International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:3, 2008

A New Group Key Management Protocol for Wireless Ad-Hoc Networks

Rony H. Rahman and Lutfar Rahman

***Abstract*—**Ad hoc networks are characterized by multi-hop wireless connectivity and frequently changing network topology. 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, i.e. fixed infrastructure. With that view in mind, group key management plays an important building block of any secure group communication. The main contribution of this paper is a low complexity key management scheme that is suitable for fully self-organized ad-hoc networks. 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 analyze our protocol to show the computation and communication burden on individual nodes for key establishment.

***Abstract*—**Ad-hoc Networks, Group Key Management, Key Management Protocols, Password Authentication

I. 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. To ensure security, encryption can be used to protect messages exchanged among group members. A vital element of any encryption technique is the *cryptographic key* (also called *group key* in ad-hoc networks). In ad-hoc networks, secure distribution of the group key to all valid participants is a very big issue.

In this paper, we are going to propose an efficient *group key distribution* (most commonly known as *group key agreement*) protocol which is based on multi-party Diffie-Hellman group key exchange and which is also password-authenticated.

The rest of this paper is organized as follows. In the next section, a review of related works is given. In section III, we present the details of the various stages of the proposed protocol. Finally, we discuss some performance issues in section IV and conclude in section V.

II. Related Works

Key agreement in ad-hoc networks is divided into three main classes:

1) Centralized group key management protocols: A single entity called the Key Distribution Center (KDC) is employed for controlling the whole group.
2) Decentralized group key management protocols: The management of a large group is divided among subgroup managers, trying to minimize the problem of concentrating the work in a single place.
3) Distributed group key management protocols: There is no explicit KDC, and all the members participate in the generation of the group key and each member contributes to a portion of the key.

A. 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) [1], Logical Key Hierarchy (LKH) [2], One-way Function Tree (OFT) [3], Efficient Large-Group Key (ELK) Protocol [4] etc.

B. Decentralized Group Key Management Protocols

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 [5], Kronos [6], Intra-Domain Group Key Management (IGKMP) [7], Hydra [8] are some of the popular protocols that follow the decentralized architecture.

C. Distributed Group 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:* The protocol should try to minimize the number of iterations among the members to reduce processing and communication requirements.
- *Number of messages:* The overhead introduced by every message exchanged between members produces unbearable delays as the group grows. Therefore, the protocol should require a minimum number of messages.
- *DH key:* Identify whether the protocol uses Diffie-Hellman (DH) to generate the keys. The use of DH to generate the group key implies that the group key is generated in a contributory fashion.
- *Number of Exponentiations:* Since exponentiations impose more overhead than additions/multiplications, the number of exponentiations performed by a node should be kept to as low as possible.

Some popular protocols in this category are Burmester and Desmedt (BD) Protocol [9], Group Diffie-Hellman Key Exchange (G-DH) [10], Octopus Protocol [11], Conference Key Agreement (CKA) [12], Diffie-Hellman Logical Key Hierarchy (DH-LKH) [13], Password Authenticated Multi-Party Diffie-Hellman Key Exchange (PAMPDHKE) Protocol [14].

III. 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 (i.e. a small group of people at a conference coming together in a room for an ad hoc meeting, a small military troop deployed in a hostile environment wanting to maintain secure communication with each other etc.) and hence no pre-assumption can be made about the overall physical structure of the group. Our proposed protocol is based on the Password Authenticated Multi-Party Diffie-Hellman Key Exchange (PAMPDHKE) Protocol described in [14]. The protocol described in [14] 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. So we will try to better it by giving it some kind of structure. Also the structure of the final session key is not the same as the one described in [14].

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,…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*.

A. 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. The initial network is shown in Fig. 1 and the spanning tree constructed by applying the above protocol is shown in Fig. 2. 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.

