eastern University, House 15/2, Road 3, Dhanmondi R/A, Dhaka-1205, Bangladesh

Centralized group key management protocols: With only one managing entity, the central server is a single point of failure. The group privacy is dependent on the successful functioning of the single group controller; when the controller is not working, the group becomes vulnerable because the keys, which are the base for the group privacy, are not being generated/regenerated and distributed. Furthermore, the group may become too large to be managed by a single party, thus raising the issue of scalability. The group key management protocol used in a centralized system seeks to minimize the requirements of both group members and KDC in order to augment the scalability of the group management. The efficiency of the protocol can be measured by: Storage requirements, size of messages, backwards and forward secrecy and collusion. Some popular centralized protocols are: Group Key Management Protocol (GKMP) (Harney and Muckenhirn, 1997a), Logical Key Hierarchy (LKH) (Wallner *et al*., 1999), One-way Function Tree (OFT) (McGrew and Sherman, 1998), Efficient Large-group Key (ELK) Protocol (Perrig *et al*., 2001).

Decentralized architectures: In the decentralized subgroup approach, the large group is split into small subgroups. Different controllers are used to manage each subgroup, minimizing the problem of concentrating the work on a single place. In this approach, more entities are allowed to fail before the whole group is affected. We use the following attributes to evaluate the efficiency of decentralized frameworks: Key independence, Decentralized controller, Local rekey, Keys vs. data and Rekey per membership. Scalable Multicast Key Distribution (Ballardie, 1996), Iolus (Mittra, 1997), Dual-Encryption Protocol (DEP) (Dondeti *et al*., 2000), Kronos (Setia *et al*., 2000), Intra-Domain Group Key Management (IGKMP) (Decleene *et al*., 2001), Hydra (Rafaeli and Hutchison, 2002) are some of the popular protocols that follow the decentralized architecture.

Distributed key management protocols: The distributed key management approach is characterized by having no group controller. The group key can be either generated in a contributory fashion, where all members contribute their own share to computation of the group key, or generated by one member. In the latter case, although it is fault-tolerant, it may not be safe to leave any member to generate new keys since key generation requires secure mechanisms, such as random number generators, that may not be available to all members. Moreover, in most contributory protocols (apart from tree-based approaches), processing time and communication requirements increase linearly in term of the number of members. Additionally, contributory protocols require each user to be aware of the group membership list to make sure that the protocols are robust. Our proposed protocol falls in this category. We use the following attributes to evaluate the efficiency of distributed key management protocols: Number of rounds, Number of messages, Processing during setup, DH key and Number of exponentiations. Some popular protocols in this category are Burmester and Desmedt (BD) Protocol (Burmester and Desmedt, 1994), Group Diffie-Hellman Key Exchange (G-DH) (Steiner *et al*., 1996), Octopus Protocol (Becker and Wille, 1998), Diffie-Hellman Logical Key Hierarchy (DH-LKH) (Kim *et al*., 2000), Password Authenticated Multi-Party Diffie-Hellman Key Exchange (PAMPDHKE) Protocol (Asokan and Ginzboorg, 2000).

THE PROPOSED PROTOCOL

The basic idea of the protocol is to securely construct and distribute a secret session key, K, among a group of nodes/users who want to communicate among themselves in a secure manner. The group is formed in an ad hoc fashion. Our proposed protocol is based on the Password Authenticated Multi-Party Diffie-Hellman Key Exchange (PAMPDHKE) Protocol described in (Asokan and Ginzboorg, 2000). The protocol described in (Asokan and Ginzboorg, 2000) does not give us any idea about the structure of the ad hoc network and is described in a very vague way without mentioning the details of every action. Without detailed description, some actions beg the question of validity in ad hoc scenario. The proposed protocol starts by constructing a spanning tree on-the-fly involving all the valid nodes in the scenario. It is assumed, like all other protocols, that each node is uniquely addressed and knows all its neighbors (i.e., the protocol runs on top of the network layer and it assumes that a valid route among the nodes have already been constructed by some underlying routing protocol). It is also assumed that each valid member of the scenario shares a password (also called a weak secret) P. After that the tree is traversed from bottom-to-top where each node i, sends to its parent, its Diffie-Hellman contribution α^{g_i}, where α is a generator of the multiplicative group Z_p^* (i.e., the set $\{1,2,\dots p-1\}$) and g_i is node i's secret. In this way every contribution ultimately reaches the root of the tree. Then the root creates separate messages for each of its children where each message contains sufficient information so that the child can compute the secret session key K. This process continues in a top-to-bottom fashion from every internal node to all its children. In the end, all the valid nodes in the tree contain sufficient information to construct the session key K. The messages passed from one node to another may be encrypted by the shared weak secret P according to necessity. When all the valid nodes in the group have K, they can communicate with one another in a secure manner by encrypting every message with K.

