A three round authenticated group key agreement protocol for ad hoc networks

Daniel Augot, Raghav Bhaskar, Valérie Issarny, Daniele Sacchetti

Elsevier Editorial System(tm) for Pervasive and Mobile Computing

Manuscript Draft

Manuscript Number: PMC-D-05-00043R1

Title: A three round Authenticated Group Key Agreement Protocol for Ad hoc Networks

Article Type: Special Issue Article

Section/Category: Special Issue on Security in Wireless Mobile Computing Systems

Keywords:

Corresponding Author: Raghav Bhaskar,

Corresponding Author's Institution: INRIA

First Author: Daniel Augot

Order of Authors: Daniel Augot; Raghav Bhaskar; Valerie Issarny; Daniele Sacchetti

Manuscript Region of Origin:

A three round Authenticated Group Key Agreement Protocol for Ad hoc Networks

Daniel Augot and Raghav Bhaskar [a,*]

[a] *Projet CODES, INRIA Rocquencourt, 78153 Le Chesnay, France*

Valérie Issarny and Daniele Sacchetti [b]

[b] *Projet ARLES, INRIA Rocquencourt, 78153 Le Chesnay, France*

Abstract

Group Key Agreement (GKA) protocols enable the participants to derive a key based on each one's contribution over a public network without any central authority. They also provide efficient ways to change the key when the participants change. While some of the proposed GKA protocols are too resource consuming for the constraint devices often present in ad hoc networks, others lack a formal security analysis. In this paper, we propose a simple, efficient and secure GKA protocol well-suited to ad hoc networks and present results of our implementation of the same in a prototype application.

Key words: key agreement, ad hoc networks, provable security, cryptographic protocols.

1 Introduction

Ad hoc networks are a step closer to a pervasive world in which devices discover peer nodes and communicate with them in the absence of any central/fixed infrastructure. They find applications in a wide range of scenarios, varying from sensor networks, as in Smartdust [16], to collaborative conferencing applications as in AdhocFS [8]. The term ad hoc network has come to be employed for all networks which exhibit certain characteristics like wireless communication,

* Corresponding author. Email: raghav.bhaskar@inria.fr, Tel: +33 139635075, Fax: +33 139635051

absence of any central infrastructure, high dynamism in network composition and limited computational abilities of devices.

Before ad hoc networks can be used for critical applications, the pertinent question of security has to be solved. These networks pose additional challenges in meeting the goals of security. Challenges relate to limited computational power of devices, high communication costs, lack of any permanent trusted third party and ease of intercepting wireless communication. One essential step in securing a network is to devise a secure and efficient way of managing the security keys i.e. key management. Group Key Agreement (GKA) [26] protocols seem to provide a good solution. All the nodes in the network participate in a contributory protocol whereby they come up with a key, which is known only to the contributors. When the group composition changes (as in case of merger or partition of groups), one employs supplementary key agreement protocols to get a new key. These supplementary protocols are cheaper than executing the GKA protocol anew.

1.1 Related Work

Many Group Key Agreement protocols [15,19,4,2,32,22,23,1] have been proposed in literature, most being derived from the two-party Diffie-Hellman (DH) key agreement protocol. Some have no formal proofs while some are secure against passive adversaries only (for instance [31,22]). Provably secure protocols in a well-defined model of security were first provided by Bresson et al. [14,12,13]. Their security model extended the earlier work of Bellare et al. [6,5]. The number of rounds in these protocols is linear in the number of participants, thus making them unsuitable for large ad hoc networks. Both TGDH [22] and Dutta [17] make use of key trees, but such protocols require special ordering of the group members which is not easily achieved in ad hoc networks and make the protocol less robust to message losses. They require $O(\textit{height of tree})$ rounds of communication. Katz-Yung [21] proposed the first provably-secure constant-round group key agreement protocol inspired from the works of Burmester et al. [15]. In the same work, they also proposed a scalable compiler to transform any GKA protocol secure against a passive adversary into one which is secure against an active adversary. But with upto $3m$ broadcast messages, the protocol is quite expensive to implement in most ad hoc networks. It lacks procedures to handle group dynamism and again requires ordering of the members in a ring which is difficult to implement in ad hoc networks. Boyd et al. [10] proposed an efficient constant round protocol where the bulk of the computation is done by one participant (the current group leader), thus making it highly efficient for heterogeneous ad hoc networks. It is provably secure in the Random Oracle model [6], but lacks forward secrecy (i.e. compromise of long-term key compromises all session keys). Cata-

Table 1
Efficiency Comparison of GKA protocols

	Expo per U_i	**Rounds**	**Messages** Unicast	Broadcast	**Security**
GDH.3 [31]	3 (m for leader)	$m+1$	$2m-3$	2	Passive
TGDH [22]	$\log_2 m + 1$	$\log_2 m$	0	m	Passive
GDH.2 [13,2]	$i+1$	m	$m-1$	1	Active
Dutta [17]	$\log_3 m^\star$	$\log_3 m$	0	m	Active
Yung (BD)[21]	3	3	0	$3m$	Active
Catalano [11]	$3m$	2	0	$2m$	Active
Won [27]	2 ($2m^\dagger$ for leader)	3	$m-1$	$m+1$	Active
Ours	2 (m for leader)	3	$m-1$	2	Active

m: Number of participants
$\star$: Pairings (more expensive operation) instead of exponentiations
$\dagger$: m inverse calculations or $O(m^2)$ multiplications apart from m exponentiations

lano et. al [11] proposed a two-round protocol achieving security against active adversaries but with upto $3m$ exponentiations for each member, the protocol is way too expensive for ad hoc networks. Subsequent to the present work [1], Won et al. [27] also solve this problem but their proposition turns out to be expensive computationally. Also they use the compiler of [21] which adds to its message complexity as well. In Table 1, we compare GKA protocols achieving basic security goals of key secrecy, key independence and forward secrecy (see Section 2.1). We compare the number of exponentiations performed by each member, the number of rounds (multiple independent messages can be sent in a single round) as well as the total number of messages exchanged and mention the security level achieved by each protocol.

1.2 Our Contributions

We propose a three round authenticated GKA protocol with efficient procedures for group mergers and partitions. The protocol is shown secure against an active adversary (in the standard model) and has a *tight* security reduction. The protocol is simple (a very natural extension of the 2-party DH key agreement) and thus carries a simple proof of security. It benefits from the following features:

1) **Relevance to ad hoc networks**: This protocol is well suited to ad hoc

[1] Preliminary version of our protocol was published at TSPUC 2005 [3] which used Yung's compiler for authentication.

networks as it requires no special ordering of the participants. For each execution of the protocol, a random participant can be chosen as the group leader. It is robust as loss of messages from some participants towards the leader, does not prevent other participants from calculating the group key. It has efficient Merge and Partition procedures to handle dynamism in ad hoc networks and also provide a mechanism to change the group leader in each session. Also the bulk of the computation can be assigned to more powerful devices, as most ad hoc networks are expected to be composed of devices of unequal computing power.

