An Efficient Group Key Agreement Protocol for Ad hoc Networks

Daniel Augot, Raghav Bhaskar, Valérie Issarny and Daniele Sacchetti
INRIA Rocquencourt
78153 Le Chesnay
France
{Daniel.Augot, Raghav.Bhaskar, Valérie.Issarny, Daniele.Sacchetti}@inria.fr

Abstract

A Group Key Agreement (GKA) protocol is a mechanism to establish a cryptographic key for a group of participants, based on each one's contribution, over a public network. The key, thus derived, can be used to establish a secure channel between the participants. When the group composition changes (or otherwise), one can employ supplementary GKA protocols to derive a new key. Thus, they are well-suited to the key establishment needs of dynamic peer-to-peer networks as in ad hoc networks. While many of the proposed GKA protocols are too expensive to be employed by the constrained devices often present in ad hoc networks, others lack a formal security analysis. In this paper, we present a simple, secure and efficient GKA protocol well suited to dynamic ad hoc networks. We also present results of our implementation of the protocol in a prototype application.

1. Introduction

Ad hoc networks are networks composed of constrained devices communicating over wireless channels in the (partial) absence of any fixed infrastructure. Moreover, network composition is highly dynamic with devices leaving/joining the network quite frequently. Securing such networks becomes a more difficult task with additional challenges in the form of: lack of trusted third parties, expensive communication, ease of interception of messages and limited computational capabilities of the devices. Key establishment is a vital step in securing in any network. In ad hoc networks, key distribution techniques are not useful as there is not enough trust in the network so as to agree on a key decided by one member or some central authority. Group Key Agreement (GKA) [8] protocols, which enable the participants to agree on a common secret value, based on each participant's public contribution, seem to provide a good solution. They don't require the presence of a central authority. Also, when the group composition changes (as in case of merger or partition of groups), one can employ supplementary key agreement protocols to get a new group key. Thus a transient secure channel can be constructed during the lifetime of one session of a group.

1.1. Related Work

Many GKA protocols [5, 11, 7, 4, 6, 3] have been proposed in literature, most being derived from the two-party Diffie-Hellman (DH) key agreement protocol. While some are secure against passive adversaries only, others do not have a rigorous security proof. A security proof typically involves showing that an attack on a protocol can be used to solve a well-known hard problem under some standard assumptions. Provably secure protocols in a well-defined model of security were first provided by Bresson et al. [4]. Their security model extended the earlier work of Bellare et al. [1]. The number of rounds in these protocols is linear in the number of participants, thus making them unsuitable for large ad hoc networks.

Yung et al. [6] proposed the first provably-secure constant round GKA protocol inspired from the works of Burmester et al. [5]. In the same work, they also proposed a scalable "compiler" to transform a GKA protocol, secure against a passive adversary, into one which is secure against an active adversary. But one round in their protocol consists of 1 broadcast and $n-1$ simultaneous receives by each user. Achieving this is not possible in most networks. Also it lacks procedures to handle group dynamism. Boyd et al. [3] proposed an efficient constant round protocol where the bulk of the computation is done by one participant, thus making it efficient for heterogeneous ad hoc networks. It is provably secure in the Random Oracle model [1] but lacks perfect forward secrecy (i.e., compromise of long-term key compromises all past session[1] keys). We propose a provably secure and efficient protocol which

[1]A session refers to one instance of GKA protocol execution in some group.

Protocol	**Expo per U_i** (Max Expo)	**Rounds** (Messages)	**PS**
[11]	3 (m)	$m+1$ ($2m-3$)	No
[7]	$\log_2 m + 1$	$\log_2 m$ (m)	No
[4]	$i+1$	m (m)	Yes
[6]	3	$2^\star$ ($2m$)	Yes
[9]	2 ($2m^{\star\star}$)	$2^\star$ (m)	Yes
Ours	2 (m)	$2^\star$ (m)	Yes

m: Number of participants
$\star$: 3 rounds for authenticated GKA
$\star\star$: m inverse calculations or $O(m^2)$ multiplications apart from m exponentiations

Table 1. Efficiency Comparison of GKA protocols (PS: Provably Secure)

achieves perfect forward secrecy as well. Subsequent to our work, Won et al. [9] also solve this problem but their proposition turns out to be expensive computationally. In table 1, the number of exponentiations per member for our protocol are compared with some well-known protocols (including maximum number of exponentiations by any member for asymmetric protocols). Also the number of rounds (multiple independent messages can be sent in a round) and total number of messages are provided.