Construction of the spanning tree: Our key agreement protocol functions in an arbitrary rooted tree structure. For this purpose, a spanning tree over the graph has to be constructed first. This can be done in several ways. Below we will describe one possible protocol for constructing a spanning tree where the node initiating the protocol becomes the root. The tree is indexed using universal addresses. In the initial state it is assumed that the nodes know their neighbors. The initiator sends a message to each of its neighbors. It thereby becomes the root of the tree and the neighbors become its children. After receiving a message, a node acknowledges it and sends a similar message to all its neighbors, except to the parent. The nodes that acknowledge a message from a node become its children in the tree. If a node gets more than one of these messages, it acknowledges and processes only the message that it receives first. Consequent messages are ignored. This continues until every node has received this kind of a message. A leaf is a node that does not receive acknowledgements from any of its neighbors. It should be noted that the structure of the final spanning tree might be different based upon the order in which messages are received by each individual node.

Phase I of the protocol: Let Z_p^* (i.e. the set {1,2,...p-1}) be a finite multiplicative group where p is a prime and let α be the generator of the group. A participant/node, i, is assumed to pick his/her secret exponent g_i randomly where $1 \leq g_i \leq p-1$. The steps of phase I are described below:

1. Every internal node gets the contributions from all its children.
2. Each node generates its own contribution and multiplies all its descendants' contributions with its own.
3. If node i is not the root, then it executes this step. Node i sends the product obtained from step (2) to its parent along with its own contribution and all the other contributions (of i's descendant nodes) that was forwarded to *i* from its immediate children.
4. If node i is the root, then it executes this step. The product obtained from step (2) represents the final group session key K. Phase I stops here.

Formally,

- In phase I, each node×sends a message $M = \{M_1, M_2, M_3\}$ (having 3 parts) to its parent y, where M_1 = the product of x's contribution and all contributions from the descendants of x, M_2 = x's own contribution and M_3 = all contributions from the descendants of x.
- If x is a leaf node with contribution α^{gx} and y is its parent, then, x sends to y a message containing the quantity α^{gx}, i.e.,

$$x \rightarrow y : \{\alpha^{g_x}, \alpha^{g_x}, -\}$$

- If x is an internal non-root node with contribution α^{gx}, y is its parent and a, b, c etc. are its children (with contributions α^{ga}, α^{gb}, α^{gc} etc. respectively) and if x receives messages M_a, M_b, M_c etc. from a, b, c etc., respectively, then, x sends to y a message containing the quantity [α^{gb}, A. B. C. D, α^{gx}, M', i.e.,

$$\begin{aligned}
&x \rightarrow y : \{\alpha^{g_x}.A.B.C.D, \alpha^{g_x}, M'\}, \text{ where,}\\
&M' = \{\{M_2 \text{ part of } M_a\} \cup \{M_3 \text{ part of } M_a\} \cup\\
&\quad \{M_2 \text{ part of } M_b\} \cup \{M_3 \text{ part of } M_b\} \cup\\
&\quad \{M_2 \text{ part of } M_c\} \cup \{M_3 \text{ part of } M_c\} \cup\\
&\quad \{M_2 \text{ part of etc.}\} \cup \{M_3 \text{ part of etc.}\}\}\\
&A = M_1 \text{ part of } M_a \text{ ; } B = M_1 \text{ part of } M_b \text{ ;}\\
&C = M_1 \text{ part of } M_c \text{ ; } D = M_1 \text{ part of etc.}
\end{aligned}$$