2) **Simple and Efficient**: The protocol along with the merge and partition procedures is simple and efficient. It has a simple yet tightest proof of security in the standard model under the Decisional Diffie-Hellman Assumption.

1.3 Outline

The paper is organized as follows: In Section 2, we discuss the security goals, recapitulate the security model and security definitions. In Section 3, we present a new key agreement protocol for ad hoc environments and in section 4 a security analysis of the same. In Section 5, we present results of our implementation of the protocol in a prototype application. Finally, we conclude in Section 6.

2 The Security Model

In this section we define the security goals expected to be met by any GKA protocol, recapitulate the security model of Katz-Yung [21] (based on the model of [13]) and define the Decisional Diffie-Hellman (DDH) assumption.

2.1 Security Goals

The following security goals can be identified for any GKA protocol.

1) **Key Secrecy**: The key can be computed only by the GKA participants.
2) **Key Independence**: Knowledge of any set of group keys does not lead to the knowledge of any other group key not in this set (see [9]).
3) **Forward Secrecy**: Knowledge of some long term secret does not lead to the knowledge of past group keys.

2.2 The Model

The security model used to provide proof, models interaction of the real participants (modeled as oracles) and an adversary via queries which the adversary makes to the oracles. It is a kind of a "game" between the adversary and the participants, where the adversary makes some queries and finally tries to distinguish a group key from a random quantity for some session he chooses. The model is defined in details below:

Participants. The set of all potential participants is denoted by $\mathcal{P} = \{U_1, \ldots, U_l\}$ where l is polynomially bounded by the security parameter k. At any given time any subset of $\mathcal{P}$ may be involved in a GKA session. We denote this subset by an index set ($\mathcal{M}$, $\mathcal{J}$ or $\mathcal{D}$) which contains the indices of the session participants with respect to $\mathcal{P}$. Also at any given instant, a participant may be involved in more than one session. We denote by U_i^s the s^{th} instance of participant U_i and by U^s the s^{th} instance of a generic participant U. The group key associated with the instance U^s is denoted by sk_s^U. Before the first protocol run, each participant U runs an algorithm $\mathcal{G}(1^k)$ to generate public/private keys (PK_U, SK_U) which are used for the signature algorithm defined later. Each participant stores its own private key while all the public keys are made available to all participants (and the adversary).

Partners. Partners of an instance U_i^s are all the instances which calculate the same session key as U_i^s. Formally, partnering is defined by session ID (sid_U^s) and partner ID (pid_U^s) of U^s. Refer to [21] for details.

Correctness. Sessions for which all participant instances compute the same session key are admissible, all others are rejected.

Adversary. The adversary $\mathcal{A}$ interacts with the participant instances via the following queries:

- Send(U, s, M): This query essentially models the capabilities of an active adversary to send modified/fabricated messages to the participants. The message M is sent to the instance U^s and outputs the reply generated by the instance (in accordance with the protocol).

As any dynamic GKA protocol P consists of three protocols: **IKA**, **Join** and **Delete**, we define three kinds of **Execute** queries which essentially model the cababilities of a passive adversary.

- Execute$_{ika}(\mathcal{M})$: This executes the **IKA** protocol between unused instances of the specified users and returns the transcript of the session.
- Execute$_{join}(\mathcal{M}, \mathcal{J})$: This executes the **Join** protocol by adding the users indexed by $\mathcal{J}$ to an existing group indexed by $\mathcal{M}$ and returns the transcript

of the session.
- **Execute**$_{del}(\mathcal{M}, \mathcal{D})$: This executes the **Del** protocol by deleting participants indexed by $\mathcal{D}$ from the existing group indexed by $\mathcal{D}$ and returns the transcript of the session.
- **Reveal**(U, s): This query outputs the session key of instance U^s.
- **Corrupt**(U): This query outputs the long-term secret (private key) SK_U of participant U.
- **Test**(U, s): This query is allowed only once to the adversary (to be made to a $fresh$ instance; see below) during the duration of its execution. A random bit b is generated; if $b = 1$ the session key is returned to the adversary else a random bit string is returned.

Freshness. An instance U^s is $fresh$ if both of the following are true: (a) The query **Reveal** has not been made to the instance U^s or any of its partners; (b) No **Corrupt**(U) query has been asked to U or any of U^s's partners since the beginning of the game.

Definitions of security. The semantic security of a GKA protocol, P, is tested with the help of a "game" (denoted as $Game_{\mathcal{A},P}$ or in short G_0) which the adversary plays with the protocol participants. The goal of the adversary in game G_0 is to correctly guess the bit b used in answering the **Test** query. If $\mathcal{A}$ correctly guesses the bit b, we say that the event Win has occurred. Then the advantage of an adversary $\mathcal{A}$ in attacking the protocol P is defined as $\mathrm{Adv}_{\mathcal{A},P} = 2.\Pr_{\mathcal{A},P}[Win]-1$. The maximum advantage over all such adversaries making q queries and operating in time at most t is denoted as $\mathrm{Adv}_P(t, q)$. For a passive adversary we use the notation $\mathrm{Adv}_P(t, q_{ex})$, while for an active adversary we use $\mathrm{Adv}_P(t,\ q_{ex},\ q_s)$, where q_{ex} and q_s denote the number of **Execute** and **Send** queries respectively.

We note that the above model does not address the issue of malicious insiders. Session participants can be corrupted, but only after the session on which the adversary will make the **Test** query has passed. Also our definition of **forward secrecy** does not give access to internal data (any short term secrets or any data stored by the participant) to the adversary. Only the long term key is revealed. This definition is sometimes referred to as **weak forward secrecy** in literature. Achieving **strong forward secrecy** (giving access to the long term secret as well as all internal data) in GKA protocols with efficient procedures for merge and partition remains a challenge [2].

[2] Infact the only way to achieve strong forward secrecy seems to be to clear all internal data of each instance when the session key has been calculated. This makes it difficult to reuse data for efficient **Join** and **Delete** procedures.

2.3 Decisional Diffie-Hellman Assumption

The Decisional Diffie-Hellman (DDH) Assumption [7] captures the notion that the distributions $(g, g^{r_a}, g^{r_b}, g^{r_a r_b})$ and $(g, g^{r_a}, g^{r_b}, g^{r_c})$ are computationally indistinguishable, where g is a generator of some group G and r_a, r_b, r_c are randomly chosen from $[1, |G|]$. Thus the advantage of a DDH algorithm D runnning in time t for G is defined as (see [14]):

$$\mathrm{Adv}_{DDH}(t) = |\Pr[D(g, g^{r_a}, g^{r_b}, g^{r_a r_b}) = 1] - \Pr[D(g, g^{r_a}, g^{r_b}, g^{r_c}) = 1]|$$

2.4 Secure Signature Scheme

A digital signature scheme $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is defined by a triplet of algorithms:

- $\mathcal{G}$: A probabilistic key generation algorithm which on input 1^k outputs a pair of matching public and private keys (PK, SK).
- $\mathcal{S}$: An algorithm that, given a message m and a key pair $(PK,\ SK)$ as inputs, outputs a signature σ of m.
- $\mathcal{V}$: An algorithm that on input (m, σ, PK), outputs 1 if σ is a valid signature of the message m with respect to PK, and 0 otherwise.