1.2. Outline

The paper is organized as follows: In Section 2, we present a new key agreement protocol for ad hoc environments. It is efficient both in communication and computation terms. Also, most of the exchanged messages are independent of each other, thus making it possible to collect them before the group is defined. In Section 3 we present a security analysis of the same and convert it into an authenticated key agreement protocol. In Section 4, we present our implementation results. Finally, we conclude in Section 5.

2. A New Group Key Agreement Protocol

We propose a new GKA protocol in this section. This protocol is unauthenticated and secure against passive adversaries only. We first introduce the notations used, illustrate the basic principle of key exchange, followed by detailed explanation of how it is employed to derive Initial Key Agreement (IKA), Join/Merge and Delete/Partition procedures for ad hoc groups.

2.1. Notation

G: A subgroup (of prime order q with generator g) of some mathematical group.
M_i: i^{th} participant in the current session.
M_l: The group leader: A member that is elected to coordinate group-level computation such as group-membership and group key management. Can be chosen randomly or by some application specific criteria.
r_i: A random number (from $[1, q-1]$) generated by member M_i for each session. Also called the *secret* for M_i.
g^{r_i}: The *blinded secret* for M_i, which is a public quantity.
$\mathcal{M}$: The set of indices of the participants in the current session (the session being considered).
$\mathcal{J}$: The set of indices of the joining participants (joining the current session).
$\mathcal{D}$: The set of indices of the leaving participants (leaving the current session).
$x \leftarrow y$: x is assigned y.
$x \overset{r}{\leftarrow} \mathcal{S}$: x is randomly drawn from the uniform distribution S.
$M_i \longrightarrow M_j : \{M\}$: M_i sends message M to participant M_j.
$M_i \xrightarrow{B} \mathcal{M} : \{M\}$: M_i broadcasts message M to all participants indexed by $\mathcal{M}$.

2.2. A Two Round Protocol

Protocol Steps:
Round 1: Each M_i responds to the initial request, $INIT$, with its *blinded secret* g^{r_i} to the initiator.
Round 2: The group composition is calculated and the group leader M_l is elected and passed all the received data[2]. M_l raises each joining member's *blinded secret* to its secret (r_l) and broadcasts them along with the original contributions to the group, i.e., it sends $\{g^{r_i}, g^{r_i r_l}\}$ for all $i \in \mathcal{M} \setminus \{l\}$.
Key Calculation: Each M_i checks if its contribution is included correctly and then removes its secret r_i from $g^{r_i r_l}$ to get g^{r_l}. 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.
The protocol is formally defined in table 2. We now see how this protocol can be used to derive IKA, Join/Merge and Delete/Partition procedures for ad hoc networks.

IKA: Secure ad hoc group formation procedures typically involve peer discovery and connectivity checks before a

[2]Note this is part of the group management protocol.

Round 0

$\exists j \in \mathcal{M}, M_j \xrightarrow{B} \mathcal{M} : \{INIT\}$

Round 1

$\forall i \in \mathcal{M} \setminus \{j\}, r_i \overset{r}{\leftarrow} [1, q-1], M_i \longrightarrow M_j : \{g^{r_i}\}$

Round 2

$l \overset{r}{\leftarrow} \mathcal{M}, r_l \overset{r}{\leftarrow} [1, q-1]$

$M_l \xrightarrow{B} \mathcal{M} : \{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$

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

Table 2. IKA

group key is derived. Thus, a discovery request is issued by a member (possibly multiple members) and all interested peers respond. The responses are collected and connectivity checks are carried out to ensure that all members can listen/broadcast to the group (see for instance [2, 10]). 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 0 and Round 1 of the above protocol can take place in the discovery stage as the exchanged messages are independent of each other. In this way, *blinded secrets*, g^{r_i}'s, of all potential members, M_i's, are collected before the group composition is defined. When the fully connected ad hoc group is defined, a single message (Round 2 in table 2) from the group leader, M_l, (using contributions of only the joining members) helps every member to compute the group key. Note that if in the group management protocol, the initiator and the leader are different entities, the leader will be passed on all the *blinded secrets* (along with other management data) during the group management stage. An example is provided below.