- If x is the root node with contribution α^{gx} and a, b, c etc. are its children (with contributions α^{ga}, α^{gb}, α^{gc} etc., respectively) and if *x* receives messages M_a, M_b, M_c etc. from a, b, c etc., respectively, then, x computes the final session key K,

$$\begin{aligned}
&K = \alpha^{g_x}.A.B.C.D, \text{where,}\\
&\quad A = M_1 \text{ part of } M_a; \ B = M_1 \text{ part of } M_b;\\
&\quad C = M_1 \text{ part of } M_c; \ D = M_1 \text{ part of etc.}
\end{aligned}$$

- The quantity α^{gx} A. B. C. D indicates the product of α^{gx}, A, B, C and D.

Phase II of the protocol: Before the beginning of phase II, first, the root takes the union of all the M_2 parts and all the M_3 parts of all the messages it has received from its immediate children. Then it raises each quantity of this newly formed set by its own secret exponent. The steps of phase II are described below:

- Every internal node x sends to its child i sufficient information needed by i to construct the session key K. The node x also sends to i a quantity encrypted by K for authentication purpose and forwards sufficient information so that descendants of i may successfully construct the session key K.
- When every leaf node gets messages from its parent, phase II stops. Every valid node now has the session key K and has been authenticated.

Formally,

- In phase II, each internal node *x* sends a message $M_i^* = \{P(\overline{M_1}), \overline{M_2}, \overline{M_3}, \overline{M_4}\}$ (having 4 parts) to each of its child i, where $\overline{M_1}$ = i's contribution raised to the power of root's secret exponent, $\overline{M_2}$ = all contributions from all other nodes except the root and the descendants of i, $\overline{M_3}$ = all contributions of the descendants of I raised to power of the root's secret exponents and $\overline{M_4}$ = K(n) = a quantity encrypted with the session key K needed for authentication.
- If x is an internal node with contribution α^{g_x}, y is its parent (may be null if x is the root) and a, b, c etc. are its children (with contributions α^{g_a}, α^{g_b}, α^{g_c} etc. respectively) and if *x* receives messages M_a, M_b, M_c etc. from a, b, c etc., respectively and creates the message M in phase I and receives M_x^* from y, then, x sends to its child i ϵ {a, b, c, etc.,} a message M_i^* containing the quantity $\{P(\alpha^{g_i g_{root}}), G_i, H_i, K(n_x)\}$, i.e.,

$$x \rightarrow i : \{P(\alpha^{g_i g_{root}}), G_i, H_i, K(n_x)\}, \text{for } i \in \{a, b, c, \text{etc.}\} \text{where,}$$

$\alpha^{g_i g_{root}}$ is obtained from $\overline{M_3}$ part of M_x^* or is already present in x (if x = root),

$G_i = \{\{\overline{M_2} \text{ part of } M_x^*\} \cup \{\alpha^{g_x}\} \cup \{M_1 \text{ part of } M_j\}\}$

where $j \in \{a, b, c, \text{etc.}\}$ and $j \neq i$ and also

$[\ \{\alpha^{g_x}\} = \phi \text{ if } x = \text{root}\ ]$

$H_i = \{k^{g_{root}}\}$ where $k \in \{M_3 \text{ part of } M_i\}$

$K(n_x)$ = ID of x encrypted by session key K

- If a non-root node x receives the message M_x^* and also creates the message M in phase I, then, it calculates the key K as follows,
    - It first decrypts $\overline{M_1}$ with the weak password P to retrieve $L = \alpha^{g_x g_{root}}$.
    - Then it retrieves $\alpha^{g_{root}}$ by performing $\frac{1}{L^{g_x}}$.
    - $K = \alpha^{g_{root}}.s. < M_1 \text{ part of } M >,$
      $\forall s \in \{\overline{M_2} \text{ part of } M_x^*\}$
- So now all the nodes have the key $K = \alpha^{\Sigma g_i}$, where i is a node of the network and g_i is its secret exponent.
- After that, each node decrypts $\overline{M_4}$ with K and verifies whether the quantity is the identity of its parent. This step authenticates the parent to all of its children.

PERFORMANCE EVALUATION

In this study, the proposed is evaluated in light of security and efficiency (more precisely, communication complexity).