We denote by $Succ_{\mathcal{F},\Sigma}(k)$ the probability that an adversary $\mathcal{F}$ succeeds with an existential forgery under adaptive chosen message attack ([18]). We say that a signature scheme Σ is secure if $Succ_{\mathcal{F},\Sigma}(k)$ is negligible for any probabilistic polynomial time adversary $\mathcal{F}$. We denote by $Succ_{\Sigma}(k, t)$ the maximum value of $Succ_{\mathcal{F},\Sigma}(k)$ over all adversaries $\mathcal{F}$ running in at most time t.

3 The New Group Key Agreement Protocol

We propose a new group key agreement protocol in this section. We first illustrate the basic principle of key exchange, followed by a detailed explanation of how it is employed to derive Initial Key Agreement, Join/Merge and Delete/Partition procedures for ad hoc groups.

3.1 Notation

G: A subgroup (of prime order q with generator g) of some group.
U_i: i^{th} participant amongst the n participants in the current session.
U_l: The current group leader ($l \in \{1, ...n\}$).

r_i: A random number (from $[1, q-1]$) generated by participant U_i. Also called the *secret* for U_i.
g^{r_i}: The *blinded secret* for U_i.
$g^{r_i r_l}$: The *blinded response* for U_i from U_l.
$\mathcal{M}$: The set of indices of participants (from $\mathcal{P}$) in the current session.
$\mathcal{J}$: The set of indices of the joining participants.
$\mathcal{D}$: The set of indices of the leaving participants.
$x \leftarrow y$: x is assigned y.
$x \xleftarrow{r} \mathcal{S}$: x is randomly drawn from the uniform distribution S.
$U_i \longrightarrow U_j : \{M\}$: U_i sends message M to participant U_j.
$U_i \xrightarrow{B} \mathcal{M} : \{M\}$: U_i broadcasts message M to all participants indexed by $\mathcal{M}$.
N_i: Random nonce (maximum k_1 bits) generated by participant U_i.

3.2 A Three Round Protocol

Please note that in the following rounds each message is digitally signed by the sender (σ_i^j is signature on message msg_i^j in Tables 2-4) and is verified by the receiver before following the protocol.
Protocol Steps:
Round 1: The chosen group leader, M_l makes a initial request ($INIT$) with his identity, U_l and a random nonce N_l to the group $\mathcal{M}$.
Round 2: Each interested M_i responds to the $INIT$ request, with his identity U_i, nonce N_l and a blinded secret g^{r_i} to M_l (see Table 2 for exact message contents).
Round 3: M_l collects all the received blinded secrets, raises each of them to its secret (r_l) and broadcasts them along with the original contributions to the group, i.e. it sends $\{U_i, N_i, g^{r_i}, g^{r_i r_l}\}$ for all $i \in \mathcal{M} \setminus \{l\}$.

Key Calculation: Each M_i checks if its contribution is included correctly and obtains g^{r_l} by computing $(g^{r_i r_l})^{r_i^{-1}}$. The group key is

$$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}.$$

Note:
1) The original contributions g^{r_i} are included in the last message as they are required for key calculation in case of group modifications (see below).
2) Even though $\Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l}$ is publicly known, it is included in key computation, to derive a key composed of everyone's contribution. This ensures that the key is not pre-determined and is unique to this session.
3) Even though the current group leader chooses his contribution after others, he cannot pre-determine the group key. See section 4.1 for details.

The protocol is formally defined in Table 2. We now see how this protocol can

Round 1

$l \stackrel{r}{\leftarrow} \mathcal{M}, N_l \stackrel{r}{\leftarrow} \{0,1\}^k$

$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l^1 = \{INIT, U_l, N_l\}, \sigma_l^1\}$

Round 2

$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_l}\{msg_l^1, \sigma_l\} == 1)$, $r_i \stackrel{r}{\leftarrow} [1, q-1], N_i \stackrel{r}{\leftarrow} \{0,1\}^k$,

$U_i \longrightarrow U_l : \{msg_i = \{IREPLY, U_l, N_l, U_i, N_i, g^{r_i}\}, \sigma_i\}$

Round 3

$r_l \stackrel{r}{\leftarrow} [1, q-1], \forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)$

$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l^2 = \{IGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$

Key Computation

$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} is as contributed

$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 2
IKA

be used to derive IKA, Join/Merge and Delete/Partition procedures for ad hoc networks.

3.3 Initial Key Agreement

Secure ad hoc group formation procedures typically involve peer discovery and connectivity checks before a group key is derived. Thus, an $INIT$ request is issued by some participant and all interested peers respond. The responses are collected and connectivity checks are carried out to ensure that all participants can listen/broadcast to the group (see for instance [29]). After the group membership is defined, GKA procedures are implemented to derive a group key. Such an approach is quite a drain on the limited resources of ad hoc network devices. Thus an approach which integrates the two separate procedures of group formation and group key agreement is required. The above protocol fits well with this approach. Round 1 and Round 2 of the above protocol can be incorporated into the group formation procedures. In this way, blinded secrets, g^{r_i}'s, of all potential members, U_i's, are collected before the group composition is defined. When the fully connected ad hoc group is defined, a single message (Round 3 in Table 2) from the group leader, U_l [3], using contributions of only the joining participants enables every participant to compute the group key. An example is provided below.

[3] The group leader can be different from the initiator; see section 4.2 for leader election issues.

Round 1

$\forall i \in \mathcal{J}, r_i \xleftarrow{r} [1, q-1], N_i \xleftarrow{r} \{0,1\}^k,$

$U_i \xrightarrow{B} \mathcal{M} : \{msg_i = \{JOIN, U_i, N_i, g^{r_i}\}, \sigma_i\}$

Round 2

$\forall i \in \mathcal{J}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)\ r_l \xleftarrow{r} [1, q-1], l' \xleftarrow{r} \mathcal{M} \cup \mathcal{J}$

$U_l \longrightarrow U_{l'} : \{msg_l = \{JREPLY, \{U_i, N_i, g^{r_i}\}_{\forall i \in \mathcal{M} \cup \mathcal{J}}\}, \sigma_l\}$

Round 3

$if(\mathcal{V}_{PK_i}\{msg_l, \sigma_l\} == 1), l \leftarrow l', r_l \xleftarrow{r} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{J}$

$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^2 = \{JGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$

Key Computation

$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} is as contributed

$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 3
Join/Merge

Suppose U_1 initiates the group discovery and initially 5 participants express interest and send g^{r_2}, g^{r_3}, g^{r_4}, g^{r_5} and g^{r_6} respectively along with their identities and nonces. Finally only 3 join because of the full-connectivity constraint. Suppose the participants who finally join are U_2, U_4 and U_5. Then the group leader, U_1, broadcasts the following message: $\{g^{r_2}, g^{r_4}, g^{r_5}, (g^{r_2})^{r_1}, (g^{r_4})^{r_1}, (g^{r_5})^{r_1}\}$. On receiving this message, each participant can derive g^{r_1} using his respective secret. Thus the key $g^{r_1(1+r_2+r_4+r_5)}$ can be computed.