B. 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 $1 \le g_i \le p-1$. The steps of phase I are described below:

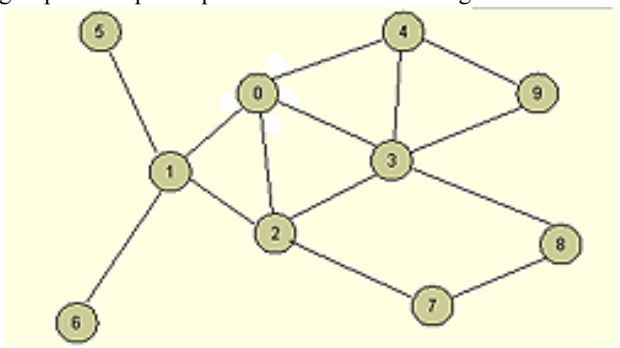


Fig. 1 The Initial Network

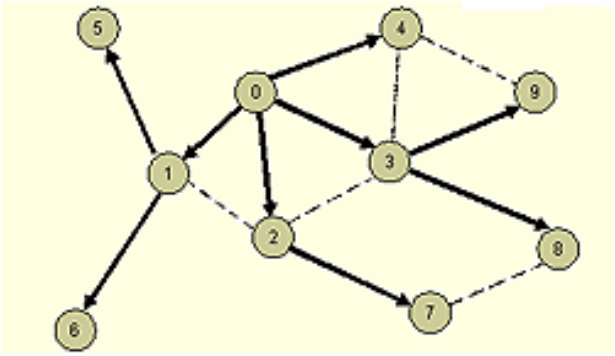


Fig. 2 The Final Spanning Tree

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 *x* 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^{g_x} and *y* is its parent, then, *x* sends to *y* a message containing the quantity α^{g_x}, i.e. $x \to y: \{\alpha^{g_x}, \alpha^{g_x}, -\}$.
- If *x* is an internal non-root node with contribution α^{g_x}, *y* is its parent 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, then, *x* sends to *y* a message containing the quantity $\{\alpha^{g_x}.A.B.C.D, \alpha^{g_x}, M'\}$, i.e.

  $x \to y: \{\alpha^{g_x}.A.B.C.D, \alpha^{g_x}, M'\}$, where,

  $$M' = \{\{M_2 \text{ part of } M_a\} \cup \{M_3 \text{ part of } M_a\} \cup \{M_2 \text{ part of } M_b\} \cup \{M_3 \text{ part of } M_b\} \cup \{M_2 \text{ part of } M_c\} \cup \{M_3 \text{ part of } M_c\} \cup \{M_2 \text{ part of etc.}\} \cup \{M_3 \text{ part of etc.}\}\}$$

  $A = M_1 \text{ part of } M_a$; $B = M_1 \text{ part of } M_b$; $C = M_1 \text{ part of } M_c$; $D = M_1 \text{ part of etc.}$
- If *x* is the root node with contribution α^{g_x} 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, then, *x* computes the final session key *K*, $K = \alpha^{g_x}.A.B.C.D$, where,

  $A = M_1 \text{ part of } M_a$; $B = M_1 \text{ part of } M_b$; $C = M_1 \text{ part of } M_c$; $D = M_1 \text{ part of etc.}$
- The quantity $\alpha^{g_x}.A.B.C.D$ indicates the product of α^{g_x}, *A*, *B*, *C* and *D*.

C. 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:

1) 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.
2) 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 $i \in \{a,b,c,etc.\}$ a message M_i^* containing the quantity $\{P(\alpha^{g_i g_{root}}), G_i, H_i, K(n_x)\}$, i.e.

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

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

  $where\ \ j \in \{a,b,c,etc.\}\ \&\ j \neq i\ [\{\alpha^{g_x}\} = \phi\ if\ x = root]$,

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

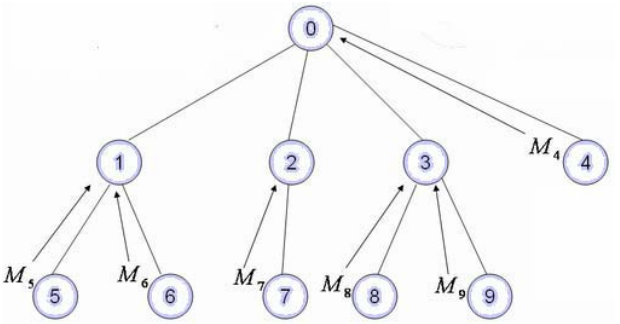
  $K(n_x) = ID\ of\ x\ encrypted\ \ by\ session\ \ key\ K$.
- If a non-root node x receives the message M_x^* and also creates the message M in phase I, then, it calculates the key K as follows,

  i) It first decrypts $\overline{M_1}$ with the weak password P to retrieve $L = \alpha^{g_x g_{root}}$.

