Group Key Agreement in Ad hoc Networks

Raghav Bhaskar

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Group Key Agreement in Ad hoc Networks

Raghav Bhaskar

N° 4832

May 2003

THÈMES 2 et 1

ISSN 0249-6399 ISRN INRIA/RR--4832--FR+ENG

Group Key Agreement in Ad hoc Networks

Raghav Bhaskar

Thèmes 2 et 1 — Génie logiciel
et calcul symbolique — Réseaux et systèmes
Projets CODES et ARLES

Rapport de recherche n° 4832 — May 2003 — 34 pages

Abstract: As the ubiquitous and pervasive computing paradigms gain popularity, Mobile Ad hoc NETworks (MANET) are receiving the increased attention of the research community. Essentially a MANET is a collection of mobile nodes communicating over wireless channels with little (if any at all) fixed, wired infrastructure. Such networks present a number of new challenges to many cryptographic techniques for collaborative group communication including Group Key Agreement. In this report are presented these challenges, some Group Key Agreement protocols proposed for Collaborative Groups and a discussion on their suitability for the MANET environment.

Key-words: key agreement, Ad hoc networks, secure group communication, security protocols

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

Accord de Cle de groupe dans des réseaux Ad hoc

Résumé : Comme les paradigmes des reseaux onmiprésents et dominants gagnent en popularité, les réseaux Ad Hoc mobiles (MANET) revoivent la plus grande attention de la communauté des chercheurs, Essentiellement, un Manet est un collection de noeuds mobiles qui communiquent sur des canaux sans fil avec peu (voire pas du tout) d'infrastructure fixe. De tels réseaux présentent un certain nombre de défis dans le domaine cryptographique pour de la communication groupe collaborative, comprenant notamment la mise en accord de clé de groupe. Dans ce rapport ces défis sont présentés, quelques protocoles de mise en accord de clé de groupe et une discussion sur leur adéquation pour les environnement MANET.

Mots-clés : Accord de Clé, réseaux Ad hoc, sécurité de communication de groupe, protocoles de sécurité

Contents

1 Introduction

As the computing power available per unit dollar of money increases by the day, there is a growing tendency to embed a microprocessor in more and more consumer devices. As a result we are being surrounded by "smart" devices which can perform some really powerful computations. At the same time, advancements in communication technologies especially wireless communication are ensuring that these devices do not remain isolated but are able to talk to each other and share data and services. For instance consider the case of Smart Homes [10]. It is envisioned that in the future there would be homes where there would exist a communication infrastructure enabling all the devices to talk to each other and each device would not have to replicate features which already exist in other devices. Thus a door entry system would use a video camera to spot a visitor, broadcast that information on the TV if there is someone at home or otherwise on a mobile phone. It would accept commands via a remote (or a phone) to take appropriate action. The visitor could leave a message on an answering machine if there is no answer. Thus we have a number of devices providing as well as accessing each other's services. In such a collaborative group the relationships between the devices can be aptly defined by the "peer to peer" paradigm where the devices communicate with others as and when the need arises without the intervention of any central authority. Thus we have what are known as Ad hoc Networks, a collection of nodes communicating over wireless channels in the absence of any fixed, central infrastructure.

Ad hoc Networks are a step closer to realizing the "anytime , anywhere" computing paradigm. The term Ad hoc networks is most often used in the context of mobile devices, and in that case referred to as MANET (Mobile Ad hoc NETwork). They are often characterized by the following:

1) Network topology is dynamic.
2) Nodes have a limited power capacity.
3) There is limited wireless bandwidth for communication.
4) Absence of any central authority.

Ad hoc networks are increasingly becoming popular due to the ease and speed of their deployment. Though initially meant to be used for military applications, they are finding new applications in the personal area networking domain (e.g. cellphone, laptop, wristwatch), civilian environments (e.g. taxi cab network, meeting rooms, smart homes) etc. While most Ad hoc networks demonstrate the above mentioned characteristics, one Ad hoc network may differ greatly from another with respect to the computational capabilities, dynamic patterns and traffic characteristics of the member nodes. Even in a particular Ad hoc network, all nodes may not have identical capabilities and responsibilities.

But before such networks can be put to "high-value" use, security mechanisms have to be provided to ensure safe and reliable communication. For instance in the smart home scenario one would not want one's neighbor to be able to control the devices in your home

with his remote. Due to the inherent nature of Ad hoc networks they pose a number of challenges to the task of securing them, including:

1) **Communication constraints**: Communication in Ad hoc networks has a limited transmission range as well as is a big drain on its battery.
2) **Limited computational capabilities**: More often the nodes in an Ad hoc network have much less computational capabilities compared to an average desktop system.
3) **Lack of Central infrastructure**: Most Ad hoc networks have no or very little fixed, central infrastructure to make use of.
4) **Ease of eavesdropping**: Communication is on wireless channels which can be received by anyone in the transmission range.
5) **Packet loss**: Loss of packets is expected especially in multi-hop networks.
6) **Highly dynamic**: The network topology is highly dynamic with frequent partitions and merges.
7) **Tampering of Devices**: The devices forming the Ad hoc network have a higher risk of being tampered with because of their mobility in unsecured places.

These challenges, unseen in any existing collaborative group application (as on the Internet), have turned many traditional solutions redundant and forced the research community to rethink on some of the basic techniques in cryptography.
Ongoing work in security of Ad hoc networks can be broadly classified into the following categories.

1) **Trust and Key Management**: Lack of a priori shared secrets and absence of a central authority makes the task of key management complicated. Also in devices lacking computational ability to perform cryptographic operations, distributed trust models are being proposed. For example, see [16],[12].
2) **Secure Routing**: In multi-hop networks nodes have to rely on peer nodes for routing thus making it prone to attacks if a node is compromised. Thus a need to secure the routing process. For example, see [17],[13].
3) **Availability**: Due to the dynamic nature of the network, ensuring the services are available across the network is an important ask. For example, see [3],[5].
4) **Cryptographic Protocols**: Computational and communication constraint nature of the network devices makes the design of protocols a critical issue. For example, see [4].

In this report we concentrate on key management, especially Key Agreement in Ad hoc networks which is an essential step in securing Ad hoc networks. The rest of the report is organized as follows: In Section 2 we introduce some basic concepts of Cryptography, in Section 3 we present some group key agreement protocols proposed for group communication along with an analysis of their computational and communicational complexity which is of key importance in the study of their suitability for resource-constrained Ad hoc environments. In Section 4 we compare these protocols (in terms of communicational and

computational complexity) and discuss their suitability for Ad hoc environments and finish with the conclusions in Section 5.

2 Cryptographic Concepts

Cryptography [1] is defined to be the study of the mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication. These goals are defined as follows:

1. **Confidentiality** means to keep away the content of information from unintended recipients.

2. **Data integrity** means to prevent unauthorized alteration of data. Alteration of data includes such things as insertion, deletion and substitution.

3. **Authentication** is a process by which one can verify the identity of a claimant. While entity authentication is about two (or more) parties (entering into a communication) being able to identify each other, data origin authentication is concerned with the authentication of information delivered over a channel as to its origin, date of origin, data content, time sent, etc.

4. **Non-repudiation** is a process which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted. A procedure involving a trusted third party is needed to resolve the dispute.

We next discuss some common ways of achieving these goals.