3.4 Join/Merge

Join is quite similar to **IKA**. Each joining participant, $U_i (i \in \mathcal{J})$, sends a $JOIN$ request along with its identity, U_i, random nonce, N_i and blinded secret, g^{r_i}. The old group leader (U_l) chooses a new random secret, r_l, and sends all the blinded secrets to the new group leader, $U_{l'}$, (which can be chosen randomly). The new group leader broadcasts a message similar to the round 3 message in **IKA** i.e. all the blinded secrets and the blinded secrets raised to his (new) secret. It is worth noting that the new group leader discards the secret he used during the $JOIN$ request (or secret from last session) and generates a new random secret for the broadcast message. During merge of two groups, all members of the smaller merging group (including the group leader) can be seen as joining members to the larger group. See Table 3 for formal specification and below for an example.

Round 1
$\forall i \in \mathcal{D}, U_i \longrightarrow U_l : \{msg_i = \{DEL, U_i, N_i\}, \sigma_i\}$
Round 2
$\forall i \in \mathcal{D}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1), r_l \stackrel{r}{\leftarrow} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \setminus \mathcal{D}$
$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l = \{DGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l, \sigma_l\} == 1)$ and g^{r_i} is as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 4
Delete/Partition

Suppose new participants, U_9 and U_{10} join the group of U_1, U_2, U_4 and U_5 with their contributions g^{r_9} and $g^{r_{10}}$ respectively. Then the previous group leader (U_1) changes its secret to r_1^* and sends $g^{r_1^*}$, g^{r_2}, g^{r_4}, g^{r_5}, g^{r_9}, $g^{r_{10}}$ to U_{10} (say the new group leader). U_{10} generates a new secret r_{10}^* and broadcasts the following message to the group: $\{g^{r_1^*}, g^{r_2}, g^{r_4}, g^{r_5}, g^{r_9}, g^{r_{10}^* r_1^*}, g^{r_{10}^* r_2}, g^{r_{10}^* r_4}, g^{r_{10}^* r_5}, g^{r_{10}^* r_9}\}$. And the new key is $g^{r_{10}^*(1+r_1^*+r_2+r_4+r_5+r_9)}$.

3.5 Delete/Partition

When participants leave the group, the group leader changes his secret contribution and sends an **IKA** Round 3 like message to the group, omitting the leaving participants' contributions. Partition of a group can be see as deletion of multiple members. Refer to Table 4 and below for an example.

Suppose a participant, U_2, leaves the group of U_1, U_2, U_4, U_5, U_9 and U_{10}. Then the leader, U_{10} changes its secret to $r_{10}^{''}$ and broadcasts $\{g^{r_1^*}, g^{r_4}, g^{r_5}, g^{r_9}, (g^{r_1^*})^{r_{10}^{''}}, (g^{r_4})^{r_{10}^{''}}, (g^{r_5})^{r_{10}^{''}}, (g^{r_9})^{r_{10}^{''}}\}$ to the group. And the new key is $g^{r_{10}^{''}(1+r_1^*+r_4+r_5+r_9)}$.

4 Security Analysis

Below we show that the above defined protocol is secure against active adversaries in the standard security model.

Theorem 1: Let P be the protocol as defined in the last section. Let $\mathcal{A}$ be an active adversary making q_{ex} **Execute** queries and q_s **Send** queries to the

participants and running in time t. Then Protocol P is a secure GKA protocol. Namely:

$$\mathrm{Adv}_P(t, q_{ex}, q_s) \leq \mathrm{Adv}_{DDH}(t') + |\mathcal{P}| * Succ_{\Sigma}(k, t') + \frac{q_s^2 + q_{ex} q_s}{2^{k_1}}$$

where k_1 is the size (in bits) of the nonces and $t' \leq t + |\mathcal{P}|(q_{ex} + q_s)(t_{exp} + t_{lkup})$, t_{exp} is the time to perform an exponentiation in G and t_{lkup} is the time to perform a look-up in tables L and $Sessions$, to be defined in the proof.

Proof: Let $\mathcal{A}$ be an adversary that plays in the game G_0 against the protocol P. We will define a series of games $G_1, \ldots G_6$ such that each game G_i differs "slightly" from its precedent game G_{i-1}. We denote the event Win in the game G_i by Win_i. Thus by explicitly quantifying the effect the slight difference in the games has on the winning probability of the adversary, one can relate the winning probability of $\mathcal{A}$ in the original game G_0 ($\Pr[Win_0]$) to any other game. We stop when we eventually reduce to a simple game (here G_6) for which we can calculate $\Pr[Win_i]$. Thus by relating all the probabilities we can eventually calculate $\Pr[Win_0]$.

All queries made by $\mathcal{A}$ are answered by a simulator Δ. Δ maintains two tables: $Sessions$ and L. In table $Sessions$, Δ keeps transcripts of each and every session generated by him (either with a single **Execute** query or multiple **Send** queries). While in table L, Δ maintains a list of all *blinded secrets* generated by him during the game and their corresponding secrets.

Game G_0: This game G_0 is the real game as defined earlier. Δ initializes the game by generating public-private key pairs for all the participants as specified by the protocol and choosing a random bit b, which is used by him to answer the **Test** query. Then it answers all queries of the adversary in accordance with the protocol P.

Game G_1: The game G_1 is identical to G_0 except that Δ aborts if a signature forgery occurs for some player U before any **Corrupt(U)** query was made. We denote such an event by E_1. Using a well-know lemma we get: $|\Pr[Win_0] - \Pr[Win_1]| \leq \Pr[E_1]$. Note that Δ can detect a signature forgery for some player U when he finds a valid message, not generated by him (all messages generated by Δ are stored in the $Sessions$ table), in some session before the **Corrupt** query was made to U.

Calculation of $\Pr[E_1]$: The event E_1 occurs when the adversary makes an existential signature forgery for any one of the protocol participants. The probability of this happening is bounded by$|\mathcal{P}| * Succ_{\mathcal{F},\Sigma}(k)$ where $Succ_{\mathcal{F},\Sigma}(k)$ is the success probability of an existential signature forgery against a signature scheme Σ, given some public key PK.

Game G_2: The game G_2 is identical to G_1 except that Δ aborts if a nonce

used in some **Send** query has already been used in some **Execute** or **Send** query before. We denote the occurrence of the nonce being repeated in some **Send** query as event E_2. Then: $|\Pr[Win_1] - \Pr[Win_2]| \leq \Pr[E_2]$. Δ can detect event E_2 as he can track all nonces generated, via the *Sessions* table.

