Group Key Agreement Protocols for Dynamic Peer Groups

Nirav Jasapara
jasapara@isi.edu

ABSTRACT

With the increased use of distributed services and applications, secure group communication over unsecured networks (eg. Internet) has become crucial. Key Agreement is a critical part of providing security services for group communication systems. This paper attempts to study the basic infrastructure required for Key Agreement in Dynamic Peer groups (DPG), approaches of some notable distributed key management techniques, their working principles and their performance characteristics with respect to the problem of Key Agreement in DPG.

KEYWORDS

Authentication, dynamic groups, key establishment, Deffie Hellman key exchange, Cryptography.

1. INTRODUCTION

With the increased use of distributed services and applications, secure group communication over unsecured networks (eg. Internet) has become crucial. These applications, which require a reliable group communication infrastructure, include voice and video-conferencing, white-boards, multi-user games, replicated servers, etc. Key management is the base for providing common security services (data secrecy, authentication and integrity) for group communication.

This paper tries to introduce a reader to the basics of Key Agreement in DPG with a focus on performance of some notable protocols proposed for the same. We will first discuss the important terms one comes across while talking about Key Agreement in DPG. We then discuss some issues which makes Key Management challenging in dynamic peer groups. We then talk about classification and different types of Key Management Protocols and typical operations that they support. The protocols we describe are Group Diffie Hellman (GDH)[1], Centralized Key Distribution (CKD), Tree-based Group Diffie Hellman (TGDH)[2], Skinny Tree (STR) [9] and Burmester-Desmedt (BD)[8].

Finally we discuss about the performance characteristics of each protocol w.r.t computation cost of generating a key and communication cost of distributing a key.

The rest of the paper is organized as follows. Section 2 describes the basics of Group Key Management. We briefly describe the working of each protocol in Section 3. Section 4 discusses their performance characteristics and we conclude in Section 5.

1.1. DEFINATIONS

The most important properties for any practical Key Management Protocol includes:

- Key Agreement Protocol
- Perfect Forward Secrecy
- Resistance to known-key attacks
- Key integrity

We now present the definition for the above terms (adapted from [7])

a. A **key agreement protocol** is a key establishment technique whereby a shared secret key is de- rived by two or more speci_ed parties as a function of in- formation contributed by, or associated with, each of these, such that no party can predetermine the resulting value.

b. A protocol offers **perfect forward secrecy (PFS)** if compromise of a long-term key(s) cannot result in the compromise of past session keys.

c. A protocol is said to be vulnerable to **known-key attack** if compromise of session keys allows:
1) a passive adversary to compromise keys of other sessions or
2) an active adversary to impersonate one of the protocol parties.

d. Key Integrity New members cannot derive the key used in the older sessions and old members (no longer a part of the group) cannot derive the current key being used in the group.

Initial Key Agreement(IKA): A kind of group genesis, when the key is agreed upon. It requires contacting every prospective group member. In contributory key agreement, a key share from each member is obtained.

Auxiliary Key Agreement (AKA): Refers to a groups of operations needed to support secure communication in a dynamic group.

2. GROUP KEY MANAGEMENT

Key management for groups has some unique properties which makes key communications protocols for these more challenging. The issues are: 1. Peer groups are small (< 100 members) 2. They have no hierarchy 3. A subset of the group should be able to function as a group itself (in case of network failure).

A comprehensive group key agreement solution must handle all 'Group Membership Events'. These events include membership addition or deletion, disconnection of a member or subgroup, merging of two groups, etc. At each such event it may have to invalidate old keys, generate new keys and distribute them securely. Typical events are:

a. Join: A new member is added to the group.
b. Leave: A existing member leaves the group.
c. Merge: When two groups merge to form a super-group.
d. Partition: A subset of members from the current group are disconnected/partitioned, maybe to form a new group.

It should be noted that every time a new member/group joins or leaves the current group, the old key has to be discarded and a new key is generated and distributed to all the 'current' members of the group. Here the cost of key generation vs cost of key distribution comes into picture.