  ii) Then it retrieves $\alpha^{g_{root}}$ by performing $L^{\frac{1}{g_x}}$.

  iii) $K = \alpha^{g_{root}}\ .s. < M_1\ part\ of\ M >\ , \forall s \in \{\overline{M_2}\ part\ of\ M_x^*\}$
- So now all the nodes have the key $K = \alpha^{\sum g_i}$, 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.

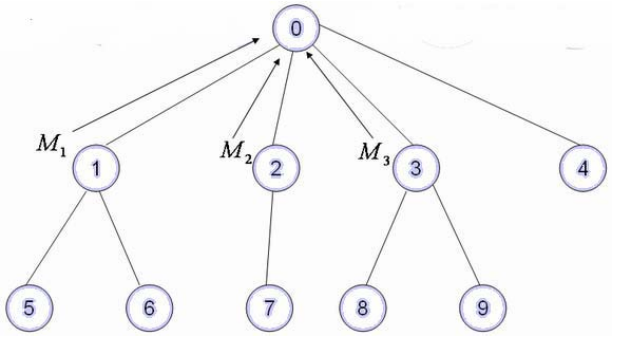
Fig. 3, Fig. 4, Fig. 5, Fig. 6 and Fig. 7 demonstrate the aforementioned phases I and phase II of the proposed protocol.



$M_5 = \{\alpha^{g_5}, \alpha^{g_5}, -\}; M_6 = \{\alpha^{g_6}, \alpha^{g_6}, -\}; M_7 = \{\alpha^{g_7}, \alpha^{g_7}, -\}$

$M_8 = \{\alpha^{g_8}, \alpha^{g_8}, -\}; M_9 = \{\alpha^{g_9}, \alpha^{g_9}, -\}; M_4 = \{\alpha^{g_4}, \alpha^{g_4}, -\}$

Fig. 3 Phase I, Round 1

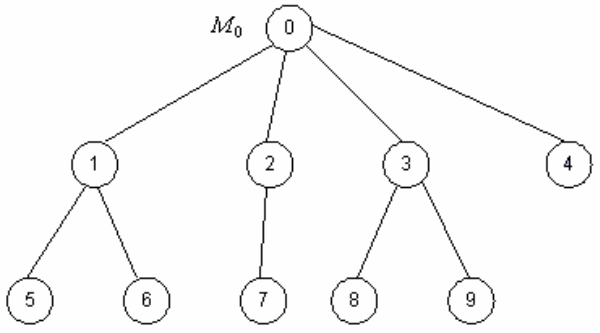


$M_1 = \{\alpha^{g_1 + g_5 + g_6}, \alpha^{g_1}, \{\alpha^{g_5}, \alpha^{g_6}\}\}$

$M_2 = \{\alpha^{g_2 + g_7}, \alpha^{g_2}, \{\alpha^{g_7}\}\}$

$M_3 = \{\alpha^{g_3 + g_8 + g_9}, \alpha^{g_3}, \{\alpha^{g_8}, \alpha^{g_9}\}\}$

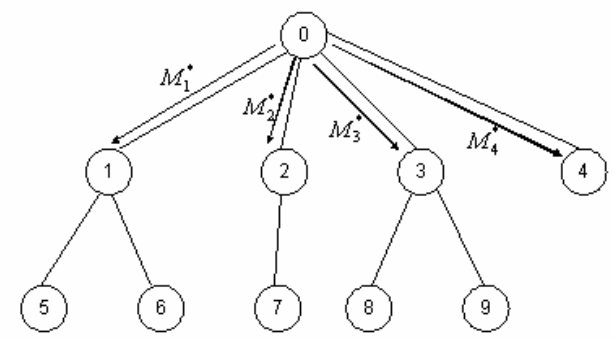
Fig. 4 Phase I, Round 2



$M_0 = \{\alpha^{g_0 + g_1 + g_5 + g_6 + g_2 + g_7 + g_3 + g_8 + g_9 + g_4}, \alpha^{g_0},$

