AGDH (Asymmetric Group Diffie Hellman) An Efficient and Dynamic Group Key Agreement Protocol for Ad hoc Networks

Raghav Bhaskar[1], Daniel Augot[2], Cédric Adjih[2], Paul Mühlethaler[2], Saadi Boudjit[3]

[1] Microsoft Research India.
[2] INRIA, BP 105, Rocquencourt, 78153 Le Chesnay Cedex, France.
[3] GET/ENST, 37/39, rue Dareau 75014 PARIS, FRANCE.

Abstract. Confidentiality, integrity and authentication are more relevant issues in Ad hoc networks than in wired fixed networks. One way to address these issues is the use of symmetric key cryptography, relying on a secret key shared by all members of the network. But establishing and maintaining such a key (also called the session key) is a non-trivial problem. We show that Group Key Agreement (GKA) protocols are suitable for establishing and maintaining such a session key in these dynamic networks. We take an existing GKA protocol, which is robust to connectivity losses, and discuss all the issues for the correct functioning of this protocol in Ad hoc networks. We give implementation details and network parameters, which significantly reduce the computational burden of using public key cryptography in such networks.

1 Introduction

A Mobile Ad hoc NETwork (MANET) is a collection of mobile nodes connected via a wireless medium forming an arbitrary topology. Implicit herein is the ability for the network topology to change over time as links in the network appear and disappear. To maintain network connectivity, a routing protocol must be used. An important security issue is that of the integrity of the network itself. A considerable number of studies have already been carried out to resolve security issues in existing routing protocols (see [14],[29],[3],[1]).

An orthogonal security issue is that of maintaining confidentiality and integrity of data exchanged between nodes in the network. The task of ensuring end-to-end security of data communications in MANETs is equivalent to that of securing end-to-end security in traditional wired networks. Many studies have been carried out to solve this problem. One widespread solution is to create a virtual private network (VPN) in a tunnel between the two communicating nodes. IPSec is a well known security architecture which allows such VPNs to be built between two communicating nodes. However this solution requires a different secret key for each end-to-end connection. Moreover the VPN solution can simply handle unicast traffic. An alternative solution is the use of a shared secret key. Such an approach raises many issues. First this key must be distributed among the network nodes. Second, to avoid the compromising of this key it is necessary to renew the key often. One solution to these two issues is the use a Group Key Agreement protocol, which relies on the principles of public key cryptography.

A Group Key Agreement protocol (GKA) is a key establishment technique in which a shared secret is derived by more than two participants as a function of information publicly contributed by each of them. They are especially well suited to moderate sized groups with no central authority to distribute keys. An authenticated group key agreement protocol provides the property of key authentication (also called implicit key authentication), whereby each participant is assured that no other party besides the participants can gain access to the computed key. GKA protocols are different from group key distribution (or key transport) protocols wherein one participant chooses the group key and communicates it to all the others. GKA protocols help in deriving keys which are composed of each one's contribution. This ensures that the resulting key is fresh (for a given session) and does not favor one participant in any way. The following security goals can be identified for any GKA protocol.

1) **Key Secrecy**: The key can be computed only by the 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 [6]).
3) **Forward Secrecy**: Knowledge of some long-term secret does not lead to the knowledge of past group keys.

An important advantage of a group key agreement protocol over a simple group key distribution scheme is the forward secrecy. This property can be particularly interesting in situations where some nodes are likely to be compromised (e.g. in military scenarios). In such scenarios, using a GKA, the knowledge of the long-term secret of this node does not compromise all past session keys. From a functional point of view, it is desirable to have procedures to handle the dynamism in the network. These procedures enable efficient merging or partitioning of two groups in the network.

2 Related Work

Key establishment protocols for networks can be broadly classified into three classes: *Key transport using symmetric cryptography*, *Key transport using asymmetric cryptography* and *Key agreement using asymmetric cryptography*. In key transport protocols, one participant chooses the group key and securely transfers it to the other participants using a priori shared secrets (symmetric or asymmetric). These protocols are not suitable for ad hoc networks for two reasons: firstly, they require a single trusted authority to distribute keys and secondly, if the a priori secret of any participant is compromised, this breaches the security of all past group keys, thus failing to provide forward secrecy. Thus GKA protocols are more relevant since they provide this forward secrecy property.