2.1 Encryption

A cryptographic solution to achieve confidentiality is using **encryption**. An Encryption function maps a plain text M into a cipher text (meaningless data) C for a given **key**, e. A key is thus a compact way to specify the encryption function (from the set of all encryption functions) to be used. Before any sense can be made out of the cipher text, it needs to be decrypted using a decrypting key, d. Thus the encryption and decryption functions along with the key pair (e, d) define an encryption scheme, (E_e, D_d). An encryption scheme is said to be **symmetric-key** if for each associated key pair (e, d), it is computationally "easy" to determine d knowing only e, and to determine e from d. Most often $d = e$. So two parties wishing to communicate securely need to share the key e over some secure channel before they can use the encryption scheme to communicate over an insecure channel. In contrast, in **asymmetric-key** systems it is infeasible to determine d given e. Thus every user in such

a system has a key pair (e, d) which is unique to him/her. While d is known only to that user (say *Alice*), e is made public. Thus anyone wishing to communicate securely with *Alice* can send her data encrypted using key e. Only *Alice* can decrypt it using d. Thus, such a system does away with the need for a secure channel at any time. While symmetric-key techniques are much faster than asymmetric ones, they require the parties to have a pre-shared secret. Thus a common solution is to exchange a symmetric key using asymmetric techniques. This holds particularly true for Ad hoc networks as the use of asymmetric-key cryptography for securing all communication is practically impossible.

2.2 Digital Signatures

A cryptographic primitive which helps achieve the goals of Data integrity, Authentication as well as Non-repudiation is the **digital signature**. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of signing entails transforming the message and some secret information held by the entity into a tag called a signature. A generic description follows.

1) M is the set of messages which can be signed.
2) S is a set of elements called signatures, possibly binary strings of a fixed length.
3) S_A is a transformation from the message set M to the signature set S, and is called a signing transformation for entity A. The transformation S_A is kept secret by A, and will be used to create signatures for messages from M.
4) V_A is a transformation from the set $M * S$ to the set $\{true, false\}$. V_A is called a verification transformation for A's signatures, is publicly known, and is used by other entities to verify signatures created by A.

Thus entity A (the signer) creates a signature for a message M by computing $s = S_A(M)$ and transmitting the pair (M, s). s is called the signature for message M. To verify that a signature s on a message M was created by A, an entity B (the verifier) obtains the verification function V_A of A and computes $u = V_A(M, s)$ and accepts the signature as having been created by A if $u = true$, and rejects the signature if $u = false$. Thus a digital signature helps achieve data integrity (if data is altered the signature wouldn't match), authentication (the data came from the signer) and non-repudiation (the signer cannot refuse one's signature on a message).

Digital signatures based on asymmetric-key systems require the public key of the signer to be made available to any verifier. Two common ways of achieving it are either by making available an online certification authority to provide authenticated copies of a user's public key or by a digital certificate (issued by the certification authority) binding a public key to its owner.

2.3 Key Establishment

We have seen that a key is an essential component of most cryptographic operations. Thus designing the set of techniques and procedures supporting the establishment and maintenance of keys between authorized parties (known as **Key Management**) is an important task in itself. In this section we study in detail **Key establishment**, which is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use. Thus it deals with the basic task of making available to the user the keys required for all future cryptographic operations.

Key establishment may be broadly subdivided into key transport and key agreement. A **key transport** protocol or mechanism is a key establishment technique where one party creates or otherwise obtains a secret value, and securely transfers it to the other(s). A **key agreement** protocol or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) such that no party can predetermine the resulting value. Thus in scenarios, where there is no central authority and the task of key generation cannot be assigned to a single (or few) participant(s)(as is most often the case in Ad hoc networks), key agreement is a good alternative to key transport. But more often in a key agreement protocol, it is required that a participating member be assured that no other party aside from a specifically identified party (or parties) may gain access to a particular secret key. Such a protocol is known as an **authenticated key agreement** protocol. It is worth noting here that the term authentication, in this context, means to have the knowledge of the identity of parties who may gain access to the key. To corroborate the fact that the same identities actually participated in the protocol, one has to rely on other mechanisms like entity authentication. Key agreement protocols can be designed using symmetric or asymmetric techniques. To further illustrate the above concepts we present the two-party Diffie Hellman key agreement protocol and then its authenticated version.

2.3.1 Diffie Hellman Key Agreement

This is a key agreement protocol (for two parties) based on asymmetric key techniques. It can be summarized as follows:
Two parties A (Alice) and B (Bob) (who have no pre-shared secrets) agree on an appropriate prime p and a generator α of $Z_p^*(2 \leq \alpha \leq p-2)$. They perform the following actions each time a shared key is required.
(**Notation**: $A \longrightarrow B : X$ denotes A sending B the quantity X over a public channel.)
1a) A chooses a random secret $x, 1 \leq x \leq p-2$
1b) $A \longrightarrow B : \alpha^x \bmod p$
2a) B chooses a random secret $y, 1 \leq y \leq p-2$
2b) $B \longrightarrow A : \alpha^y \bmod p$
B computes the shared key as $K = (\alpha^x)^y \bmod p$ and A computes the shared key as $K = (\alpha^y)^x \bmod p$. Thus both end up with the common secret $\alpha^{xy} \bmod p$. For adequate

security, the prime p should be around 1024 bits in size. Thus the exchanged key is of approximately the same size. The security of this protocol lies in the hardness of the Discrete Log problem (which is to calculate x given $\alpha^x \bmod p$) for such large numbers. If one uses groups based on elliptic curves one can get the same level of security with 160 bit sized keys.

But this protocol is susceptible to the following man-in-the-middle attack. An intruder Carol C can intercept messages from both A and B and change them to establish individually keys with both A and B, while both A and B think that they have established a key between themselves. To prevent this kind of attack (active attack) it is required that the participants be authenticated. Thus we have the following protocol:

1a) A chooses a random secret $x, 1 \leq x \leq p-2$
1b) $A \longrightarrow B : \alpha^x \bmod p$
2a) B chooses a random secret $y, 1 \leq y \leq p-2$
2b) $B \longrightarrow A : \alpha^y \bmod p, E_k(S_B(\alpha^y, \alpha^x))$
3a) $A \longrightarrow B : E_k(S_A(\alpha^x, \alpha^y))$

where $S_B(\alpha^y, \alpha^x)$ is B's signature on α^y, α^x. $E_k()$ is some symmetric-key encryption scheme employing the key $k = \alpha^{xy}$. Thus by verifying the signatures the two parties can corroborate the fact that the message was indeed generated and sent by the claimed entity.

2.3.2 Group Key Agreement

Key agreement protocols for more than two parties are known as Group Key Agreement (GKA) protocols. These get a bit involved as the two party Diffie-Hellman key agreement does not extend trivially to more than 2 parties. They are well suited to the security needs of small, dynamic, collaborative groups as they offer the possibility of creating session keys for each group session, thus adjusting well to group membership changes. Key features of a GKA (Group Key Agreement) protocol are:

1) **Contributory**: GKA protocols are contributory in nature which means that all members participating in the protocol contribute towards the secret and even in the absence of one contribution it is infeasible to derive the secret.

2) **Lack of a central authority**: There is no single member controlling the execution of the protocol. Even if there is some "leader" it is short-lived and restricted to that particular execution of the protocol.

3) **Key Freshness**: The secret (key) derived cannot be predicted in advance (even by one of the protocol participants).

The key derived from the GKA protocol needs to meet the following security features:

1) **Group key secrecy**: It simply means that the derived secret should not be derivable by a non-participant.
2) **Forward key secrecy**: Merely knowing one of the current group keys, one should not be able to compute previous group keys.
3) **Backward key secrecy**: Merely knowing one of the current group keys, one should not be able to compute future group keys.
4) **Key independence**: Knowledge of a subset of group keys in the life-cycle of a group (by a participant or outsider) should not enable knowledge of any key outside this subset.
5) **Perfect forward secrecy**: Knowledge of a long-term secret should not enable one to compute past group keys.

Thus in the context of Ad hoc networks, GKA protocols provide a mechanism to derive a (symmetric) group key in the absence of a third party.