There are several approaches to Group Key management. One approach uses a single, centralized entity, to generate keys and distribute them to the group. This approach uses a fixed trusted third party (TTP) as the key server. In this case, a key server maintains long-term shared keys with each group member in order to enable secure two-party communication for the actual key distribution. This approach has some problems: 1) the TTP must be constantly available 2) a TTP must exist in every possible subset of a group in order to support continued operation in the event of network partitions 3) It might be simply unacceptable for a single party to generate the group key. For example, each party may need assurance that the resulting group key is fresh and random (e.g., in case the key is later used for computing digital signatures) [7]. The first problem can be addressed with fault- tolerance and replication techniques. The second, however, is difficult to solve in a scalable and efficient manner. However, Amir et. al. points out that that centralized approaches work well in a one-to-many multicast scenario since a TTP placed at, or very near, the source of communication can support continued operation within an arbitrary partition as long as it includes the source.

Another key management approach involves dynamically selecting, in some deterministic manner –a group member charged with the task of generating keys and distributing them to other group members. This approach is robust and more amenable to many-to- many type of group communication since any partition can continue operation by electing a temporary key server. The drawback here is that, as in the TTP case, a key server must establish long-term pairwise secure channels with all current group members in order to distribute group keys [3]. Consequently, each time a new key server comes into play, significant costs must be incurred to set up these channels. Another disadvantage, again as in the TTP case, is the reliance on a single entity to generate good (i.e., cryptographically strong, random) keys.

In contrast to the above, contributory key management asks each group member to contribute an equal share to the common group key (computed as a function of all members' contributions). This approach avoids the problems with the single points of trust and failure. Moreover, some contributory methods do not require the establishment of pairwise secret channels among group members. However, current contributory key agreement1 protocols are not designed to tolerate failures and group membership changes during execution. In particular, nested failures, partitions and other group events are not accommodated. The protocols that we will discuss in this paper all fall under this category.

2.1 Scale

Group based protocols require all parties to share a common key. A Group Key Management (GMK) Protocol is required to manage various aspects of key generation, exchange and update. GMK protocols generally fall into two classes:

- Protocols designed for large-scale applications and relatively weak security requirements.
  The applications here have a one to many communication paradigm (eg. IP Multicasting). One example is the Group Key Management Protocol (GKMP) [5] which provides key dissemination using a dedicated group controller. Some key management approaches targeting IP Multicast use hierarchical key distribution. Another hierarchical approach [6] makes the group key itself hierarchical, usually with a tree-based structure.
- Protocols designed to support tightly-coupled dynamic peer groups and strong security requirements.
  The applications here have a many to many communication paradigm. The scalability requirements are modest (as compared to the Protocol described above). A number of Group Key Management Protocols have been proposed. Almost all of them extend the Diffie-Hellman key exchange [4] method for communication with many peer entities. The protocols that we discuss in this paper all fall under this category.

3. PROTOCOLS

In this section, we briefly discuss five notable distributed key management protocols viz; Group Diffie Hellman (GDH)[1], Centralized Key Distribution (CKD), Tree-based Group Diffie Hellman (TGDH)[2], Skinny Tree

(STR) [9] and Burmester-Desmedt (BD)[8]. All these protocols are derived from the Diffie-Hellman key exchange algorithm [4].

3.1 GDH (derived from [3], [1])

Based on group extensions of the 2-party Diffie-Hellman key exchange [4]. It provides fully contributory authenticated key agreement. GDH is fairly computation-intensive requiring O(n) cryptographic operations upon each key change. Here the shared key is never transmitted (in cleartext or encrypted) over the network. Here the group shared key is generated by each member by using the partial keys of all other members. One member, who is the group controller, distributes this list.

The protocol runs as follows. When a merge event occurs (new member/group joins), the current controller generates a new key token by refreshing its contribution to the group key and passes the token to one of the new members. When the new member receives the token, it adds its own contribution and passes the token to the next new member. Eventually, the token reaches the last new member.

When some of the members leave the group, the controller (who, at all times, is the most recent remaining group member) removes their corresponding partial keys from the list of partial keys, refreshes each partial key in the list and broadcasts the list to the group. Each remaining member can then compute the shared key. Note that if the current controller leaves, the last remaining member becomes the controller of the group.

3.2 CKD (derived from [3])

It uses centralized key distribution with the key server dynamically chosen from among the group members. A key server uses pairwise Diffie-Hellman key exchange to distribute keys.