Most group key agreement protocols are derived from the two-party Diffie-Hellman key exchange protocol. GKA protocols, not based on Diffie-Hellman, are few and include the protocols of Pieprzyk and Li [28], Tzeng and Tzeng [33] and Boyd and Nieto [7]. Both the protocols of Pieprzyk and Li [28] and Boyd and Nieto [7] fail to provide *forward secrecy* while the protocol of Tzeng and Tzeng [33] is quite resource-intensive and prone to certain attacks [7]. Forward Secrecy is a very desirable property for key establishment protocols in ad hoc networks, as some nodes can be easily compromised due to low physical security of the nodes. Thus it is essential that if one single node is compromised, this does not compromise all past session keys. In Table 1 we summarize and compare in Table 1 existing GKA protocols based on Diffie-Hellman protocols. We compare essentially the unauthenticated versions of the protocols, as most achieve authentication by using digital signatures in a very similar manner and thus have similar added costs to achieve authentication. We compare the efficiency of these protocols based on the following parameters:

- **Number of synchronous rounds**: In a single synchronous round, multiple independent messages can be sent in the network. The total time required to run a round-efficient GKA protocol can be much less than other GKA protocols that have the same number of total messages but more rounds; because the nodes spend less time waiting for other messages before sending their own.
- **Number of messages:** This is the total number of messages (unicast or broadcast) exchanged in the network to derive the group key. For multiple hop ad hoc networks, the distinction between unicast and broadcast messages is important as the latter can be much more energy consuming (for the whole network) than the former.
- **Number of exponentiations**: All Diffie-Hellman based GKA protocols require a number of modular exponentiations to be performed by each participant. Relative to all cryptographic operations, a modular operation is the most computationally intensive operation and thus gives a good indication of the computational cost for each node.

Communication costs still remain the critical factor for choosing energy-efficient protocols for most ad hoc networks. A modular exponentiation (which is most efficiently done using elliptic curve cryptography) can be performed in a few tens of milliseconds on most palmtops, whereas message propagation in multi-hop ad hoc networks can be easily of the order of a few seconds and has energy implications for multiple nodes in the network. As can be seen in Table 1, most existing GKA protocols require $O(m)$ rounds of communication for m participants in the protocol. Such protocols do not scale well in ad hoc networks. Even tree-based

GKA protocols with $O(\log m)$ rounds can be quite demanding for medium to large sized ad hoc networks. Therefore constant-round protocols are more suitable for ad hoc networks.

	Expo per U_i	Messages	Broadcasts	Rounds
ITW [15]	m	$m(m-1)$	0	$m-1$
GDH.1 [32]	$i+1$	$2(m-1)$	0	$2(m-1)$
GDH.2 [32, 10]	$i+1$	$m-1$	1	m
GDH.3 [32]	3	$2m-3$	2	$m+1$
TGDH [20]	$\leq \log_2 m$	$2m-2$	$2m-2$	$\log_2 m$
Perrig [27]	$\log_2 m+1$	m	$m-2$	$\log_2 m$
Dutta [13]	$\log_3 m$	m	m	$\log_3 m$

Table 1. Comparison of non constant rounds GKA protocols

	Expo per U_i	Mes- sages	Broad- casts	Rounds	Structure	FS
Octopus [5]	4	$3m-4$	0	4	Hypercube	Yes
BDB [12, 17]	3	$2m$	m	2	Ring	Yes
BCEP [9]	$2^{\dagger}$	$2m$	0	2	None	No
Catalano [8]	$m+1$	$2m$	0	2	None	Yes
KLL [18]	3	$2m$	$2m$	2	Ring	Yes
NKYW [24]	$2^{\ddagger}$	m	1	2	None	Yes
STR [31, 19, 23]	$(m-i)^{*}$	m	1	2	Skewed tree	Yes
TFAN [21]	$0(m)$	m	$O(m)$	3 or 4	Tree	Yes
Ours (AGDH)	2^{**}	m	1	2	None	Yes

$\dagger$: m exponentiations for the base station.
$\ddagger$: $m+1$ exponentiations and m-1 inverse calculations for the parent node.
$*$: Up to $2m$ exponentiations for the sponsor node.
$**$: m exponentiations for the leader.