2.4 Provable Security

But given a cryptographic protocol/primitive how do we conclude that it is secure ? Till a few decades ago, only heuristic proofs were provided wherein one tried to provide any variety of convincing arguments to show that any successful attack on the protocol would require resources beyond that available to any adversary. And if the protocol was able to withstand attacks over "long" periods of time, it was considered to be secure. It was not uncommon for the protocols to be found flawed years after they had been approved for "high-value" applications, thus resulting in severe security hazards and expensive re-employment. Recently there has been an upsurge in the study of security proofs of protocols.
Provable Security [9],[2] is concerned with proving security protocols secure. The basic steps involved in arriving at the proof are:

1) Define the protocol.
2) Define the "adversary" and its capabilities.
3) Define what it means by "the protocol is secure".
4) Provide a "reduction" from the attack on the protocol to an attack on the underlying "hard" problem.

So essentially we try to prove that under a given set of assumptions (as assumed in the model) any successful attack on the protocol will result in a successful attack on the underlying well-known "hard" problem (like the Discrete Log problem). Thus further assuming that the underlying "hard" problem is secure we say that the given protocol is secure. The concept of provable security is still evolving, with special efforts being made to make the proofs more useful practically. But still any "provable secure" protocol can be considered more secure than a protocol with none.

3 Group Key Agreement Protocols

Group Key Agreement protocols find applications in many group applications including telephone and video conferences, remote consultation and diagnosis systems for medical applications, contract negotiation, multi-party games, collaborative work places, electronic commerce environments such as on-line real-time auctions, and information dissemination of stock quotes. Many GKA protocols have been proposed in the literature. While some are only suitable for static groups others work in case of certain kinds of groups only (for instance groups with certain number of members or groups with the ability to listen to multiple broadcasts in a single round). While security flaws have been found in some others. We present here protocols which work in case of dynamic groups and are very generic in their assumptions about the group sizes and dynamics and with no known security flaws in them. Each protocol is defined in terms of the following operations:
1) **Initial Key Agreement (IKA)**: This refers to the setup stage when a number of new users decide to derive a new group key.
2) **Auxiliary Key Agreement (AKA)**: This refers to the group modification procedures (after a group is formed). It is essential that each of these operations lead to a change in the group key (to maintain key independence). These operations are:

a) Join: One new member wishes to join an already established group.
b) Delete: A member leaves (voluntarily or otherwise) a group.
c) Partition: A group is divided into two (or more) smaller groups. It can also be viewed as a "Delete" of more than one member.
d) Merge: Two (or more) groups get together to form a single group. It can also be viewed as a "Join" by more than one member.

In what follows we define the protocols in terms of the above mentioned operations. For the tree-based protocols, IKA operation has not been provided by the authors, but it is not hard to devise one by making n Join calls. Also the tree-based protocols lack authentication, thus we have not included the cost incurred in providing authentication in the analysis of each of the protocols. Figures (where needed) have been provided to illustrate a representative example. An analysis of the computational and communicational complexity is also provided.

3.1 Cliques-I

Cliques-I [8] assumes the following setting: n members M_1,M_2,...M_n who have previously agreed on G (a cyclic sub-group of Z_p^* of order q, $p = k * q + 1$) and α generator of G. Also each M_i shares[1] a secret with M_n, K_{in}. This secret enables the user to compute the group key from the final downflow message thus adding authentication to the protocol.

[1] This secret could be derived from the public keys of each other by doing a two-party Diffie-Hellman key agreement thus not requiring the existence of a secure channel.

Round i $(0 < i < n)$ $(n-1$ Upflow Messages)
1) M_i selects random $r_i \in Z_q^*$
2) $M_i \longrightarrow M_{i+1} : \{\alpha^{(r_1 * r_2 \ldots r_i)/r_j} : j \in [1,i]\}, \alpha^{r_1 * r_2 \ldots r_i}$
Round n (Downflow Message)
1) M_n selects random $r_n \in Z_q^*$
2) $M_n \longrightarrow$ ALL: $\{\alpha^{K_{jn} * (r_1 * r_2 \ldots r_n)/r_j} : j \in [1, n-1]\}$

Table 1: Cliques-I IKA Algorithm

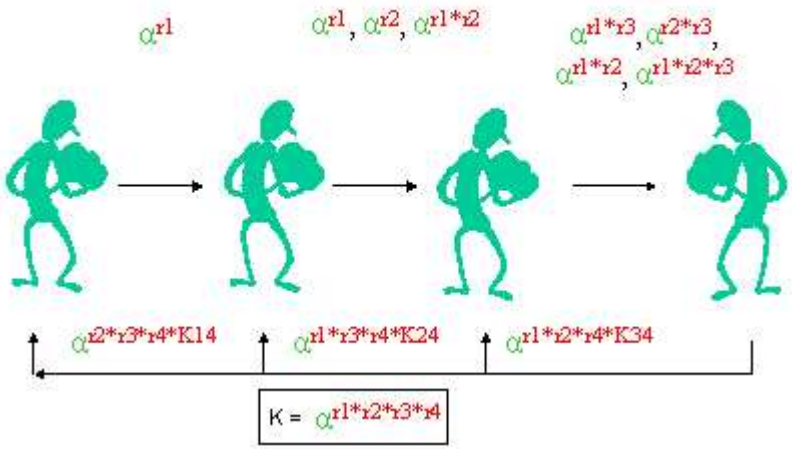

Figure 1: IKA in Cliques-I

3.1.1 IKA

IKA consists of two stages (see Table 1): upflow (Round 1 to $n-1$) and downflow (Round n). In the upflow stage, contributions from all the group members are collected. Each member M_i receives i values, $i-1$ of which are intermediate and 1 cardinal. M_i contributes its secret and passes on i intermediate and 1 cardinal values to M_{i+1}. In the downflow stage, M_n (also known as the controller) performs a similar operation and broadcasts the result to the group. Each member M_i picks the value meant for him (i.e. $\alpha^{K_{in} * (r_1 * r_2 \ldots r_n)/r_i}$) and raises it to the product of his secret contribution and the inverse of K_{in} to get the group secret. The group secret thus derived is $K_G = \alpha^{r_1 * r_2 * \cdots * r_n}$. See Fig. 1 for an example with 4 participants.
Table 2 presents the communicational and computational complexity of IKA in Cliques-I. The communicational complexity is defined in terms of the number of rounds, number of

Communication	
Rounds	n
Messages	n
Unicast	n-1
Broadcast	1
Computation	
No. of Exponentiations	$i+1$ for M_i $(i<n)$, n for M_n

Table 2: Cliques-I IKA Analysis

Round 1 (Upflow message meant for the new member)
1) M_n selects random $\underline{r_n} \in Z_q^*$
2) $M_n \longrightarrow M_{n+1} : \{\alpha^{(r_1 * r_2 \ldots \underline{r_n})/r_j} : j \in [1,n]\}, \alpha^{r_1 * r_2 \ldots \underline{r_n}}$

Round 2 (Downflow message for all)
1) M_{n+1} selects random $r_{n+1} \in Z_q^*$
2) $M_{n+1} \longrightarrow$ALL: $\{\alpha^{K_{jn+1} * (r_1 * r_2 \ldots r_{n+1})/r_j} : j \in [1,n]\}$

Table 3: Cliques-I Join Algorithm

total messages exchanged, the number of unicast messages and the number of broadcast messages in the protocol run of n users. The computational complexity specifies the number of exponentiations performed by each user in the protocol execution. As all protocols studied in this report are not authenticated, we have ignored the computation cost of authentication to be able to make a fair comparison.

3.1.2 Join/Merge

When a new member M_{n+1} arrives, the old controller M_n generates a new secret exponent and forwards a "IKA-upflow like" message to M_{n+1}. M_{n+1} generates its secret, random value and broadcasts to the group an "IKA-downflow like" message. See Tables 3 and 4 and Fig. 2 for details.
While in Merge (considered as the joining of k new members) M_{n+1} receives the Join-response like message from the old controller, and thereafter an IKA protocol is executed amongst the new members and finally M_{n+k} broadcasts a downflow message which enables all members (old and new) to compute the new group key. We just give an example (Fig. 3) and efficiency analysis (Table 5) and omit the algorithm as it is easy to derive from Join and IKA protocols.