Security: In our protocol, it is assumed that a weak secret/password P is shared among the valid users/nodes. This P helps in the authentication process and prevents man-in-the-middle attack. This assumption is not at all inappropriate. The ad-hoc scenarios that were mentioned in the beginning of this study indicate that the people involved in those scenarios trust each other. So it is possible for them to decide on a simple password because they will definitely come in contact with each other before forming the actual ad-hoc network. That password may be written down on piece of paper and circulated to all the trusted parties and the recipients can use it as the weak shared secret for our protocol. It may be noted that this password will never be used to encrypt data traffic.

It is very obvious from the example given in the previous section, that, every valid node has necessary and sufficient information to construct the session secret key K, which will be the group key for that session. Now it remains to show that a passive adversary as well as an active adversary will never be able to construct K from the messages that travel through the wireless network. First of all, it is very clear that the secret exponent g_x for some node x, is never exposed to the network. To construct K, an adversary needs the contribution α^{gl} of each valid node i. But one can see, very obviously, from phase II of the protocol that the contribution of the root, $\alpha^{g_{root}}$, is never sent into the network by itself, i.e. it is always sent in the form $\alpha^{g_{root} g_x}$, where g_x is the secret exponent of a valid non-root node x. Without knowing g_x, no one (not even another valid node y) can obtain $\alpha^{g_{root}}$ from $\alpha^{g_{root} g_x}$. The only way that an adversary (active or passive) can get a hold of g_x from x, is to hack into node x and compromise it. Moreover, the problem of calculating $\alpha^{g_{root}}$ by using $\alpha^{g_{root} g_x}$ and $\alpha^{g_{root}}$ is a Decision Diffie-Hellman Problem (DDHP) (Boneh, 1998) which is intractable. In a nutshell, unless a valid node is compromised, an adversary (passive or active) will never be able to construct any session key K by observing/obtaining the messages of phase I and II of the protocol.

Table 1: Comparison of distributed key management protocols

		No. of messages			Setup	
Scheme/feature	No. of rounds	Multicast	Unicast	DH key	Explicit leader ?	Average exponentiation(s) per node
BD	3	2n	0	N	N	n + 1
G-DH	N	N	n-1	Y	N	i + 1
Octopus	2(n-4)/4+2	0	3n-4	Y	Y	4
DH-LKH	$\log_2 n$	$\log_2 n$	0	Y	N	$\log_2 n + 1$
PAMPDHKE	n + 2	2	3n-3	Y	N	3
Our Protocol	$2\log_k n$	0	2n-1	Y	N	1

The second line of defense is the weak shared secret P. P is only used to encrypt the first part of a message in phase II. This is used to prevent active adversaries from carrying out man-in-the-middle attack. The adversary does not know P. So if it tries to mislead a valid node x by sending dummy or meaningless messages or by impersonating as a valid node, it will always fail because it does not have the capability to encrypt the first part of a message in phase II.

Efficiency: As the topology of the spanning tree is arbitrary, the number of children is not limited and exact figures are not always possible. We estimate the figures by assuming that the tree is a balanced perfect k-ary tree. In phase I, every node except the root sends a message. This makes the number of messages in phase I n-1. In phase II, every node except the root receives a message. So the total number of messages in the protocol is 2n-1. Broadcast/multicast is a serious bottleneck for wireless networks. Unlike many other protocols, ours does not need broadcast/multicast capability. Our protocol uses Diffie-Hellman key shares and hence it is contributory. The protocol needs no explicit leader election technique. Anyone wishing to form a secure group can start constructing the spanning tree and thereby can become the root of tree. And the root implicitly becomes the leader of the group. In both phases of the protocol, the number of rounds/iterations needed is $O(\log_k n)$. In phase I, each member generates its own contribution and so the total number of exponentiations in this phase is n. At the start of phase II, the root raises each member's contribution to the power of its own exponent, i.e., performs n exponentiations. In the remaining rounds of phase II, each non-root node performs one exponentiation to retrieve the root's contribution. So the total number of exponentiations in the protocol is 3n-1 = O(n). By amortized analysis, the number of exponentiations performed (on an average) by a single node is O(n)/n = O(1).