Table 2. Comparison of constant round GKA protocols

Among the constant round protocols (see Table 2), Octopus [5], BDB [17] and KLL [18] require special ordering of the participants. This results in messages sent by some participant being dependent on that of others. In such a case, failure of a single node can often halt the protocol. Thus such protocols are not robust enough to adapt well to the dynamism of ad hoc networks. The BCEP protocol [9] involves a base station, and fails to provide forward secrecy if the long-term secret of the base station is revealed. The Bresson and Catalano protocol [8] is computationally demanding with $O(m)$ exponentiations for each participant. Another drawback is that if any participant's message is lost in first round, the whole protocol is brought to a halt, as the secret sharing schemes implies that all m contributions are required to compute the key.

NKYW[24]: The original paper proposes this protocol for ad hoc networks composed of devices with unequal computational powers.

In the first round, each participant M_i unicasts its contribution $g^{r_i}, i \in [1, n-1]$ to a fixed node M_n, called the parent node. The parent node chooses random r and r_n and computes $w = g^r$, $x_n = g^{rr_n}$ and $x_i = (g^{r_i})^r$ for each received g^{r_i}. It broadcasts w and $\{x_n * \mathit{\Pi}_{j \neq i} x_j\}_i$. The key is derived from $\mathit{\Pi}_i x_i$. The protocol remains somewhat expensive computationally compared to the protocol that will be described in this paper.

STR[31, 19]: This protocol was proposed by Steer *et al.* in [31] for static groups. Perrig *et al.* proposed procedures to handle group changes in [19]. In [23] a suite of protoocls called μSTR are proposed to optimize the STR protocol for MANETs.

TFAN [21] is a merge of μSTR and μTGDH, which are optimizations of STR and TGDH, also proposed in [21]. TFAN provides a trade-off between computational and communication costs.

Among all the reviewed GKA protocols reviewed only a few of them [23, 22, 21] are designed for MANETs. The protocol proposed in this paper is more robust to messages losses than the previous one. If a contribution of a given member is lost then this member can not compute the key but the others can still agree on shared secret key.

The contributions of this paper are the following:

- an authenticated dynamic group key agreement protocol is recalled [4],
- the mechanisms that must be used in a MANET to implement this group key agreement protocol are described,
- a precise study of the cryptographic parameters that this group key agreement protocol must use in the context of an ad hoc network is carried out.

Finally the adapted version of the group key agreement protocol that we propose, we call this protocol AGDH for Asymmetric Group Diffie Hellman, is among the very few group key agreement protocols suitable for ad hoc networks. Note that, in this paper, we do not consider malicious insiders and also the unrelated issue of selfishness.

The paper is organized as follows:

- Section 3 recalls the group key agreement protocol. We only describe the basic functioning of the protocol,
- Section 4 explains how this group key agreement protocol can be implemented in an ad hoc network. The main issues discussed in this section include the election of a leader in the ad hoc network and the actions that must be undertaken to handle splits and mergers in the ad hoc network,
- Section 5 discusses the overhead of cryptographic operations.

3 Presentation of AGDH

We recall an existing 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 to handle dynamism in 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 ($l \in \{1, \ldots, n\}$).
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.
$x \overset{r}{\leftarrow} \mathcal{S}$: 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 generated by participant U_i.
$\mathcal{V}_{PK_i}\{msg_i, \sigma_i\}$: Signature verification algorithm which returns 1 if σ_i is a valid signature on message msg_i and else returns 0.

3.2 A Three Round Protocol

The formal description 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 3- 5) and is verified (along with the nonces) by the receiver before following the protocol. Thus we do not to describe these steps which are formally shown in Tables 3- 5.

Protocol Steps:

Round 1: The chosen group leader, M_l makes an initial request (**INIT**) with its 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 a **IREPLY** message which contains its identity U_i, a nonce N_l and a blinded secret g^{r_i} to M_l (see Table 3 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 an **IGROUP** message which contains $\{U_i, N_i, g^{r_i}, g^{r_i r_l}\}$ 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 the case of group modifications (see below), and also because it may be possible that a particular contribution has not been received by some member.