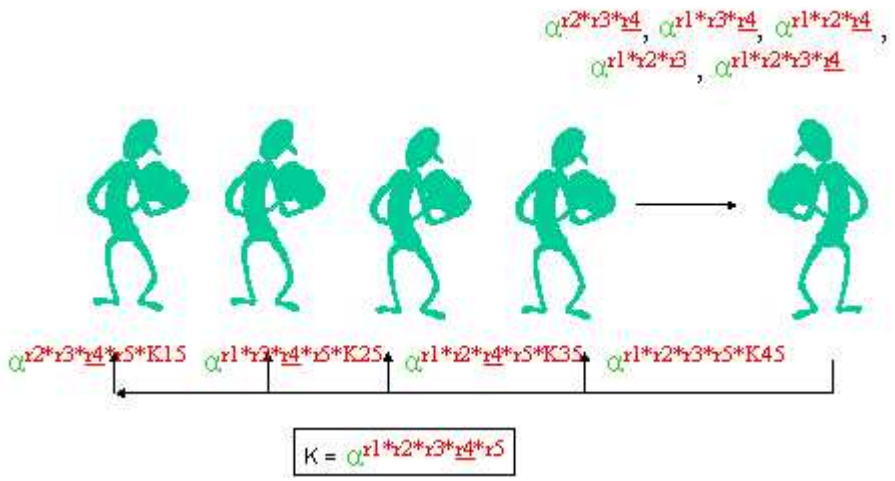

Figure 2: Join in Cliques-I

Communication	
Rounds	2
Messages	2
Unicast	1
Broadcast	1
Computation	
No. of Exponentiations	1 for M_i $(i < n)$, $n+1$ for $M_n \& M_{n+1}$

Table 4: Cliques-I Join Analysis

Communication	
Rounds	$k+1$
Messages	$k+1$
Unicast	k
Broadcast	1
Computation	
No. of Exponentiations	$n+1$ for M_n, $n+i$ for M_{n+i} $(1 < i < k)$, 1 for others

Table 5: Cliques-I Merge Analysis

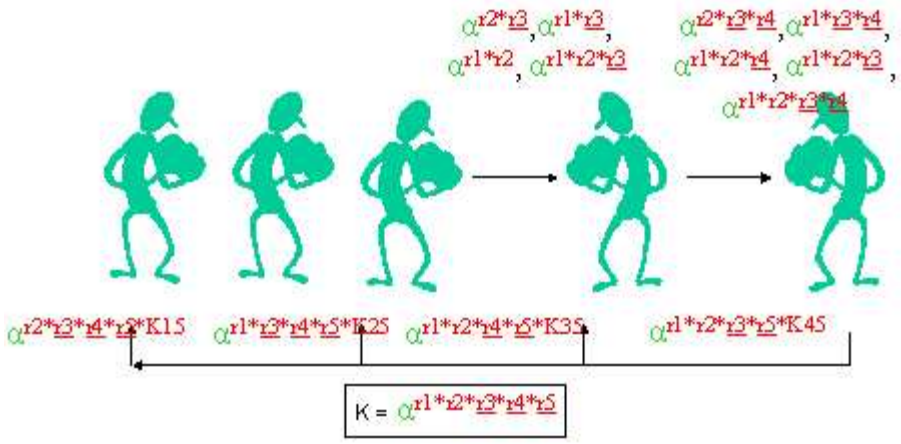

Figure 3: Merge in Cliques-I

Round 1 (Downflow message by the Highest-indexed member)
1) M_{HI} selects random $r_{HI} \in Z_q^*$
2) $M_{HI} \longrightarrow$REMAINING: $\{\alpha^{K_{jHI}*(r_1*r_2...r_{HI})/r_j} : j \in$RM$\}$

Table 6: Cliques-I Leave/Partition Algorithm

3.1.3 Leave/Partition

In a leave or partition (see Table 6), the remaining highest-indexed member chooses a random secret and broadcasts a downflow message with values which enable merely the non-leaving members to compute the group key. For example in Fig. 4, when M_4 leaves, then M_5 generates a new secret $\underline{r_5}$ and broadcasts the message to the remaining members RM= $\{M_1, M_2, M_3\}$ with the value $\alpha^{r_1*r_2*r_3*\underline{r_5}*K45}$ missing.

3.2 Cliques-II

Cliques-II [11] minimizes the computation performed by each member in IKA (and so also Merge) at the expense of more communication messages than Cliques-I.

3.2.1 IKA

In the upflow round (see Table 8), each member raises the received message to its secret. Thus instead of i exponentiations (for the i^{th} member) as in Cliques-I, one has to perform

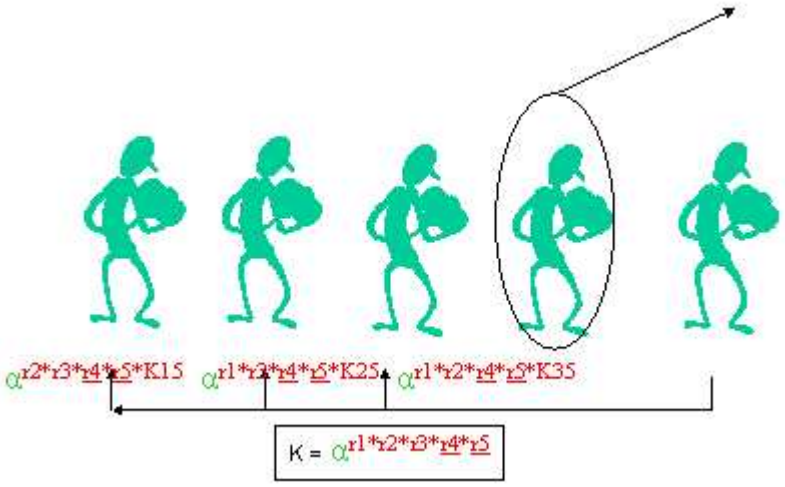

Figure 4: Leave/Partition in Cliques-I

Communication	
Rounds	1
Messages	1
Unicast	0
Broadcast	1
Computation	
No. of Exponentiations	1 for M_i $(i < HI)$, $n-k$ for M_{HI}

k- Number of leaving members

Table 7: Cliques-I Leave/Partition Analysis

Round $i (0 < i < n-1)$ (Upflow messages)
1) M_i selects random $r_i \in Z_q^*$
2) $M_i \longrightarrow M_{i+1} : \{\alpha^{(r_1 * r_2 \ldots r_i)}\}$

Round $n-1$ (M_{n-1} broadcasts to all)
1) M_{n-1} selects random $r_{n-1} \in Z_q^*$
2) $M_{n-1} \longrightarrow$ALL: $\{\alpha^{(r_1 * r_2 \ldots r_{n-1})}\}$

Round n (Unicast message from all to M_n)
1) ALL $M_i \longrightarrow M_n : \{\alpha^{(r_1 * r_2 \ldots r_{n-1})/r_i}\}, i \in [1, n-1]$

Round $n+1$ (M_n broadcasts a downflow message to all)
1) $M_n \longrightarrow$ALL: $\{\alpha^{K_{in} * (r_1 * r_2 \ldots r_n)/r_i} : i \in [1, n-1]\}$

Table 8: Cliques-II IKA Algorithm

Communication	
Rounds	$n+1$
Messages	$2n-1$
Unicast	$2n-3$
Broadcast	2
Computation	
No. of Exponentiations	4 for M_i $(i < n-1)$, 2 for M_{n-1}, n for M_n

Table 9: Cliques-II IKA Analysis

simply one exponentiation. When this message reaches M_{n-1}, he does the same and broadcasts the resulting message to all. At this stage all members $(M_i, i < n)$ factor out their contribution (by raising the received message to the inverse of their secret) and send (unicast) it to M_n. M_n computes a downflow message (just like in Cliques-I) and broadcasts it to all.