Calculation of $\Pr[E_2]$: Clearly the probability of event E_2 happening is $\frac{q_s(q_{ex}+q_s)}{2^{k_1}}$.

Game G_3: In game G_3, Δ modifies the way it answers the queries slightly. Δ chooses a DDH-tuple (g, g^{r_a}, g^{r_b}, $g^{r_a r_b}$). Δ follows the protocol as before to generate query responses but changes the way it generates the *blinded secrets* used in the transcript. The change is as follows:

Whenever a *blinded secret* is to be generated for some session participant M_i, instead of raising the group generator g to a randomly chosen number r_i (from $[1, q-1]$, as specified by the protocol), it raises g^{r_b} (from the given tuple) to r_i. Thus, in brief, Δ uses the value $g^{r_b r_i}$ as *blinded secret* for participant M_i in the transcript. The corresponding *blinded response* is generated as before by raising the *blinded secret* to the group leader's secret r_l (which is also randomly chosen from $[1, q-1]$, as specified by the protocol). Also, Δ stores the *blinded secret* so generated and the corresponding r_i in table L. In this way, Δ knows all *blinded secrets* generated by him during the game and their corresponding *secrets*. Clearly from the adversary's point of view there is no change in the game. Thus $|\Pr[Win_2] = \Pr[Win_3]|$.

Game G_4: Game G_4 is same as game G_3 except that Δ modifies the way it generates the *blinded responses*. Using the same DDH-tuple (g, g^{r_a}, g^{r_b}, $g^{r_a r_b}$), Δ does the following:

Whenever a *blinded response* is to be generated for some session participant M_i, Δ retrieves the corresponding *blinded secret* g^{r_i} from the table *Sessions*. Then it looks for this *blinded secret* in table L. If Δ finds it in the table, it retrieves the corresponding *secret* entry (r_i) and raises $g^{r_a r_b}$ (from the DDH-tuple) to it to get $g^{r_a r_b r_i}$. It further raises it to the *secret*, r_l (randomly chosen from $[1, q-1]$), of the group leader. The resulting value $g^{r_a r_b r_i r_l}$ is used as the *blinded response* for participant M_i in the session transcript. If on the other hand, Δ does not find g^{r_i} in table L, this means that this *blinded secret* has been introduced by the adversary $\mathcal{A}$ and Δ does not know the corresponding secret. Thus for a session where any of the *blinded secrets* is not found in table L, Δ continues to generate the *blinded responses* (for all the participants) as in game G_3 (by raising *blinded secret* to the *secret* of the leader).

Thus, in brief, Δ uses the value $g^{r_a r_b r_i r_l}$ as the *blinded response* for participant M_i, if all *blinded secrets* in that session were generated by him otherwise it uses the value $g^{r_b r_i r_l}$. Note that as $\mathcal{A}$ can make a **Test** query only on a $fresh$

participant instance, this rules out those sessions where $\mathcal{A}$ has been able to introduce *blinded secrets* on his own (by asking the **Corrupt** query). Thus Δ can respond to the queries of such sessions without using data from the DDH-tuple [4].

Clearly again from the adversary's point of view there is no change in the game. Thus $|\Pr[Win_3] = \Pr[Win_4]|$.

Game G_5: Game G_5 is same as Game G_4 except that instead of a DDH-tuple $(g, g^{r_a}, g^{r_b}, g^{r_a r_b})$, Δ chooses a random tuple $(g, g^{r_a}, g^{r_b}, g^{r_c})$. Δ continues answering the queries as in Game G_4, except that the role of $g^{r_a r_b}$ is taken by g^{r_c}. And now when answering the **Reveal** query or **Test** query (in case bit $b = 1$), Δ uses g^{r_a} in computing the session key instead of $g^{\frac{r_c}{r_b}}$ which the protocol demands. Thus the only difference between games G_4 and G_5 is the computational distance between a DDH-tuple and a random tuple, therefore: $|\Pr[Win_4] - \Pr[Win_5]| \leq \mathrm{Adv}_{DDH}(t')$.
where t' is bounded by $t + |\mathcal{P}|(q_{ex} + q_s)(t_{exp} + t_{lkup})$, t_{exp} being the time to perform an exponentiation in G and t_{lkup} the time to perform a look-up in tables L and $Sessions$.

Game G_6: Game G_6 is same as Game G_5 except that irrespective of the value of the bit b, Δ answers the **Test** query with a random value. Then clearly, $\Pr[Win_6] = \frac{1}{2}$. Also $\Pr[Win_5] = \Pr[Win_6]$ because while in Game G_5, Δ answers with $g^{r_a r_l}$ as a response to the **Test** query, in Game G_6 Δ answers with g^{random}. But essentially both responses are uniformly distributed in G.

Combining all the above results, we get: $\Pr[Win_0] \leq \Pr[E_1] + \Pr[E_2] + \mathrm{Adv}_{DDH}(t') + \frac{1}{2}$ and so the desired result follows.

4.1 Key Control

The fact that any of the participants in a group key agreement protocol cannot pre-determine the key before the actual execution of the protocol, makes it different from a key transport protocol. Although the group leader in our protocol chooses its contribution after other members have chosen theirs, it does not imply the group leader can pre-determine the group key. Infact like many other protocols (including GDH.2, GDH.3, TGDH, won in table 1) the group member choosing last his contribution might have some advantage but it does not translate to key control as discussed in [1,28]. We show below that the group leader in our protocol cannot fix the group key to a given value K_f. The group key $g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$ can be viewed as a two-party Diffie-Hellman

[4] As Δ does not have to guess the query for which he wants to use data from the DDH-tuple, there is no q_s or q_{ex} factor in the final security reduction.

key where one participant's contribution is g^{r_l} and the other's $g^{(1+\sum_{i\in\mathcal{M}\setminus\{l\}} r_i)}$. Denoting $g^{(1+\sum_{i\in\mathcal{M}\setminus\{l\}} r_i)}$ by g^B, the group leader needs a polynomial time algorithm $\mathcal{A}$ which given the desired group key $K_f = g^{r_K}$ and g^B as inputs can output $g^{\frac{r_K}{B}}$ to be used as g^{r_l} i.e. $\mathcal{A}(g^{r_K}, g^B) = g^{\frac{r_K}{B}}$. But infact this algorithm can be used to solve the computational Diffie-Hellman problem as follows: Given g^α and g^β, $\mathcal{A}(g^\alpha, g^\beta) = g^{\frac{\alpha}{\beta}}$; $\mathcal{A}(g^{\frac{\alpha}{\beta}}, g^\alpha) = g^{\frac{1}{\beta}}$; $\mathcal{A}(g^\alpha, g^{\frac{1}{\beta}}) = g^{\alpha\beta}$.