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 can not be pre-determined and is unique to this session.

The protocol is formally defined in Table 3. Table 4 (respectively Table 5) show how the protocol is run when a group wants to join (respectively leave) an existing group

Example runs of the protocol We now see how this protocol can be used to derive Initial Key Agreement (IKA), Join/Merge and Delete/Partition procedures for ad hoc networks.

$\boxed{\textbf{Round 1}}$
$l \overset{r}{\leftarrow} \mathcal{M}, N_l \overset{r}{\leftarrow} \{0,1\}^k$
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^1 = \{\ \textbf{INIT}\ , U_l, N_l\}, \sigma_l^1\}$
$\boxed{\textbf{Round 2}}$
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_l}\{msg_l^1, \sigma_l\} == 1),$
$r_i \overset{r}{\leftarrow} [1, q-1],\ N_i \overset{r}{\leftarrow} \{0,1\}^k,$
$U_i \longrightarrow U_l : \{msg_i = \{\ \textbf{IREPLY}, U_l, N_l, U_i, N_i, g^{r_i}\}, \sigma_i\}$
$\boxed{\textbf{Round 3}}$
$r_l \overset{r}{\leftarrow} [1, q-1],\ \forall i \in \mathcal{M} \setminus \{l\},$
$if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)$ and N_l is as contributed
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^2 =$
$\{\textbf{IGROUP}, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$
$\boxed{\textbf{Key Computation}}$
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l (1 + \sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 3. IKA

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 a 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 [30]). After the group membership has been 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 has been defined, a single broadcast message (Round 3 in Table 3) from the group leader, U_l, (using contributions of only the joining participants) helps every participant to compute the group key. An example is provided below.

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 its respective secret. Thus the key $g^{r_1(1+r_2+r_4+r_5)}$ can be computed.

$\boxed{\textbf{Round } 1}$
$\forall i \in \mathcal{J}, r_i \overset{r}{\leftarrow} [1, q-1], N_i \overset{r}{\leftarrow} \{0,1\}^k,$
$U_i \xrightarrow{B} \mathcal{M} : \{msg_i = \{\ \textbf{JOIN}, U_i, N_i, g^{r_i}\}, \sigma_i\}$
$\boxed{\textbf{Round } 2}$
$\forall i \in \mathcal{J}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)\ r_l \overset{r}{\leftarrow} [1, q-1], l' \overset{r}{\leftarrow} \mathcal{M} \cup \mathcal{J}$
$U_l \longrightarrow U_{l'} : \{msg_l = \{\ \textbf{JREPLY}, \{U_i, N_i, g^{r_i}\}_{\forall i \in \mathcal{M} \cup \mathcal{J}}\}, \sigma_l\}$
$\boxed{\textbf{Round } 3}$
$if(\mathcal{V}_{PK_i}\{msg_l, \sigma_l\} == 1), l \leftarrow l', r_l \overset{r}{\leftarrow} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{J}$
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^2 =$
$\quad \{\ \textbf{JGROUP}, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$
$\boxed{\textbf{Key Computation}}$
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 4. Join/Merge

Join/Merge 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 $g^{r_{10}^{'}(1+r_1^{'}+r_2+r_4+r_5+r_9)}$.

Delete/Partition When participants leave the group, they send a **DEL** message, the group leader changes its secret contribution and sends an **IKA** Round 3 like message to the group, omitting the leaving participants' contributions. Refer to Table 5 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 $g^{r_{10}^{''}(1+r_1^{'}+r_4+r_5+r_9)}$.

Security proof A full security proof has been given in [4] either for the case of the passive adversary or for the case of the active adversary in the widely used security model of [11]. This proof ensures the secrecy of the key under the assumption of the hardness of the Decisional Diffie-Hellman problem.

In the next Section we discuss how to adapt this protocol when message losses occur. Note however that the security proof of [4] does not provide a security proof of the adapted protocol.