Join and leave/partition protocols for Cliques-II are same as in Cliques-I.

3.2.2 Merge

Merge is similar to the one in Cliques-I with the Cliques-II IKA being used by the new members instead. We provide the efficiency analysis in Table 10.

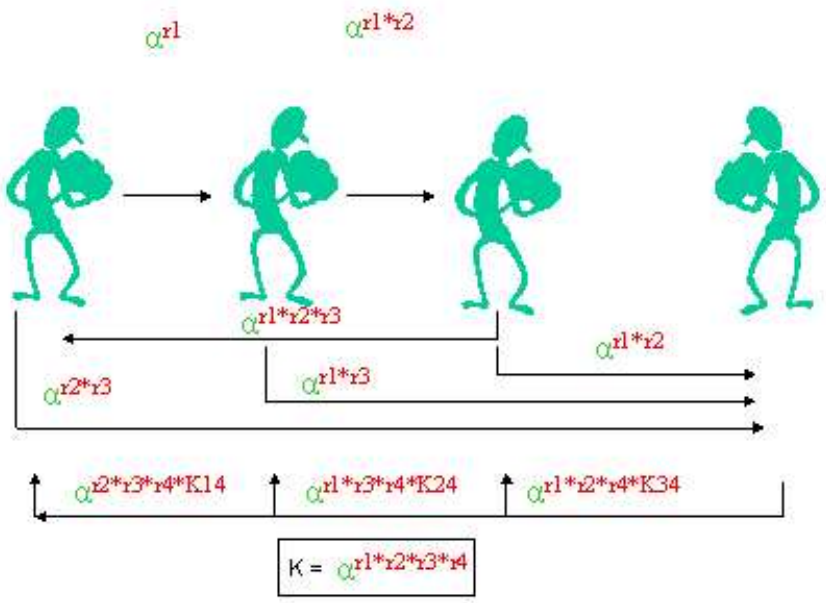

Figure 5: IKA in Cliques-II

Communication	
Rounds	$k+2$
Messages	$n+2k$
Unicast	$n+2k-2$
Broadcast	2
Computation	
No. of Exponentiations	2 for M_i $(i<n)$, 3 for M_n, 2 for M_{n+i},$n+k$ for M_{n+k}

Table 10: Cliques-II Merge Analysis

3.3 Provably Secure Cliques

Referred to as AKE [7] (Authenticated Key Exchange) in the remainder of this report, this protocol modifies Cliques-I to derive an authenticated protocol. It also does away with the need to have any pre-shared secrets between the members. Also it is the only protocol in its category which is provably secure. The key to achieve an authenticated and provably secure protocol is the use of digital signatures to sign all messages exchanged during the protocol run. Thus it requires each protocol participant to have a pair of public-private keys to sign their messages.
So, in general the messages in AKE look like:

Upflow Message
$msg.uf = \{Data, Sig(Data)\}$
$Data = \{msg.id, members, msg.cliques\}$

where
$msg.id$ = Message Identification Number
$members$ = List of names of all participants
$msg.cliques$ = Upflow message as in Cliques-I
and $Sig(Data)$ is a digital signature on $Data$.

Downflow Message
$msg.df = \{Data, Sig(Data)\}$
$Data = \{msg.df.old, msg.id, members, msg.cliques\}$
where
$msg.df.old$ = Last Downflow Message in the group

[7] provides a proof which demonstrates that any successful attack on the protocol would require either a digital signature forger or the Computational GDH problem (computing α^{xy} given α^x and α^y in a finite group) instance solver.

3.4 Tree Based Key Agreement - I

Tree Based Key Agreement - I (or TGDH - Tree Group Diffie Hellman) [14] protocol blends the concept of key trees with Diffie-Hellman key exchange. We begin by explaining the concept of a key tree with an example.
In Fig. 6 we have a key tree hosting 6 members. The root is located at level 0 and the lowest leaves are at level $h(=3)$. Since we use binary trees, every node is either a leaf or a parent of two nodes. The nodes are denoted (l, v), where $0 \leq v \leq 2^l - 1$ since each level l hosts at most 2^l nodes. Each node (l, v) is associated with the a secret key $K_{(l,v)}$ and its blinded version $BK_{(l,v)} = f(K_{(l,v)})$ where the function $f()$ is modular exponentiation in prime order groups, i.e., $f(k) = \alpha^k \bmod p$. While $K_{(l,v)}$ is secret, $BK_{(l,v)}$ is public. Assuming a leaf node (l, v) hosts the member M_i, then the node (l, v) has M_i's secret key $K_{(l,v)}$. Furthermore, the

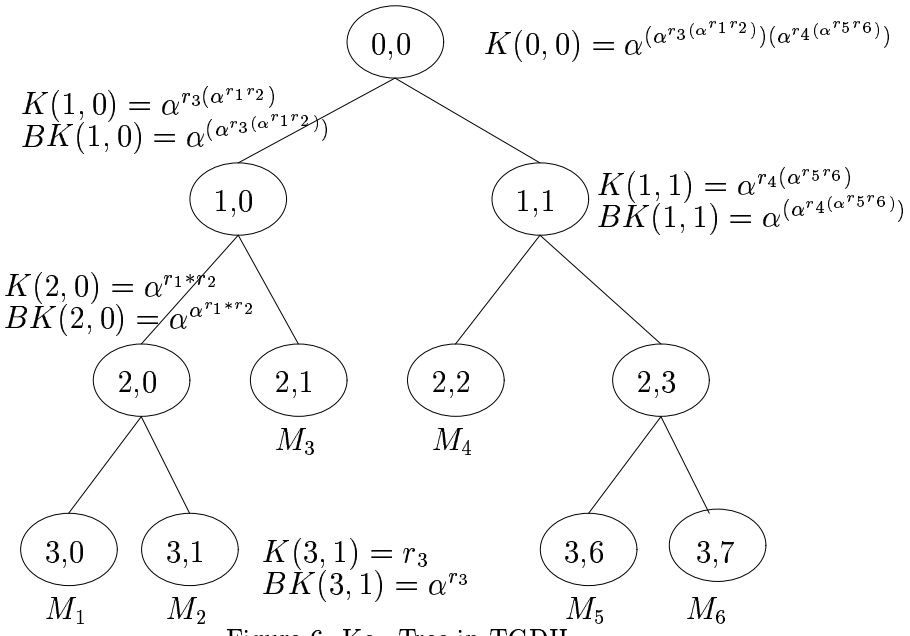

Figure 6: Key Tree in TGDH

member M_i at node (l, v) can compute every key along the path from (l, v) to $(0, 0)$, referred to as the $key-path$ and denoted $KEY_i^\star$. Every key $K_{(l,v)}$ is computed recursively as follows:

$$\begin{aligned} K_{(l,v)} &= (BK_{(l+1,2v+1)})^{K_{(l+1,2v)}} \bmod p \\ &= (BK_{(l+1,2v)})^{K_{(l+1,2v+1)}} \bmod p \\ &= \alpha^{K_{(l+1,2v)} K_{(l+1,2v+1)}} \bmod p \\ &= f(K_{(l+1,2v)} K_{(l+1,2v+1)}) \end{aligned}$$

In other words, computing a key at (l, v) requires the knowledge of the secret key of one of the two children nodes and the blinded key of the other child node. $K_{(0,0)}$ at the root node is the group secret shared by all members. This value is never used as a cryptographic key for the purposes of encryption, authentication or integrity. Instead, such keys are derived from the group secret, e.g., by setting $K_{group} = h(K_{(0,0)})$ where h is a cryptographically strong hash function.
For example, in the figure, M_2 can compute $K_{(2,0)}$, $K_{(1,0)}$ and $K_{(0,0)}$ using $BK_{(3,0)}$, $BK_{(2,1)}$, $BK_{(1,1)}$ and $K_{(3,1)}$. The final group key $K_{(0,0)}$ is:

Communication	
Rounds	2
Messages	2
Unicast	0
Broadcast	2
Computation	
No. of Exponentiations	h(Height of tree)

Table 11: TGDH Join/Merge Analysis

$$K_{(0,0)} = \alpha^{(\alpha^{r_3(\alpha^{r_1 r_2})})(\alpha^{r_4(\alpha^{r_5 r_6})})}$$

The term co-path, denoted as $CO_i^\star$, is defined to be the set of siblings of each node in the key-path of member M_i. For example, the co-path $CO_2^\star$ of member M_2 in the figure is the set of nodes $(3,0),(2,1),(1,1)$. Consequently, every member M_i at leaf node (l,v) can derive the group secret $K_{(0,0)}$ from all blinded keys on the co-path $CO_i^\star$ and its secret key $K_{(l,v)}$.