We can do better by requiring the group leader to commit to its contribution before others as follows: The current group leader sends also the hash value of his contribution in the $INIT$ message ($H(g^{r_l})$ in table 5). Thus the group leader is no longer at any advantage. Other kinds of attacks influencing the key including attacks by collusion of malicious insiders have not been well studied till now. Very recent work of Katz and Shin [20] is the first attempt to formally model attacks by a collusion of malicious insiders. Thus it is of interest in the future to provide security proofs in this new evolving model. A simple but costly way to detect such kind of attacks is to add a round of *key confirmation*; where in a single round of broadcasts, each participant broadcasts a well-known public quantity encrypted with the current session key.

4.2 Group leader election

Group leader election is a non-trivial issue in asynchronous networks like ad hoc networks. A lot of literature exists on this issue; see for example [25,30]. Leader election protocols that ensure that a single leader is elected at the end of the protocol run, can be quite expensive to implement (requiring several rounds of communication). So if a mechanism exists for merging multiple groups into a single group with a single leader, much simpler leader election protocols can be employed for the sake of efficiency. We choose to use an auto-election mechanism to choose a group leader, wherein: In the absence of a group leader, each node wishing to form a group sets a random timer. If by the expiry of this timer, no $INIT$ message is received, the node issues an $INIT$ message of its own. Thus other nodes can reply to this $INIT$ request. If multiple $INIT$ messages are received by a node, a simple rule (like the initiator with a lower ID U_i, or the initiator with a larger group) can help the node to decide which $INIT$ message to reply to. Thus multiple groups can exist in the network at the same time (with potentially some common members). If total connectivity is ensured between these groups, it is possible to merge them easily as well.

Table 5
Modified IKA

Round 1

$l \xleftarrow{r} \mathcal{M}, N_l \xleftarrow{r} \{0,1\}^k$, $r_l \xleftarrow{r} [1, q-1]$

$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^1 = \{INIT, U_l, N_l, H(g^{r_l})\}, \sigma_l^1\}$

Round 2

$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_l}\{msg_l^1, \sigma_l\} == 1)$, $r_i \xleftarrow{r} [1, q-1], N_i \xleftarrow{r} \{0,1\}^k$,

$U_i \longrightarrow U_l : \{msg_i = \{IREPLY, U_l, N_l, U_i, N_i, g^{r_i}\}, \sigma_i\}$

Round 3

$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)$

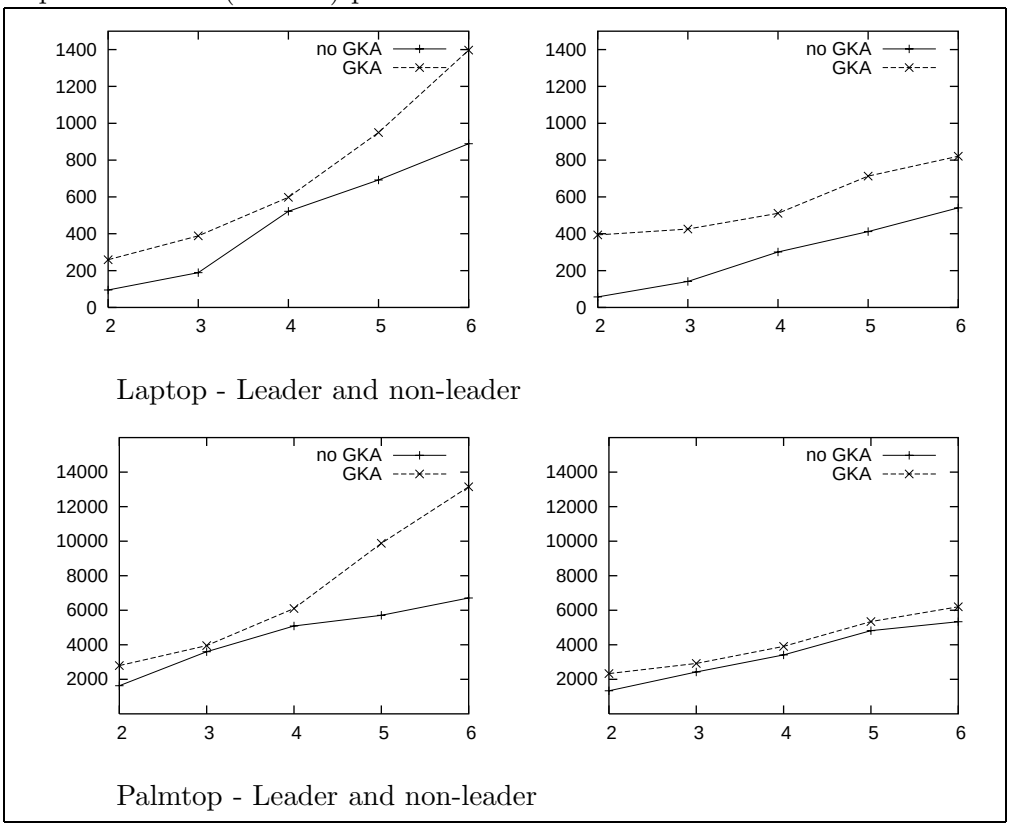
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^2 = \{IGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$

Key Computation

$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} is as contributed and hash value of g^{r_l} matches that sent in **Round 1**

$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 6
Computation time (in msec) per device with and without GKA



5 Implementation

To test the performance of this new GKA protocol, we incorporated it in the group management protocol of [8]. The group management of [8] consists of three communication rounds: $DISC$, $JOIN$ and $GROUP$. The $DISC$ stage initiates the group formation by calling for interested participants. Each interested participant responds with a $JOIN$ message. The group membership is defined and announced by the group leader (chosen randomly) by the $GROUP$ message. The design of the new GKA protocol allowed us to piggy-back GKA data on group management messages, thus member contributions towards the group key are collected during $JOIN$ messages while the $GROUP$ message carries the message from the group leader which enables everyone to compute the group key. Thus no additional communication round is required to derive a group key, irrespective of the group size. It is worth mentioning that it would not have been possible with most of the protocols presented in table 1, as the messages sent by group members are dependent on messages sent by other members. A comparison of the computation times on a device in the absence and presence of GKA procedures is plotted in table 6. The data shown is for an experimental setup consisting of laptops (Compaq 500 Mhz running Linux) and palmtops (Compaq ipaq 400MHz running Linux familiar 0.7). All random contributions for the group key were chosen from a Diffie-Hellman group of prime order of 1024 bits. The code was written in Java except the exponentiation function which was implemented in native code with the GMP library [24]. The graphs in table 6 plot computation time (in milliseconds on Y axis) against group-size with and without GKA. There are separate plots for the cases when the device was a leader/non-leader. Leader for group management was randomly chosen. As expected, the time for non-leader members increases (when employing GKA protocol) by an almost constant factor (order of time to perform two 1024 bit exponentiations), while for a leader it increases linearly as the group size increases. As most ad hoc networks are expected to be composed of devices of unequal computing power, more powerful devices (like laptops) can assume the role of a leader more often. Use of elliptic curve groups can lead to much better computation times.