Suppose M_1 initiates the group discovery and initially 5 members express interest and send g^{r_2}, g^{r_3}, g^{r_4}, g^{r_5} and g^{r_6} respectively. Finally only 3 join because of connectivity constraints. Suppose the members who finally join are M_2, M_4 and M_5. Then the group leader, say M_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 member 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.

Join/Merge: Join is quite similar to IKA. Each joining member, $M_i(i \in \mathcal{J})$, sends a $JOIN$ request along with its *blinded secret*, g^{r_i} to the existing group. The group leader (M_l) chooses a new random secret, r_l, and sends all the *blinded secrets* to the new group leader[3], $M_{l'}$. The new

[3]For each session, one may want to elect a new leader.

Round 0

$\forall i \in \mathcal{J}, r_i \overset{r}{\leftarrow} [1, q-1]$,

$M_i \xrightarrow{B} \mathcal{M} : \{JOIN, g^{r_i}\}$

Round 1

$r_l \overset{r}{\leftarrow} [1, q-1], \mathcal{M} = \mathcal{M} \cup \mathcal{J}, l' \overset{r}{\leftarrow} \mathcal{M}$

$M_l \longrightarrow M_{l'} : \{g^{r_i}\}_{i \in \mathcal{M} \setminus \{l'\}}$

Round 2

$l \leftarrow l', r_l \overset{r}{\leftarrow} [1, q-1]$

$M_l \xrightarrow{B} \mathcal{M} : \{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$

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

Table 3. Join/Merge

group leader broadcasts a message similar to the round 2 message in IKA, i.e., all the *blinded secrets* and the *blinded secrets* raised to his (new) secret. It is worth noting that when a member, whose *blinded secret* is public, is chosen as the group leader, he chooses a new pair of secret and *blinded secret*. See table 3 for formal specification and below for an example.

Suppose new members, M_9 and M_{10} join the group of M_1, M_2, M_4 and M_5 with their contributions g^{r_9} and $g^{r_{10}}$ respectively. Then the previous group leader (M_1) changes its secret to r_1^* and sends $g^{r_1^*}$, g^{r_2}, g^{r_4}, g^{r_5}, g^{r_9} to M_{10} (say the new group leader). M_{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 $g^{r_{10}^*(1+r_1^*+r_2+r_4+r_5+r_9)}$.

Delete/Partition: Delete is quite similar to Join. When members leave the group, a new group leader is randomly chosen from the remaining members and he changes his secret contribution and sends an IKA Round 2 like message to the group, omitting the leaving members' contributions. We omit the details.

3. Security Result

The protocol presented in the earlier section is provably secure against passive adversaries in the model of [4], from where the notations and definitions are taken.

Theorem 1: Let P be the protocol as defined above. Let $\mathcal{A}$ be a passive adversary making $q_{ex} = (q_{ika} + q_{join} + q_{delete})$ **Execute** queries to the parties and running in time t. Then Protocol P is a secure GKA protocol. Namely:

$$Adv_P^{\mathcal{A}}(t, q_{ex}) \leq 2q_{ex} * Succ^{DDH}(t')$$

where $t' \leq t + q_{ex}|\mathcal{P}|t_{exp}$, t_{exp} is the time to perform an exponentiation in G and $|\mathcal{P}|$ being the maximum number of participants in the protocol.