3.4.1 Join/Merge

A new member M_{n+1} initiates the protocol by sending a join request message that contains its own blinded key $BK_{(0,0)}$. When current group members receive this message, they determine the insertion node in the tree. The insertion node is the shallowest rightmost node, where the join does not increase the height of the key tree. Otherwise, (if the key tree is well balanced), the new member joins to the root node. The sponsor is the rightmost leaf node in the subtree rooted at the insertion node. Next, the sponsor creates a new intermediate node and a new member node, and promotes the new intermediate node to be the parent of both the insertion node and the new member node. After updating the tree, the sponsor computes the new group key (since it knows all the necessary blinded keys) and broadcasts the new tree which contains all blinded keys. All other members update their trees accordingly and compute the new group key. A merge is similar, where instead of one member joining we have the smaller tree joining the larger one.

For example, in Fig. 7, M_4 makes a Join request, node M_3 acts as both the insertion and sponsor node and creates a new intermediate node $(1,1)$ with M_3 as its left child and M_4 as the right one. Thus the new tree looks as in Fig. 8.

3.4.2 Leave

Assume that we have n members and a member M_d leaves the group. In this case, the sponsor is the right-most leaf node of the subtree rooted at the leaving member's sibling

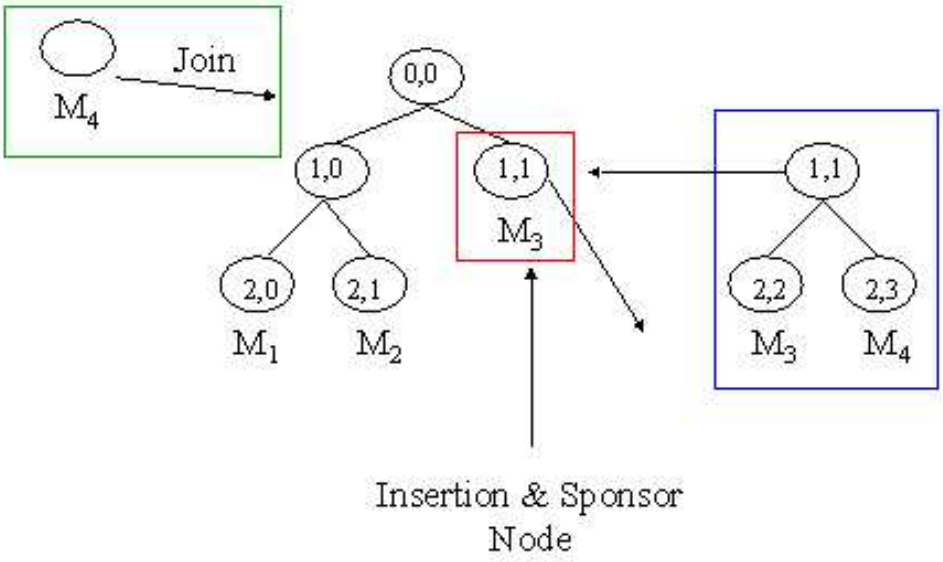

Figure 7: Join/Merge in TGDH

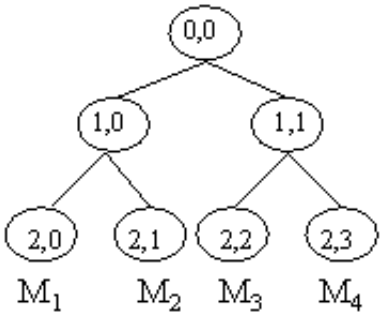

Figure 8: Join/Merge in TGDH - Final Tree

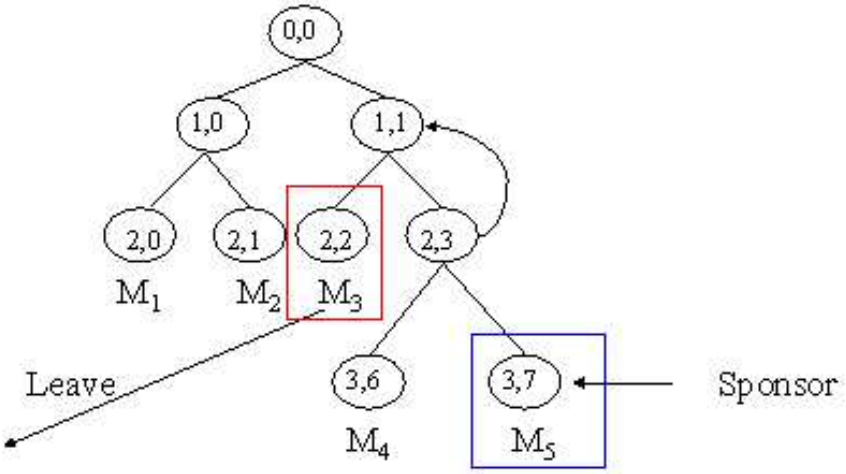

Figure 9: Leave in TGDH

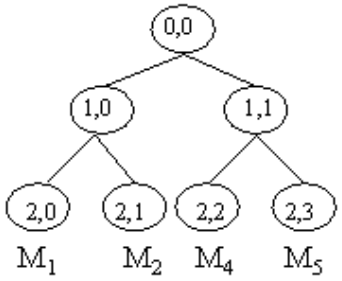

Figure 10: Leave in TGDH - Final Tree

node. In the leave protocol every member updates its key tree by deleting the leaf node corresponding to M_d. The former sibling of M_d is promoted to replace M_d's parent node. The sponsor picks a new secret share, computes all keys on its key path up to the root, and broadcasts the new set of blinded keys to the group. This information allows all members to recompute the new group key.
For example, in Fig. 9, node M_3 issues a Leave request, M_5 acts as the sponsor and deletes node M_3 and promotes the intermediate node $(2,3)$ to replace its parent $(1,1)$. Thus the new tree looks like as in Fig. 10.

Communication	
Rounds	1
Messages	1
Unicast	0
Broadcast	1
Computation	
No. of Exponentiations	h(Height of Tree)

Table 12: TGDH Leave Analysis

Communication	
Rounds	h
Messages	h
Unicast	0
Broadcast	h
Computation	
No. of Exponentiations	h(Height of Tree)

Table 13: TGDH Partition Analysis

3.4.3 Partition

A Partition is a multi-round protocol in which the Leave protocol is executed for each leaving node. It terminates when all members know the new blinded keys required to compute the group key.
For example, in Fig. 11, nodes M_1 and M_4 issue Leave request. Correspondingly M_2 and M_6 act as sponsors. M_2 deletes node M_1 and promotes itself to replace its parent $(2,0)$ while node M_6 deletes M_4 and promotes the intermediate node $(2,3)$ to replace its parent $(1,1)$.