6 Conclusion

We have proposed a new group key agreement protocol, particularly well suited to ad hoc networks. It is efficient in the number of rounds (only three rounds, the first two rounds may be executed along with group management procedures), and also efficient in computational terms. It requires no special ordering of the participants. Any participant can be possibly chosen as the group leader for one session and the role can be easily rotated amongst the other

participants in latter rounds. The key, thus derived, is independent of keys in other sessions. Long-term secrets are used for authentication purposes only, thus providing *weak forward secrecy*. Achieving *strong forward secrecy* without compromising on efficiency of $JOIN$ and $DELETE$ procedures is an interesting area for future work.

The protocol is proved secure against active adversaries in the framework of [13], in the Standard model and with the Decisional Diffie-Hellman (DDH) assumption in any group. The protocol has one of the tightest reductions to the DDH assumption amongst group key agreement protocols.

References

[1] N. Asokan and P. Ginzboorg. Key agreement in ad-hoc networks. *Computer Communication Review*, 23(17):1627–1637, Nov 2000.

[2] G. Ateniese, M. Steiner, and G. Tsudik. New multiparty authentication services and key agreement protocols. *IEEE Journal of Selected Areas in Communications*, 18(4):628–639, April 2000.

[3] D. Augot, R. Bhaskar, V. Issarny, and D. Sacchetti. An efficient group key agreement protocol for ad hoc networks. In *IEEE Workshop on Trust, Security and Privacy in Ubiquitous Computing*. IEEE CS Press, 2005.

[4] K. Becker and U. Wille. Communication complexity of group key distribution. In *CCS '98: Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 1–6. ACM Press, 1998.

[5] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology - EUROCRYPT '00*, pages 139–155. LNCS 1807, 2000.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93: Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[7] D. Boneh. The Decision Diffie-Hellman problem. In *ANTS-III: 3rd Algorithmic Number Theory Symposium*, pages 48–63. LNCS 1423, 1998.

[8] M. Boulkenafed and V. Issarny. AdHocFS: Sharing files in WLANs. In *2nd International Symposium on Network Computing and Applications*, pages 156–163. IEEE Computer Society, 2003.

[9] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2003.

[10] C. Boyd and J.M.G. Nieto. Round-optimal contributory conference key agreement. In *Public Key Cryptography '03*, pages 161–174. LNCS 2567, 2003.

[11] E. Bresson and D. Catalano. Constant round authenticated group key agreement via distributed computation. In *Proceedings of Public Key Cryptography*, pages 115–119. LNCS 2567, 2004.

[12] E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group Diffie-Hellman key exchange - the dynamic case. In *Advances in Cryptology - ASIACRYPT '01*, pages 290–309. LNCS 2248, 2001.

[13] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie Hellman key exchange under standard assumptions. In *Advances in Cryptology - EUROCRYPT '02*, pages 321–326. LNCS 2332, 2002.

[14] E. Bresson, O. Chevassut, D. Pointcheval, and J.J. Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *CCS '01: Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 255–264. ACM Press, 2001.

[15] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology - EUROCRYPT '94*, pages 275–286. LNCS 950, 1994.

[16] Smart Dust. http://robotics.eecs.berkeley.edu/~pister/smartdust.

[17] R. Dutta and R. Barua. Dynamic group key agreement in tree-based setting. In *ACISP*, pages 101–112, 2005.

[18] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[19] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology - ASIACRYPT '96*, pages 36–49. LNCS 1163, 1996.

[20] J. Katz and J.S. Shin. Modelling insider attacks on group key-exchange protocols. http://eprint.iacr.org/2005/163.pdf.

[21] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange - full version. In *Advances in Cryptology - CRYPTO '03*, pages 110–125. LNCS 2729, 2003.

[22] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *CCS '00: Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 235–244. ACM Press, 2000.

[23] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, July 2004.

[24] GNU Multi Precision Arithmetic Library. http://www.swox.com/gmp.

[25] N. Malpani, J.L. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of Dial M Workshop*, pages 96–103. ACM, 200.

[26] A. J. Menezes, P. C. van Oorschot, and S. Vanstone. *HandBook of Applied Cryptography*. CRC Press, 1996.

[27] J. Nam, J. Lee, S. Kim, and D. Won. DDH based group key agreement for mobile computing. http://eprint.iacr.org/2004/127, 2004.

[28] J. Pieprzyk and H. Wang. The key control in multi-party key agreement protocols. In *Proceedings of Workshop on Coding, Cryptography and Combinatorics*, pages 277–288. PCS, Birkhauser, 2004.

[29] G-C. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 381–388. IEEE Computer Society, 2001.

[30] G. Singh. Leader election in complete networks. *SIAM Journal of Computing*, 26(3):772–785, June 1997.

[31] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to groups. In *ACM Conference on Computer and Communications Security*, pages 31–37. ACM Press, 1996.

[32] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, August 2000.

Author Biography- AUGOT

Daniel was a student in Mathematics at the \'Ecole Normale Sup\'erieure de Fontenay-aux-Roses. Daniel Augot receive his PHD in Computer Science, where he proposed to find minimum weight codewords of cyclic codes by the mean of the resolution of algebraic systems. Daniel Augot is researcher at the French National for Research in Computer Science and Control, inside the Codes team.

His research interest are Coding and Cryptography, and the interactions between these two fields.

*** Author Photo- AUGOT**

Click here to download high resolution image

*** Author Biography- BHASKAR**

Raghav did his Masters of Technology in Mathematics and Computing from the Indian Institute of Technology, Delhi, India in 2001. After brief stints with Novell Software and British Telecom Research, he started his PhD at INRIA, Rocquencourt, France in 2003.

His research interests are cryptogtaphy, security protocols and ad hoc networks.

*** Author Photo- BHASKAR**

Click here to download high resolution image

*** Author Biography- ISSARNY**

Valerie Issarny got her PhD and her "habilitation a diriger des recherches" in computer science from the University of Rennes I in 1991 and 1997,respectively. She is currently senior research scientist at INRIA. Since 2002, she is the head of the ARLES INRIA research project-team at INRIA-Rocquencourt. Her research interests relate to distributed systems, software architectures, mobile systems and middleware. She is chairing the executive committee of the AIR&D consortium on Ambient Intelligence Research and Development. Further information about Valerie's research interest and her publications can be obtained from http://www-rocq.inria.fr/arles/members/issarny.html

*** Author Biography- SACCHETTI**

Daniele Sacchetti was a student in Computer Science at the Computer Science Faculty of Bologna where he received his master degree in Computer Science. Daniele is a Research Engineer at the France National Institute for Research in Computer Science and Control, inside the Arles team.

His research interests are Middleware and Software Architectures for Mobile Distributed Systems.

*** Author Photo- SACCHETTI**

Click here to download high resolution image

*** Response to Reviewers**

List of Changes

---

1) Table 1 has now comparison with more protocols. It has been explained in Section 1.1, that the requirement of ordering of the participants is the reason why most protocols are difficult to implement in ad hoc networks.

2) The Security Model (Section 2.2) has been refined.