Proof: Due to lack of space, we only give a sketch of the proof. The complete proof will appear in the full version[4]. We show that an adversary who achieves an advantage in calculating the session key, can be used to build an attacker Δ which gains an advantage in solving an instance of the Decisional Diffie-Hellman (DDH) Problem. The **Send** and **Corrupt** queries are not applicable as we are dealing with a passive adversary and there are no long-term secrets. Thus the only relevant queries are the **Execute**, **Reveal** and **Test** queries. Assume the adversary $\mathcal{A}$ distinguishes the session key with a probability non-negligibly greater than 0.5. We construct from $\mathcal{A}$ a DDH attacker Δ that receives as input an instance $D = \{g, g_1, g_2, g_3\}$ and predicts if it is an instance from $(g, g^{r_a}, g^{r_b}, g^{r_a r_b})$ or $(g, g^{r_a}, g^{r_b}, g^{r_c})$ with a non-negligible advantage.

The Attacker Δ feeds $\mathcal{A}$ with elements derived from the instance D in the reply to the **Execute** query of the session for which $\mathcal{A}$ will make the **Test** query. So Δ picks at random c_{test} from $[1, q_{ex}]$ which is its guess for the number of the **Execute** query, corresponding to the session, for which $\mathcal{A}$ makes the **Test** query. For all other sessions, Δ responds to **Execute** queries with randomly generated data.

Δ replies to the **Test** query with a session key, sk, constructed using data from the instance D. sk is a valid session key only if the instance D is a DH tuple. Thus, if the adversary $\mathcal{A}$ correctly identifies sk as the session key, the tuple (g, g_1, g_2, g_3) is indeed a DH tuple otherwise it is a random tuple. The success probability of Δ is the probability that it correctly guesses the session for which $\mathcal{A}$ makes the **Test** query ($1/q_{ex}$), multiplied by the success probability of $\mathcal{A}$. Thus if we denote by p the probability of adversary $\mathcal{A}$ distinguishing the session key, the probability of success of Δ is: $Succ^{DDH}(t') \geq p/q_{ex}$.

The running time of Δ is bounded by the running time of $\mathcal{A}$ and the time to perform at most $|\mathcal{P}|$ exponentiations during q_{ex} queries.

3.1. An authentication compiler

In [6], Yung et al. introduced a scalable compiler which transforms any GKA protocol P, secure against passive adversary, to an authenticated GKA protocol P', secure against an active adversary. It achieves this by enhancing the protocol to include a (pre-)round where everyone broadcasts its identity and a random nonce. Thereafter each message is accompanied by a signature on the message, identities of the participants and their nonces (see [6] for details). Then if P is a secure GKA protocol, then the protocol P' is a secure Authenticated GKA protocol. Namely,

Theorem 2: $Adv_{P'}^{\mathcal{A}'}(t,\ q_{ex},\ q_s) \leq \frac{q_s}{2} * Adv_P^{\mathcal{A}}(t', 1) + Adv_P^{\mathcal{A}}(t', q_{ex}) + |\mathcal{P}| * Succ_\Sigma(t') + \frac{q_s^2 + q_{ex} q_s}{2^k}$

[4]http://www.inria.fr/rrrt/index.en.html

where:
q_{ex} and q_s are the number of **Execute** and **Send** queries respectively.
$t' = t + (|\mathcal{P}|q_{ex} + q_s).t_{P'}$, $t_{P'}$ is the time to execute P'.
$Adv_{P'}^{\mathcal{A}'}(t, q_{ex}, q_s)$: Advantage of an active adversary ($\mathcal{A}'$) against the authenticated protocol P', making q_{ex} **Execute** queries and q_s **Send** queries in time t.
$Adv_P^{\mathcal{A}}(t', 1)$: Advantage of a passive adversary ($\mathcal{A}$) against the protocol P, making 1 **Execute** query in time t'.
$Adv_P^{\mathcal{A}}(t, q_{ex})$: Advantage of a passive adversary ($\mathcal{A}$) against the protocol P, making q_{ex} **Execute** queries in time t'.
$Succ^{DDH}(t')$: Success probability of an adversary against an instance of the DDH problem in time t'.
$Succ_\Sigma(t')$: Success probability of an adversary against the signature scheme Σ in time t'.
and k is the security parameter.