Unfortunately none of the other existing protocols take the issue of setting up the initial infrastructure under consideration. More or less all of the existing protocols pre-assume some sort of infrastructure among the nodes. This assumption is made based on the working principle of the respective protocol. Since one can not predetermine the structure of an ad-hoc network, the suitability as well as applicability of those existing protocols, to a certain extent, depend on the structure of the wireless network. So they are not suited for truly ad-hoc (fully infrastructure less) environment. Since our protocol begins by constructing a logical ad-hoc structure (spanning tree) on the fly, it is much more superior to existing protocols simply because of its capability to work under and adapt to any truly ad-hoc environment. Table 1 compares the properties of existing distributed key management protocols with our proposal.

CONCLUSION

In this study, we have proposed a new group key agreement protocol suitable for wireless ad-hoc networks of arbitrary topology. It is a 2-phase protocol based on Diffie-Hellman contributions. We have demonstrated the protocol with an extensive example and mentioned the contents of all the messages at various rounds of each phase. We have also shown that the protocol is secure against active and passive adversaries and its overall communication complexity, based on Table 1, is very low compared to existing protocols in the same category.

Several lines of future research are possible. Formal security analysis is a missing step. Moreover, changes in the physical topology of the group during and after the execution of the protocol have to be studied thoroughly, more precisely, joining-the-group (join protocol) and leaving-the-group (leave protocol) operations. The research on join/leave protocol is already underway. Finally, we conclude that group key distribution/management is still an open research area. Much work is still needed to secure group communication in an ad-hoc network with perfection and efficiency.

REFERENCES

Asokan, N. and P. Ginzboorg, 2000. Key Agreement in Ad hoc networks. In Elsevier Journal of Computer Communications. Comput. Commun., 23: 1627-1637.

Ballardie, A., 1996. Scalable Multicast Key Distribution. RFC 1949.

Becker, C. and U. Wille, 1998. Communication complexity of group key distribution. 5th ACM Conference on Computer and Communications Security.

Boneh, D., 1998. The decision Diffie-Hellman problem. In Proceeding of the Third Algorithmic Number Theory Symposium (ANTS), LNCS, Portland, Oregon, USA., 1423: 48-63.

Burmester, M. and Y. Desmedt, 1994. A secure and efficient conference key distribution system. EUROCRYP, LNCS (950): 275-286.

DeCleene, B., L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan and C. Zhang, 2001. Secure group communications for wireless networks. MILCOM.

Dondeti, L.R., S. Mukherjee and A. Samal, 2000. Scalable secure one-to-many group communication using dual encryption. Comput. Commun., 23: 1681-1701.

Harney, H. and C. Muckenhirn, 1997a. Group Key Management Protocol (GKMP) Specification. RFC 2093.

Kim, Y., A. Perrig and G. Tsudik, 2000. Simple and fault-tolerant Key Agreement for Dynamic Collaborative groups. 7th ACM Conf. Comput. Commun. Security, pp: 235-244.

McGrew, D.A. and A.T. Sherman 1998. Key establishment in large dynamic groups using one-way function trees. Tech. Rep. No. 0755 (May), TIS Labs at Network Associates, Inc., Glenwood, Md.

Mittra, S., 1997. Iolus: A Framework for Scalable Secure Multicasting. ACM SIGCOMM.

Perrig, A., D. Song and J.D. Tygar, 2001. ELK, a new protocol for Efficient Large-group Key distribution. IEEE Security and Privacy Symposium.

Rafaeli, S. and D. Hutchison, 2002. Hydra: A decentralized group key management. 11th IEEE International WETICE: Enterprise Security Workshop.

Setia, S., S. Koussih, S. Jajodia and E. Harder, 2000. Kronos: A scalable group re-keying approach for secure multicast. IEEE Symposium on Security and Privacy.

Steiner, M., G. Tsudik and M. Waidner, 1996. Diffie-Hellman key distribution extended to group communication. 3rd ACM Conf. Comput. Commun. Security, pp: 31-37.

Wallner, D., E. Harder and R. Agee, 1999. Key Management for Multicast: Issues and Architectures. RFC 2627.