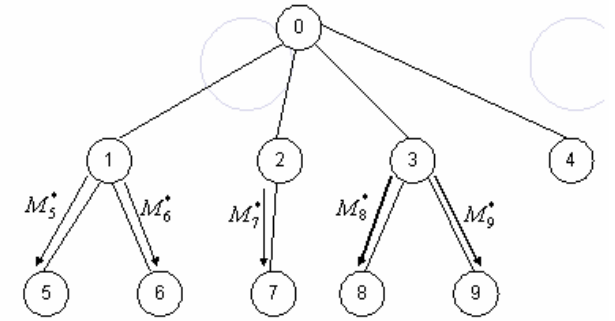
$\{\{\alpha^{g_1}, \alpha^{g_5}, \alpha^{g_6}\}, \{\alpha^{g_2}, \alpha^{g_7}\}, \{\alpha^{g_3}, \alpha^{g_8}, \alpha^{g_9}\}, \alpha^{g_4}\}\}$

Fig. 5 End of Phase I and Start of Phase II



$$M_1^* = \{P(\alpha^{g_0 g_1}), \{\alpha^{g_2+g_7}, \alpha^{g_3+g_8+g_9}, \alpha^{g_4}\}, \{\alpha^{g_0 g_5}, \alpha^{g_0 g_6}\}, K(ID_0)\}$$
$$M_2^* = \{P(\alpha^{g_0 g_2}), \{\alpha^{g_1+g_5+g_6}, \alpha^{g_3+g_8+g_9}, \alpha^{g_4}\}, \{\alpha^{g_0 g_7}\}, K(ID_0)\}$$
$$M_3^* = \{P(\alpha^{g_0 g_3}), \{\alpha^{g_1+g_5+g_6}, \alpha^{g_2+g_7}, \alpha^{g_4}\}, \{\alpha^{g_0 g_8}, \alpha^{g_0 g_9}\}, K(ID_0)\}$$
$$M_4^* = \{P(\alpha^{g_0 g_4}), \{\alpha^{g_1+g_5+g_6}, \alpha^{g_2+g_7}, \alpha^{g_3+g_8+g_9}\}, -, K(ID_0)\}$$

Fig. 6 Phase II, Round 1



$$M_5^* = \{P(\alpha^{g_0 g_5}), \{\alpha^{g_2+g_7}, \alpha^{g_3+g_8+g_9}, \alpha^{g_4}, \alpha^{g_1}, \alpha^{g_6}\}, -, K(ID_1)\}$$
$$M_6^* = \{P(\alpha^{g_0 g_6}), \{\alpha^{g_2+g_7}, \alpha^{g_3+g_8+g_9}, \alpha^{g_4}, \alpha^{g_1}, \alpha^{g_5}\}, -, K(ID_1)\}$$
$$M_7^* = \{P(\alpha^{g_0 g_7}), \{\alpha^{g_1+g_5+g_6}, \alpha^{g_3+g_8+g_9}, \alpha^{g_4}, \alpha^{g_2}\}, -, K(ID_2)\}$$
$$M_8^* = \{P(\alpha^{g_0 g_8}), \{\alpha^{g_1+g_5+g_6}, \alpha^{g_2+g_7}, \alpha^{g_4}, \alpha^{g_3}, \alpha^{g_9}\}, -, K(ID_3)\}$$
$$M_9^* = \{P(\alpha^{g_0 g_9}), \{\alpha^{g_1+g_5+g_6}, \alpha^{g_2+g_7}, \alpha^{g_4}, \alpha^{g_3}, \alpha^{g_8}\}, -, K(ID_3)\}$$

Fig. 7 Phase II, Round 2

IV. Performance Evaluation

A. 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 paper (people coming together in a conference or a military troop deployed in an hostile environment) 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. The recipients can then enter the password in his/her computer and use it as the weak shared secret for the protocol described in this paper. It may be noted that this password will never be used to encrypt data traffic. It will merely help to authenticate the nodes. Then the paper may be destroyed to remove all physical existence of the password.