3.2. Authenticated protocol

Thus applying the above compiler to our protocol yields a 3-round authenticated GKA protocol, P' with the following security reduction:

Theorem 3: $Adv_{P'}^{\mathcal{A}'}(t,\ q_{ex},\ q_s) \leq (q_s + 2q_{ex}) * Succ^{DDH}(t') + |\mathcal{P}| * Succ_\Sigma(t') + \frac{q_s^2 + q_{ex} q_s}{2^k}$

4. Implementation

To test the performance of this new GKA protocol, we incorporated it in the group management protocol of [2]. The group management of [2] 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 4. 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 cho-

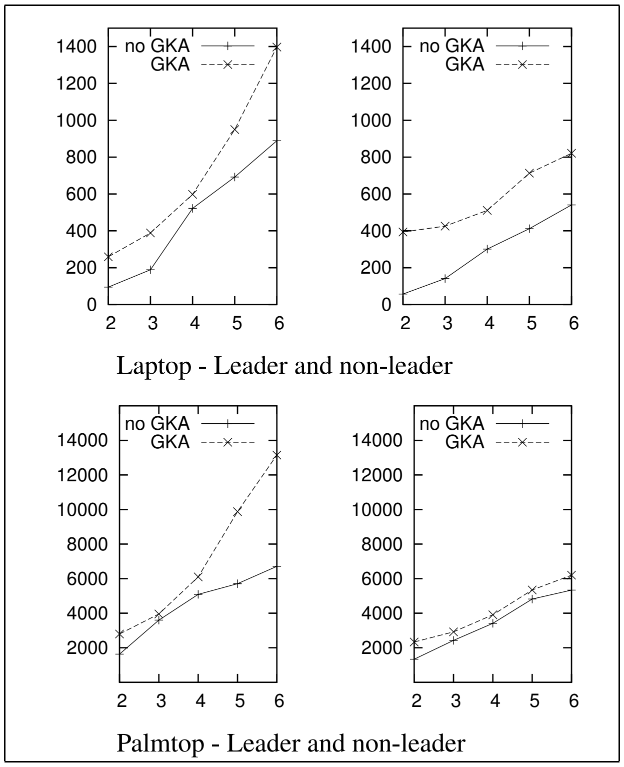

Table 4. Computation time per device with and without GKA for laptop and palmtop

sen 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 [5]. The graphs in table 4 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.

5. Conclusion

We have proposed a new group key agreement protocol, particularly well suited to ad hoc networks, and secure against a passive adversary. It is efficient in the number of rounds (only two rounds, the first round may be executed along with group management procedures), and also efficient in computational terms. It can be, using Yung et al. compiler, transformed into a three round protocol secure against an active adversary. This adds to the cost of the protocol, by adding one round of broadcasts.
The protocol is simple and we have provided a security proof in the framework of [4], using the standard model and the Decisional Diffie-Hellman assumption in any group. Experimental results show that our protocol results in a reasonable computational overhead during group formation with hardly any communication burden. Further reductions in terms of computation overhead can be made by using Elliptic curve groups.

[5]http://www.swox.com/gmp/

References

[1] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *ACM Conference on Computer and Communications Security*, 1993.
[2] M. Boulkenafed, D. Sacchetti, and V. Issarny. Using group management to tame mobile ad hoc networks. *IFIP TC8 Working Conference on Mobile Information Systems*, 2004.
[3] C. Boyd and J. Nieto. Round-optimal contributory conference key agreement. *6th International Workshop on Practice and Theory in Public Key Cryptography*, 2003.
[4] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie Hellman key exchange under standard assumptions. *In Advances in Cryptology - EUROCRYPT*, 2002.
[5] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. *Advances in Cryptology - EUROCRYPT*, 1994.
[6] J. Katz and M. Yung. Scalable protocols for authenticated key exchange - full version. *Advances in Cryptology - CRYPTO*, 2003.
[7] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. *ACM CCS*, 2000.
[8] A. J. Menezes, P. C. van Oorschot, and S. Vanstone. *HandBook of Applied Cryptography*. CRC Press, 1996.
[9] J. Nam, J. Lee, S. Kim, and D. Won. DDH based group key agreement for mobile computing. *http://eprint.iacr .org/2004/127*, 2004.
[10] G.-C. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. *International Conference on Software Engineering*, 2001.
[11] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 2000.