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

Table 5. Delete/Partition

4 Using this GKA protocol within an ad hoc network

In the following we consider a multi-hop ad hoc network. We do not assume any particular property of the routing protocol which ensures the connectivity of the network. We can use reactive protocols such as AODV or DSR [26, 16] where the connectivity is created on demand when a route is needed. We can also use proactive protocols as such as OLSR or TBRPF [2, 25] where synchronous packets are used to maintain the knowledge of the topology. We will assume that we have a broadcast mechanism to flood messages within the ad hoc network. We do not assume that this flooding mechanism is reliable, but we do assume that the network is connected and that flooding messages finally reach all the network nodes [4].

A key point in the GKA protocol described above is the existence of a group leader. Thus it is necessary to have a robust mechanism to elect such a leader in an ad hoc network. This is the first issue that we study.

4.1 Election of a group leader

A key requirement is that all members of a group agree on the same group leader. A simple solution is that the group leader periodically broadcasts messages. These messages then serve as a proof, for nodes that are within reach of the group leader, that a group leader exists and is operating properly. We can simply use the **INIT** message of GKA protocol to demonstrate the existence and the correct functioning of the group leader. When the other nodes in the network receive this **INIT** message, each replies with an **IREPLY** message including their contribution. Using these **IREPLY** messages, the group leader defines a group and sends an **IGROUP** message to all members of the group. The **INIT** message can be seen as an **IGROUP** message when the group has not yet been defined. In the following we will only use the term **IGROUP** message.

These **IGROUP** messages are sent periodically; depending on the dynamics of the group, the group leader will send a new **IGROUP** message or exactly the same message as before. If the network only comprises of the group leader, the latter will send periodically empty **IGROUP** messages. It will stop sending this message when a node joins its network by replying to its **IGROUP** message with an **IREPLY** message. The mechanism to elect a group leader simply follows from the property that, in a network with a group leader, periodic messages are broadcast by the group leader and are, in principle, received by the group members. If a node does not receive a message for a fixed period T, known a priori by the network nodes, this node sets a random timer. At the expiration of this timer and if no **IGROUP** message has been received meanwhile, the node becomes the group leader. It then sends an empty **IGROUP** message.

[4] We mean that synchronous flooded messages will finally reach all the network nodes even if there are message losses

There may be a collision on **IGROUP** messages if two or more nodes or more have selected the same value for their random timer. In such a case, there may be **IGROUP** messages generated by two (or more) group leaders. To select a group leader, we can use additional rules. The first rule is that when a group leader A receives an **IGROUP** message from a group leader B which has a smaller ID than its own ID, group leader A just stops sending its periodic messages. The group members that receive periodic messages from more than two group leaders will only consider the message issued by the group leader with the smallest index. Thus if an **IGROUP** message showing a larger ID than a previously received **IGROUP** message is received, then this message is simply discarded and no **IREPLY** message is issued. On the other hand, if an **IGROUP** message showing a smaller ID is received then the node issues a **IREPLY** message.

Another issue is how the GKA protocol takes into account the dynamism of an ad hoc network. For instance a node may leave the network without being able to send the group leader a message pointing out its departure from the network. This issue is handled in the next subsection

4.2 Handling the joining and withdrawal of a node

A node which joins the network will receive the periodic **IGROUP** message of the group leader. He will just have to send **JREPLY** message, with its contribution, to join the group. The group leader will incorporate this new contribution in its next **IGROUP** message. Actually there is no need in the protocol to differentiate between **JREPLY** and **IREPLY**. Thus, for simplicity sake, we will only keep the **IREPLY** message.

In an ad hoc network, the only conceivable way for the group leader to be sure that a node remains in a group is to receive a message from it. Thus to handle the dynamism of a group, the group leader will use the periodic reception of the **IREPLY** messages. The period with which an **IREPLY** message is sent by a member of the group should be the same for all the nodes of the group. If the group leader is not receiving a **IREPLY** message for a given number of periods (greater than 1 to handle possible packet loss), the lack of reception of these messages should be handled in the same way as the reception of a **DEL** message. In such a case the group leader will change its own contribution in the **IGROUP** message and will re-send the **IGROUP** message.