3.5 Tree Based Key Agreement - II

Referred to as STR (Skewed Tree) [15] in the remainder, its approach is quite similar to TGDH, but differs in the nature of the tree used. A skewed tree where each node is either a leaf node or has two child nodes is employed. For instance in Fig. 13, we have a tree hosting five members each at a leaf position. Each intermediate node has another intermediate node as a left child and a member node as a right node. Computing a secret key at a node is identical to that in TGDH. Thus to compute the group key every member node has to know the blinded key of its peer intermediate node and the higher member nodes. The group key for this tree is $\alpha^{r_5\alpha^{r_4(\alpha^{r_3(\alpha^{r_1 r_2})})}}$

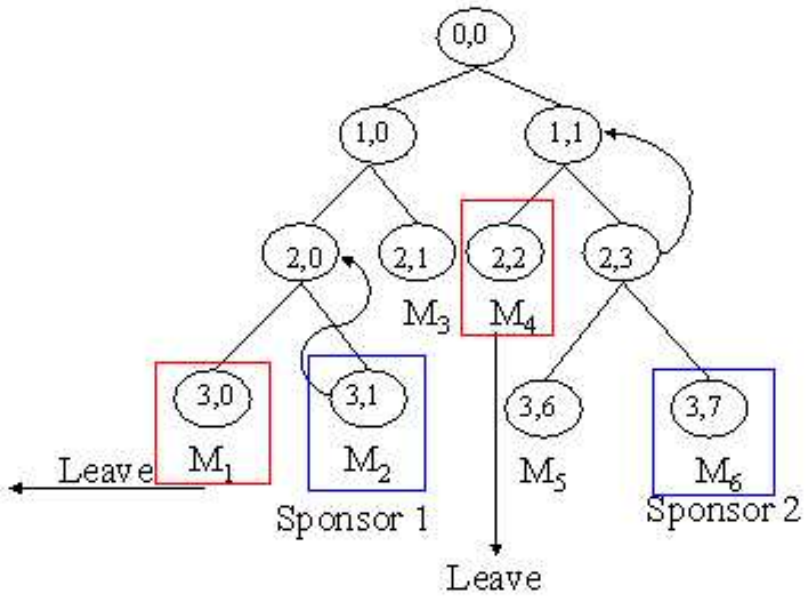

Figure 11: Partition in TGDH

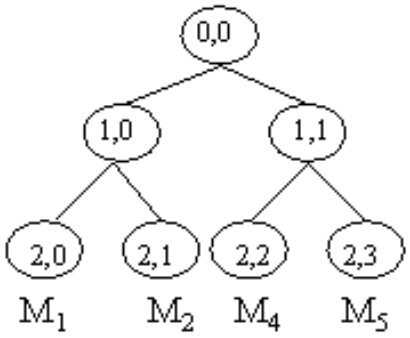

Figure 12: Partition in TGDH - Final Tree

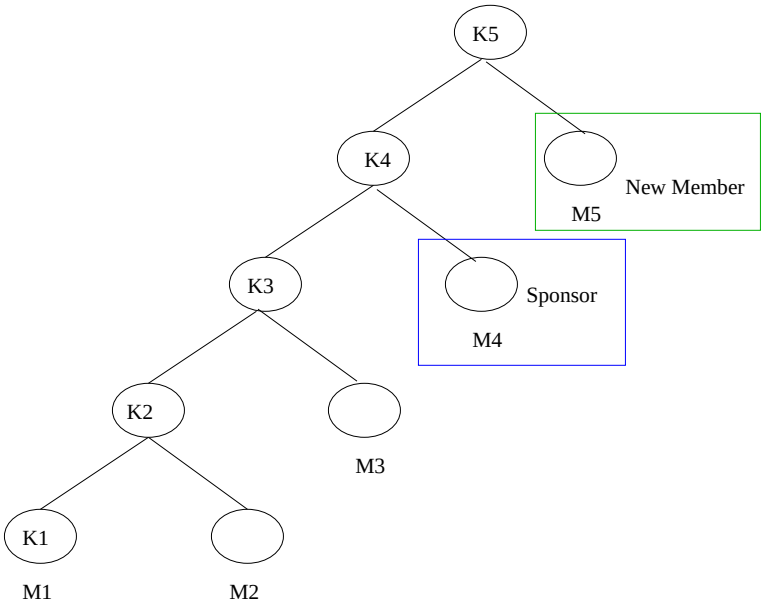

Figure 13: Join/Merge in STR

3.5.1 Join/Merge

The join procedure is quite simple; the incoming node simply joins at the root of the existing tree. The highest member in the existing tree acts as a sponsor and creates a new root node and makes the incoming node as its (the new root node's) right child and the existing tree as its left tree. It also changes its secret share and broadcasts the new blinded keys to the group. Merge is similar to Join. The smaller of the two trees joins the larger tree on the top. The highest member of the larger tree acts as a sponsor and broadcasts the new blinded keys.
For example, see Fig. 13.

3.5.2 Leave/Partition

We describe the Partition protocol, the Leave protocol is a simpler case of the same. All leaving nodes and their parent intermediate nodes are deleted from the tree. The member just below the lowest-indexed leaving member acts as the sponsor. It changes its secret share and broadcasts the new blinded keys.
For example, see Figs. 14 and 15. We propose the following n Rounds, n Messages protocol for IKA.
All the n members are assigned an index and a skewed tree with n leaf nodes and $n-1$ intermediate nodes is constructed with the lower-indexed members below the higher-indexed

Communication	
Rounds	2
Messages	2
Unicast	0
Broadcast	2
Computation	
No. of Exponentiations	Join: 1 for new, 2 for others Merge: $k+1$ for old , Distance from root for new

Table 14: STR Join/Merge Analysis

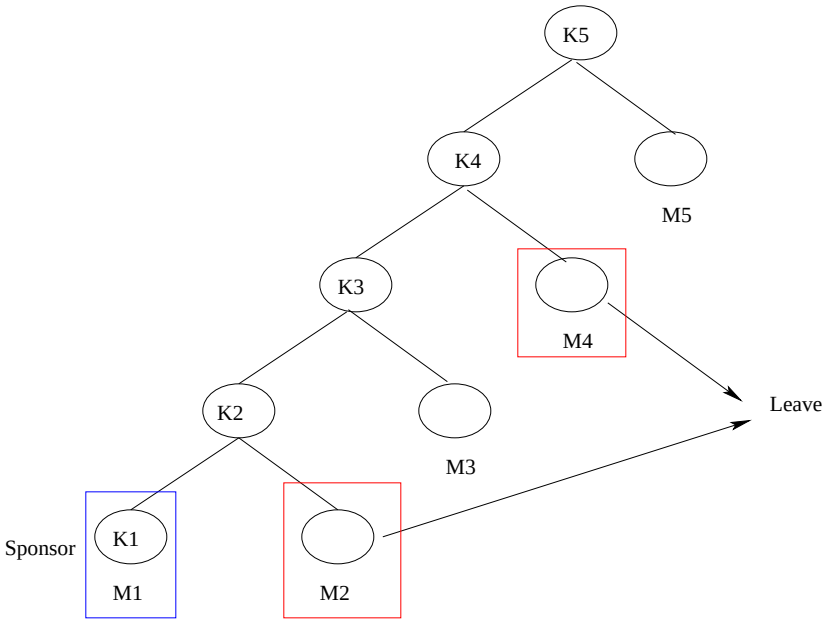

Figure 14: Leave/Partition in STR

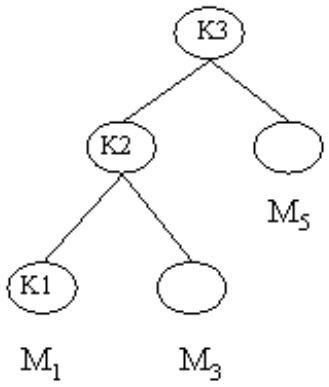

Figure 15: Leave/Partition in STR - Final Tree

Communication	
Rounds	1
Messages	1
Unicast	0
Broadcast	1
Computation	
No. of Exponentiations	* For Node i: 1+ $Min(d(\text{root,lsn}),d(\text{root},i))$

Table 15: STR Leave/Partition Analysis

* Min: Minimum function
d: distance function
root: root of the tree
lsn: Lowest numbered sponsor node