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^{g_x} of each valid node x. Each contribution α^{g_x} of each valid non-root node x, is passed through the network in plaintext. 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 $\alpha^{g_{root} g_x}$. Since g_x is never exposed to the network by x, only x can extract $\alpha^{g_{root}}$ 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. And if a node is compromised by an adversary, then any session key, past, present or future, can always be obtained by that adversary. This is true for all existing security protocols (both in wired and wireless networks). Dealing with such events is beyond the scope of this paper. 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) 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 phase II of the protocol.

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. So when a valid node gets such a message in phase II, and if it can not decrypt the first part of that message, it will immediately know that the message has come from an adversary. Then the valid node can take appropriate actions. So, although there may be a delay, ultimately the active adversary will definitely be caught by a valid node in some round of phase II. One thing needs to be mentioned that our protocol is not free from DoS (Denial of Service) attacks. An active adversary can disrupt or delay the

protocol by sending huge amount of junk messages for a period of time.

B. 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. If 2 or more nodes start to construct the tree simultaneously, only the messages originating from the root node with the lowest ID (highest priority) will be considered by all other nodes. Messages originating from other root nodes will be discarded. 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 $3n-1 = O(n)$. So by amortized analysis, the number of exponentiations performed (on an average) by a single node is $O(n) / n = O(1)$. Finally, during the construction of the spanning tree, each node can send at most n request messages and at most *1* reply message. So the total number of messages during this stage is $O(n^2)$. This quantity may seem high but it is a price one must pay if one wants this protocol (or any other security protocol) to work under any circumstances and any *"truly ad-hoc"* environment. But unfortunately none of the other existing protocols take this issue 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. So we will leave out the issue of *"the communication complexity of initial setup"* when comparing with other existing protocols. But it is very clear from the above discussion that our protocol has the capability to work under and adapt to any *"truly ad-hoc"* environment.

V. Conclusion

In this paper we have proposed a new group key agreement protocol suitable for wireless ad-hoc networks of arbitrary topology. Several lines of future work 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. Finally, we conclude that group key agreement 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

[1] H. Harney, and C. Muckenhirn. "Group Key Management Protocol (GKMP) Specification". RFC 2093, 1997.
[2] D. Wallner, E. Harder, and R. Agee. "Key Management for Multicast: Issues and Architectures". RFC 2627, 1999.
[3] D. A. McGrew, and A. T. Sherman. "Key establishment in large dynamic groups using one-way function trees". Tech. Rep. No. 0755 (May), TIS Labs at Network Associates, Inc., Glenwood, Md, 1998.
[4] A. Perrig, D. Song, and J.D. Tygar. "ELK, a new protocol for Efficient Large-group Key distribution". IEEE Security and Privacy Symposium, May 2001.
[5] A. Ballardie. "Scalable Multicast Key Distribution". RFC 1949, 1996.
[6] S. Setia, S. Koussih, S. Jajodia, and E. Harder. "Kronos: A scalable group re-keying approach for secure multicast". IEEE Symposium on Security and Privacy, May 2000.
[7] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang. "Secure group communications for wireless networks". MILCOM, June 2001.
[8] S. Rafaeli, and D. Hutchison. "Hydra: a decentralized group key management". 11th IEEE International WETICE: Enterprise Security Workshop, June 2002.
[9] M. Burmester, and Y. Desmedt. "A secure and efficient conference key distribution system". EUROCRYP'94, LNCS(950):275–286, 1994.
[10] M. Steiner, G. Tsudik, and M. Waidner. "Diffie-Hellman key distribution extended to group communication". 3rd ACM Conference on Computer and Communications Security, pages 31–37, March 1996.
[11] C. Becker, and U. Wille. "Communication complexity of group key distribution". 5th ACM Conference on Computer and Communications Security, November 1998.
[12] C. Boyd. "On key agreement and conference key agreement". Information Security and Privacy: Australasian Conference, LNCS(1270):294–302, 1997.
[13] Y. Kim, A. Perrig, and G. Tsudik. "Simple and fault-tolerant Key Agreement for Dynamic Collaborative groups". 7th ACM Conference on Computer and Communications Security, November 2000.
[14] N. Asokan, and P. Ginzboorg. "Key Agreement in ad hoc networks". In Elsevier Journal of Computer Communications. Computer Communications 23 (2000) 1627 – 1637.