When a node deliberately wishes to withdraw from a group it can use the **DEL** message to announce this wish to the group leader. Upon reception of such a message the group leader will change its own contribution in the **IGROUP** message and will re-send the **IGROUP** message. The use of the **DEL** message will speed up the taking into account of the node withdrawal.

4.3 Handling the merging or spliting of groups

The merger of (two or more) groups leads group leaders to receive **IGROUP** messages from other group leaders. The scheme used in the group leader election can be used to resolve the conflict. When the conflict is resolved only one group leader remains in the group. If a group splits, a part of the group will remain without a group leader. The technique used in the group leader election can be used in the subgroups without a leader to elect a new leader.

4.4 Renewing its contribution

The group leader and group members will have to renew their contribution periodically. For the group leader, the change of its contribution or of some member of the group will lead to a change in the content of the **IGROUP** message. To simplify we can assume that the group leader and the group members change their contribution at the same rate.

We have given all the principles of the protocol. We specify the details of the whole protocol in the next section.

4.5 Implementation issues

We will consider a given period T. To simplify, this period will be used both by the group leader and by the member of the group as a period to send their GKA messages.

A node can be in one of the following two states : **member state** or **group leader state**. A node in a member state will enter the process to become a group leader if it has not received an **IGROUP** message for a duration kT. A node which has not received any message from a group leader for a duration kT with $k \geq 2$ will suppose that there is no group leader and starts the procedure to become a leader. Since a node may not have received a packet of the group leader because this packet has been lost, k must be selected so that the probability that $k-1$ successive transmissions of a GKA message are lost is small. Then, to become a group leader, the node selects a random integer i_r between 1 and a given number l (backoff window size) and initializes a timer at $i_r t_{rtd}$, where t_{rtd} is a predefined duration computed to be at least the round trip delay of a message throughout the ad hoc network. With such a figure for t_{rtd} we can be sure that if two nodes draw different integers i_r and $i_{r'}$, the node having selected the larger integer will receive the **IGROUP** message of the other node and then will stop its election process. The backoff window size l must be chosen with respect to the total number of nodes in the network so that the probability of two nodes choosing the same integer is small. This back-off procedure is performed to avoid possibly multiple group leader candidates, for instance, when a group is set up or split into two subgroups.

When the node in the member state sends its first **IGROUP** message, it is in the group leader state.In the group leader state, a node must collect **IREPLY** messages and form the related **IGROUP** message. When there is a change in the group (arrival or withdrawal) the group leader must change its contribution. Additionally, irrespective of the modification of the composition of the group, the group leader must change its contribution periodically, to maintain the security of the session key.

When a group leader is elected, it may choose to wait additional periods before sending an **IGROUP** containing the contributions of the group members. In doing so, the group leader may avoid unnecessary changes to the session key due to not having received all the contributions in time.

In the group leader state, a node will also look out for **IGROUP** messages from another group leader. If it receives such a message from another group leader holding a smaller node index, the node changes its state to the member state. In the member state, a node will have to send **IREPLY** messages periodically. Like the group leader, a group member must change its contribution periodically with a period P see figure 1. We will assume that P is a large multiple of T. To simplify the procedure and to avoid unnecessary computations we can assume that the group leader does not instantly include a new contribution of a group member in the **IGROUP** message, but rather it will wait for the change of its own contribution to take into account all new contributions of the nodes. This is possible since the contribution of the node member is included in the **IGROUP** message.

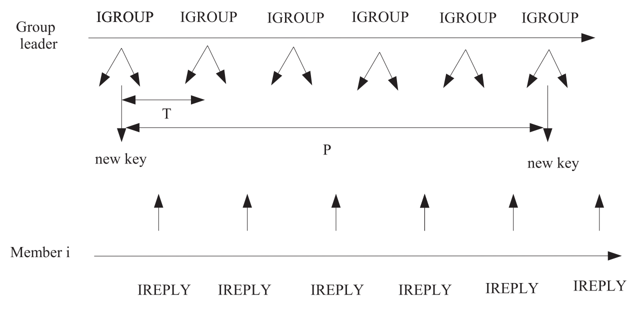

Fig. 1. Sending **IGROUP** and **IREPLY** messages

Both **IGROUP** and **IREPLY** messages must be sent periodically for each interval T. In order to reduce the probability of collision of these messages, we add a jitter to the times when the GKA messages will be sent by the group members and the group leader.