		Rounds	Msgs	Uni	Broad
Cliques-I	IKA	n	n	$n-1$	1
	Join	2	2	1	1
	Leave, Partition	1	1	0	1
	Merge	$k+1$	$k+1$	k	1
Cliques-II	IKA	$n+1$	$2n-1$	$2n-3$	2
	Join	2	2	1	1
	Leave, Partition	1	1	0	1
	Merge	$k+2$	$n+2k$	$n+2k-2$	2
TGDH	IKA	NP	NP	NP	NP
	Join, Merge	2	2	0	2
	Leave	1	1	0	1
	Partition	$O(logn)$	$O(logn)$	0	$O(logn)$
STR	IKA	NP	NP	NP	NP
	Join	2	2	0	2
	Leave, Partition	1	1	0	1
	Merge	2	3	2	1

Table 16: Communication Costs
n: Number of members forming(or in) a group.
k: Number of new members joining a pre-existing group.
NP: Not Provided.

ones. Thereafter, starting from M_1, each member broadcasts its blinded keys and the intermediate blinded keys that it is able to compute in a serial order. After the broadcast from M_n, everyone would be able to compute the group key.

4 Analysis of GKA Protocols for Ad hoc Networks

In this section we compare the GKA protocols, presented in the previous section, in terms of their communication and computation efficiency and study their suitability for Ad hoc environments. Table 16 summarizes the communication characteristics of the Group key agreement protocols. For each protocol the number of rounds, number of messages, number of unicast messages and number of broadcast messages for IKA, Join, Leave, Partition and Merge operations are provided. Note that the communication characteristics of AKE are identical to Cliques-I (with higher message sizes) and so not provided here. Also note that some of the messages are unicast rather than broadcast but that is of little consequence in the context of wireless networks as they operate by default in broadcast mode. As we can see from the table, STR protocol is the most communication-efficient of all GKA proto-

cols. Especially for Partition and Merge, it has constant order communication requirements.

Table 17 summarizes the computational characteristics of the Group key agreement protocols. For each protocol, the number of exponentiations per member and the serial number of exponentiations (The total number of exponentiations (for the group) that need to be executed in serial order) for IKA, Join, Leave, Partition and Merge operations are provided. Note that computational characteristics of AKE are similar to that of Cliques-I, except that the former requires one additional signature computation. Again figures for computation in IKA for STR and TGDH are not provided.
Cliques-II is the most computation efficient of all protocols requiring constant order exponentiations for most of the participants. It is particularly suitable for Asymmetric Ad hoc networks (networks where some devices have more computational capabilities than others) as bulk of the computation could be performed by the node with superior computational capabilities. For symmetric Ad hoc networks, TGDH may be a better choice.

Next we discuss certain issues which we think can further influence the choice of a GKA protocol in Ad hoc networks.

1) **Station to Station vs Message Independent protocols**: Tree based protocols enjoy the "message independence" feature which means that each message can be sent independent of the others. The Cliques suite of protocols are station to station based which means that there is ordering defined in the group. Each message is meant for some particular recipient (except the broadcast message). Though it should be feasible to have station to station protocols in Ad hoc groups, the message independence is more in tune with the nature of Ad hoc networks.

2) **Pre-Authentication**: In scenarios where entity authentication is required before the participants actually start a GKA protocol, it would be efficient to integrate the two (Entity Authentication and GKA) protocols. For instance instead of a Certificate based authentication protocol followed by a GKA protocol, we could have the participants pre-authenticate by broadcasting their certificates and then require them to sign messages during the GKA protocol.

3) **Provable Security**: Provably secure protocols have resilience to a large number of attacks and therefore must be the only choice for high-value applications.

4) **Trust**: It is clear that high-end cryptography cannot be employed in many Ad hoc environments (for example in case of Smart Dust [6]). Thus in such scenarios a solution which employs both trust and cryptography is advised.

		Exponentiations per M_i	Serial
Cliques-I	IKA	$i+1$ for M_i, n for M_n	$n(n+1)/2$
	Join	1 for M_i, $n+1$ for M_n,M_{n+1}	$2n+1$
	Leave, Partition	1 for M_i, $n-k$ for M_n	$n-k+1$
	Merge	1 for $M_1..M_{n-1}$, $n+1$ for M_n, $n+i+1$ for M_{n+i}	$n(k+1)+k(k+1)/2$
Cliques-II	IKA	3 for M_i, 2 for M_{n-1}, n for M_n	$2n$
	Join	1 for M_i, $n+1$ for M_n,M_{n+1}	$2n+1$
	Leave, Partition	1 for M_i, $n-k$ for M_n	$n-k+1$
	Merge	2 for $M_1..M_{n-1}$, 3 for $M_{n+1}..M_{n+k-1}$, $n+i$ for others	$(n+2k-1)$
TGDH	IKA	NP	NP
	Join,Merge	$O(logn)$	$O(logn)$
	Leave	$O(logn)$	$O(logn)$
	Partition	$O(logn)$	$O(logn)$
STR	IKA	NP	NP
	Join	2	2
	Leave, Partition	For Node i: 1+ Min(d(root,lsn),d(root,i))	Same as per M_i
	Merge	$k+1$	$k+1$

Table 17: Computation Costs

Min: Minimum function
d: distance function
root: root node of the tree
lsn: Lowest numbered sponsor node

5 Conclusion

We have seen that Group Key Agreement protocols provide a good solution to the problem of managing keys in Ad hoc networks, as they provide the ability to generate session keys which adapts well to the dynamic nature of Ad hoc groups. Also by providing features like Key Independence and Perfect forward secrecy, it makes it feasible to secure a group over long periods of time.
We have also seen that group key agreement is not as easy to implement in Ad hoc environments as in conventional scenarios due to some special characteristics that these networks have. Thus one has to meet the security goals but at the same time not forget the computational and communication limitations of the devices. Regarding the GKA protocols, it is easy to see that one single protocol cannot meet best the needs of all kinds of Ad hoc networks. For Ad hoc networks with asymmetric computing and communication devices, protocols like Cliques are well suited which require only some powerful devices to do bulk of the computation. While for symmetric scenarios, Tree-based protocols are well-suited as they distribute the load equally to all nodes.

References

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

[2] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of Crypto '93*, volume 773 of LNCS, 1993.

[3] S. Buchegger and J. Le Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410. IEEE Computer Society, January 2002.

[4] L. Buttyan and J.P. Hubaux. Rational exchange – a formal model based on game theory. In *Proceedings of 2nd International Workshop on Electronic Commerce (WELCOM 2001)*, Heidelberg, Germany, November 2001.

[5] L. Buttyan and J.P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.

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

[7] O. Chevassut E. Bresson and D. Pointcheval. Dynamic group diffie-hellman key exchange under standard assumptions. *In Advances in Cryptology - Proceedings of EUROCRYPT*, 2002.

[8] M. Steiner G. Ateniese and G. Tsudik. New multiparty authentication servi ces and key agreement protocols. *IEEE Journal of Selected Areas in Communications*, 18(4):1–13, 2000.

[9] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[10] Smart Homes. http://www.jrf.org.uk/housingandcare/smarthomes.

[11] G. Tsudik M. Steiner and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, August 2000.

[12] L. Buttyan S. Capkun and J. P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *to appear in IEEE Transactions on Mobile Computing.*

[13] A. Perrig Y. Hu and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *The 8th ACM International Conference on Mobile Computing and Networking*, September 2002.

[14] A. Perrig Y. Kim and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. *ACM CCS*, November 2000.

[15] A. Perrig Y. Kim and G. Tsudik. Communication-efficient group key agreement. *IFIP SEC*, June 2001.

[16] S. Yi and R. Kravets. Key management for heterogeneous ad hoc wireless networks. Technical Report UIUCDCS-R-2002-2290, UIUC, July 2002.

[17] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399