3) The authenticated version of our protocol is presented in Section 3.2. We no longer use the katz-Yung compiler. Thus not only is the protocol more efficient (no first round of n braodcasts is needed), the security proof has a much tighter reduction.

4) Section 4 has the new proof of this authenticated version of the protocol. Section 4.1 and Section 4.2 have been added to address key control and leader election issues.

Responses to Reviewer's comments (begin by --)

Reviewer 1

1) The authors seems to be unaware of work in the literature. Specially, the following two papers, provide a authenticated, distributed, contributory and provably secure (against both active and passive attacks) group key agreement protocol.

(a) N. Aoskan and P. Ginzborg "Key agreement in ad-hoc networks". Computer communication review 2000.

(b) G. Ateniese, M. Steiner and G. Tsudik "New Multi-party Authentication Services and Key Agreement Protocols" IEEE JSAC, special issue on Secure Communication, May 2000.

Authors claim that the current protocols are not efficient and suitable for ad-hoc without commenting on the work in (a) and (b).

-- (a) does mostly a review of existing protocols. Password based GKA protocols are unsuitable as they make an assumption that all nodes share a common password before hand which is unrealistic. All provably-secure protocols have been considered in thge comparison. We do not include some protocols which have no proof of security. (b) was already referenced in the paper.

2) The protocol presented uses a more or less a centralized approach according to the reviewer. The group leader acts as a central server and can influence the generation
of group key with a big share. In fact, if nodes in the group are willing to
believe the group leader and are 'comfortable' with high-share of group leader in the
group-key, then a simple protocol where a group leader generates a group key and
exchanges it with other members using a two-party authenticated DH key exchange is very
efficient and offers the same security as the proposed protocol. In GKA protocols proposed
in (a) and (b), the group leader can note influence the group key and for this reason
the protocols are contributory in nature. Further, the leader is the central point of failure, a malicious node can jam the radio range of the leader and node may never establish a group key.

-- It is shown in the paper that the group leader cannot influence the security of the group key. As long as there is one random contribution generated by one participant the key is random enough. Also the leader has no special authority about it and can be easily changed in each session.

Suggestions for improvement:

1) Comparison with reference (a) & (b): it would be nice to have a section comparing the proposed protocol and protocols presented in (a) and (b). The comparison should include differences in the key generation scheme, security, communication complexity and computation complexity.

-- Has been included.

2) Justification of leader key-share: the paper should include a paragraph justifying why the leaders' share on the group key does not influence the security of group key.

-- A new subsection (4.1) has been added.

3) Selection of group leader: the performance of the protocol depends on the selection of the group leader, random selection of a group leader may not converge the protocol. How do nodes select a unique leader in an ad-hoc environment? how would the protocol react if multiple nodes act as group leaders or would the protocol converge if there are concurrent INIT messages from multiple group leaders?

-- A new subsection (4.2) has been added.

4)Delete operation: What happens if the old group leader leaves the groups? leader leaving the group is not the same as other member leaving the group, as the group key is heavily influenced by the group leader.

-- When the group leader leaves, any other member can assume the role of a group leader and use the contents of the last GROUP message a new random contribution of its own to generate a totally independent key.

5)Merge operation: How to merge two existing groups?

-- Explained in section on Join/merge

6)As explained earlier, the leader is the central point of failure and is also the most congested due to a flood of message from rest of the members, the protocol should avoid such drawbacks.

-- Possibility to easily change the leader makes the protocol robust to leader failures.

Reviewer #2: Relevance to Ad hoc networks: Most existing group key agreement protocols, (including those with which the efficiency comparison is made) are based on dynamic peer groups. These dynamic groups are modeled as having a fixed topology like a logical ring or tree. A lot of issues are hard to deal with in ad hoc networks unless a certain model is built. For example, Rhee et al.[2] propose a group key management architecture for MANETs overseen by aerial Vehicles and describe a model for group communication and key management. Without such a fixed model, issues like leader election remain fuzzy in an ad hoc networking scenario. In page 3 of the paper, authors propose the leader to be chosen at random from the

set of users. The mechanism of choosing the leader needs more quantification otherwise issues of convergence might arise in an ad hoc network.

-- Section 4.2 added on group leader election

Constant Round Dynamic Group Key Agreement: Dutta et al.[1] have already proposed a constant round dynamic group key agreement protocol in a dynamic scenario. They chose the adversarial model following Bresson et al. (same as the one considered by the authors). Table 1 [1] shows a comparison chart with the Katz and Yung protocol [3] listing communication and computation costs.

-- Infact Dutta et al. [1] is not constant round if all members pariticipate in the protocol (which is the definition of GKA protocols). It has been added in the comparison table.

Security of the Authenticated protocol: The Katz-Yung protocol [3] is a generic protocol which may be used to convert any unauthenticated protocol into an
authenticated one. In Section 4.2, the authors seem to have applied [3] without any modifications. [1] discusses the security of the authenticated (static)
protocol following [3]: no random nonces have been used and the Corrupt oracle queries are avoided making things simpler. Such issues need to be delved into
as an effort to make things simpler (and not compromising on security). After tailoring [3] to the proposed protocol (if possible), a more rigorous proof needs to be constructed than what exists in the paper. Authors are advised to refer to [1] as an example.

-- In the earlier version of the protocol, no changes were needed to the compiler to achieve authentication. But now a new security proof (more efficient than earlier) has been provided without use of the Katz-Yung compiler.

Other issues: The leader has certainly more influence and contribution on the group key than other participating users which is not desirable. The ability of
detecting the presence of the corrupt user (if not detecting the particular user) has not been dealt with. Please refer to [1]. The section concerning the
security model (Section 2) of your paper needs more work in improving the description of the formal adversarial model and definitions.

-- A section on key control is added.

Related Work: The section of related work is incomplete. A significant number of papers dealing with key agreement protocols have not been mentioned in the
paper.

-- All provably-secure protocols have been considered in thge comparison. We do not include some protocols which have no proof of security

*** Submission Checklist**

1) Manuscript - Latex source file - submit.tex
2) Manuscript - Bibtex source file - submit.bib
3) Manuscript - Elsart class file - elsart.cls
4) Figure - Figure 1 - group_laptop_L_cputime_with_JNI_C.eps
5) Figure - Figure 2 - group_laptop_nL_cputime_with_JNI_C.eps
6) Figure - Figure 3 - group_pda_L_cputime_with_JNI_C.eps
7) Figure - Figure 4 - group_pda_nL_cputime_with_JNI_C.eps
8) Author Biography - AUGOT - augot-bio.txt
9) Author Photo - AUGOT - augot.jpg
10)Author Biography - BHASKAR - bhaskar-bio.txt
11)Author Photo - BHASKAR - bhaskar.jpg
12) Author Biography - ISSARNY - issarny-bio.txt
13) Author Biography - SACCHETTI - sacchetti-bio.txt
14) Author Photo - SACCHETTI - sacchetti.jpg
15) list-changes.txt - List of Changes file
16) Submission Checklist - Checklist - checklist.txt