In table 6, we give examples of figures for our GKA protocol. We can notice that l and t_{rtd} will greatly depend on the number of nodes in the network and on the topology of the network.

Parameter	Value	Constraint
P: key renew period	20 min	
T: period of IGROUP messages	5s	
k: number of message losses before assuming a node has left	3	large enough to be sure that the message is not simply lost
l: backoff window	20	large enough to avoid collision during the group leader election
t_{rtd}: backoff slot for leader election	100 ms	more than a round trip delay

Table 6. Protocol parameters

5 Computational overhead

Group	Size of contrib.	blindings/second =recoveries/second
Modular Field	1024 bits	10
Elliptic curve	160 bits	93

Fig. 2. Performance of elliptic curve cryptography, versus a classical group (modular integers)on a iPAQ, StrongARM-1110, using the `openssl` implementation, for a security level of 2^{80}. Blinding means computing g^{r_i}, and recovering means computing g^{r_0} from the blinded response $g^{r_i r_0}$ of the leader .

Figure 2 describes the cost, on an average small device (COMPAQ iPAQ) of elliptic curve cryptography, which is more efficient than classical cryptographic relying on bigger groups. Basically, for a security level of 2^{80}, such a device can perform almost 100 operations per second. Thus the latency of elliptic curve exponentiation is 10 msec per device, except for the leader whose computational cost grows linearly with the size of the group. Thus there is concern for this particular node. Assuming that the leader devotes half its time to cryptographic operations, managing a group of size 50 will impose a delay of 1 second before being able to send the blinded response.

The above computational load on the group leader is in the case where the group leader receives all the blinded secrets at once, and has to give the blinded response also at once. In practice, the group leader will receive the blinded secret at different time slots. It is then possible to perform operations in batch: the group leader can generate its own secret in advance, and compute on the fly the blinded responses $(g^{r_i})^{r_0}$ upon reception of each blinded secret g^{r_i}. It can also compute stepwise the product $(g^{r_1})^{r_0} \cdots (g^{r_m})^{r_0}$, where m is the index of the last received contribution. When it has to broadcast the `IGROUP` message, all the computationaly intense cryptographical operations, necessary to generate the blinded responses, have already been performed.

6 Conclusion

We have discussed a group key agreement protocol for handling ad hoc groups of small to moderate size. We have fully specified the implementation details needed to use of the protocol, relying on known network techniques such as self election, periodic broadcast and back-off techniques. The protocol is robust in the sense that connectivity losses does not impair its functioning. Experiments have shown that the computational cost of public key cryptography is kept reasonably low. If we consider constraints in ad hoc networks: no network structure, high dynamism, restricted bandwidth, etc the presented protocol is among the few GKA protocols which are suitable for ad hoc networks.

To give a proof of security in presence of message losses seems to be a non trivial task. For instance the security model of [11] does not apply easily for this case. Work is ongoing to define a security model coping with message losses and then to prove the security of the AGDH protocol.

References

1. C. Adjih, T. Clausen, Anis Laouiti, Paul Muhlethaler, and Daniele Raffo. Securing the OLSR routing protocol with or without compromised nodes. *Rapport INRIA*, RR-5494:55, February 2005. http://www.inria.fr/rrrt/rr-5494.html.
2. Cédric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized link-state routing protocol. RFC 3626, October 2003.
3. Cedric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Muhlethaler, and Daniele Raffo. Securing the OLSR protocol. In *Proceedings of the 2nd IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2003)*, Mahdia, Tunisia, June 25–27 2003.
4. Daniel Augot, Raghav Bhaskar, Valerie Issarny, and Daniele Sacchetti. A three round authenticated group key agreement protocol for AdHoc networks. *Pervasive and Mobile Computing*, 3(1):36–52, 2007.
5. K. Becker and U. Wille. Communication complexity of group key distribution. In *Proceedings of 5th ACM Conference on Computer and Communications Security*, pages 1–6. ACM Press, 1998.
6. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2003.
7. C. Boyd and J.M.G. Nieto. Round-optimal contributory conference key agreement. In *Public Key Cryptography '03*, pages 161–174. LNCS 2567, 2003.
8. 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.
9. E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval. Mutual authentication and group key agreement for low-power mobile devices. In *Proceedings of the 5th IFIP-TC6 International Conference on Mobile and Wireless Communication Networks*, pages 59–62. World Scientific Publishing, 2003.
10. 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.
11. 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.
12. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology - EUROCRYPT*, volume 839, pages 275–286. LNCS, 1994.
13. R. Dutta and R. Barua. Dynamic group key agreement in tree-based setting. In *ACISP*, pages 101–112, 2005.
14. Y. Hu, A. Perrig, and D. Johnson. Ariadne:: A secure on-demand routing protocol for ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international Conference on Mobile Computing and Networking*, pages 12–23. ACM Press, 2002.
15. I. Ingemarsson, D. T. Tang, and C.K. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, 28(5):714–720, 1982.
16. David B. Johnson, David A. Maltz, and Yih-Chun Hu. The Dynamic Source Routing protocol for mobile ad hoc networks (DSR), July 19 2004. Internet-Draft, `draft-ietf-manet-dsr-10.txt`, work in progress.
17. 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.
18. H-J. Kim, Lee S-M, and D.H. Lee. Constant-round authenticated group key exchange for dynamic groups. In *Advances in Cryptology - ASIACRYPT*, volume 3329, pages 245–259. LNCS, 2004.
19. Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, July 2004.

20. Yongdae Kim, Adrian Perrig, and Gene Tsudik. Tree-based group key agreement. *ACM Trans. Inf. Syst. Secur.*, 7(1):60–96, 2004.
21. Lijun Liao and Mark Manulis. Tree-based group key agreement framework for mobile ad-hoc networks. In *AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06)*, pages 5–9, Washington, DC, USA, 2006. IEEE Computer Society.
22. Mark Manulis. Contributory Group Key Agreement Protocols, Revisited for Mobile Ad-Hoc Groups. In *Proceedings of 2nd IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 2005), International Workshop on Wireless and Sensor Networks Security (WSNS 2005)*, pages 811–818. IEEE Computer Society, 2005.
23. Mark Manulis. Key agreement for heterogeneous mobile ad-hoc groups. In *ICPADS '05: Proceedings of the 11th International Conference on Parallel and Distributed Systems - Workshops (ICPADS'05)*, pages 290–294, Washington, DC, USA, 2005. IEEE Computer Society.
24. J. Nam, J. Lee, S. Kim, and D. Won. DDH based group key agreement for mobile computing. http://eprint.iacr.org/2004/127, 2004.
25. R. Ogier, F. Templin, and M. Lewis. Topology dissemination Based on Reverse-Path Forwarding (TBRPF), February 2004. RFC 3684, Experimental.
26. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-demand Distance Vector (AODV) routing, July 2003. RFC 3561, Experimental.
27. A. Perrig. Efficient collaborative key management protocols for secure autonomous group communication. In *Proceedings of International workshop on cryptographic techniques and electronic commerce*, pages 192–202, 1999.
28. J. Pieprzyk and C.-H. Li. Multiparty key agreement protocols. *IEE Proceedings - Computers and Digital Techniques*, 147(4):229–236, 2000.
29. Ricardo Staciarini Puttini, Ludovic Me, and Rafael Timóteo de Sousa. Certification and authentication services for securing MANET routing protocols. In *Proceedings of the 5th IFIP TC6 International Conference on Mobile and Wireless Communications Networks*, Singapore, October 2003.
30. 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.
31. D.G. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A secure audio tele-conference system. In *Advances in Cryptology - CRYPTO*, volume 403, pages 520–528. LNCS, 1988.
32. M. Steiner, G. Tsudik, and M. Waidner. Diffie-hellman key distribution extended to group communication. In *Proceedings of 3rd ACM Conference on Computer and Communications Security*, pages 31–37. ACM Press, 1996.
33. W.-G. Tzeng and Z.-J. Tzeng. Round-efficient conference key agreement protocols with provable security. In *Advances in Cryptology - ASIACRYPT*, volume 1976, pages 614–627. LNCS, 2000.