Whenever group membership changes, the controller generates a new secret and distributes it to every member encrypted under the long-term pair-wise key it shares with that member. In case of a join or merge event, the controller initially establishes a secure channel with each incoming member.

When a partition occurs, in addition to refreshing and distributing the group key, the controller discards the long-term key it shared with each leaving member. A special case is when the controller itself leaves the group. In this case, the oldest remaining member becomes the new controller. Significant additional cost is incurred, since, before distributing the new key, the new controller must first establish a secure channel with every remaining group member.

3.3 TGDH (derived from [2], [3])

It is a tree-based group Diffie-Hellman. The key tree is organized as follows: each node is associated with a key K_v and a corresponding blinded key BK_v derived from the key. The root key represents the group key shared by all members, and a leaf key represents the random contribution by of a group member. Each internal node has an associated secret key and a public blinded key. The secret key is the result of a Diffie–Hellman key agreement between the node's two children. Every member knows all keys on the path from its leaf node to the root as well as all blinded keys of the entire key tree. The protocol relies on the fact that every member can compute a group key if it knows all blinded keys in the key tree.

Following every group membership change, each member independently and unambiguously modifies its view of the key tree. Depending on the type of the event, it adds or removes tree nodes related to the event, and invalidates all keys and blinded keys related with the affected nodes (always including the root node). As a result, some nodes may not be able to compute the root key by themselves. However, the protocol guarantees that at least one member can compute at least one new key corresponding to either an internal node or to the root. Every such member (called a sponsor) computes all keys and blinded keys as far up the tree as possible and then broadcasts its key tree (only blinded keys) to the group. If a sponsor cannot compute the root key, the protocol guarantees the existence of at least one member which can proceed further up the tree, and so on. After at most two rounds (in case of a merge) or log(n) rounds (in case of a worst-case partition), the protocol terminates with all members computing the same new group (root) key.

After a partition, the protocol operates as follows. First, each remaining member updates its view of the tree by deleting all leaf nodes associated with the partitioned members and (recursively) their respective parent nodes. To prevent reuse of old group keys, one of the remaining members (the shallowest rightmost sponsor) changes its key share. Each sponsor computes all keys and blinded keys as far up the tree as possible and then broadcasts its view of the key tree with the new blinded keys. Upon receiving the broadcast, each member checks whether the message contains a new blinded key. This procedure iterates until all members obtain the new group key.

3.4 STR (derived from [9], [3])

It is an "extreme" version of TGDH with the underlying key tree completely unbalanced or stretched out. In other words, the height of the key tree is always ($n - 1$), as opposed to (roughly) log(n) in TGDH. All other features of the key tree are the same as in TGDH.

After a partition, the sponsor is defined as the member corresponding to the leaf node just below the lowest leaving member. After deleting all leaving nodes, the sponsor refreshes its key share, computes all (key, blinded key) pairs up to the level just below the root node. Finally, the sponsor broadcasts the updated key tree thus allowing each member to compute the new group key.

STR merge runs in two rounds. In the first round, each sponsor (topmost leaf node in each of the two merging tree) first refreshes its key share and computes the new root key and root blinded key. Then, the sponsors exchange their respective key tree views containing all blinded keys. The topmost leaf of the larger tree becomes the sole sponsor in the second round in the protocol. Using the blinded keys from the key tree it received in the first round, the sponsor computes every (key, blinded key) pair up to the level just below the root node. It then broadcasts the new key tree to the entire group. All members now have the complete set of blinded keys which allows them to compute the new group key.

3.5 BD (derived from [8], [3])

It is a protocol based on Burmester-Desmedt[8] variation of group Diffie-Hellman. Unlike other protocols discussed thus far, the BD protocol [Burmester and Desmedt 1994] is stateless. Therefore, the same key management protocol is performed regardless of the type of group membership change. Furthermore, BD is completely decentralized and has no sponsors, controllers, or any other members charged with any special duties.

The main idea in BD is to distribute the computation among members, such that each member performs only three exponentiations. This is performed in two communication rounds, each consisting of *n* broadcasts. More details can be found in [8].

4. PERFORMANCE EVALUATION

In this section we will discuss about various costs associated with Key Agreement Protocols, their meaning, importance and analyze them w.r.t. the five protocols discussed above.

Four types of events can lead to a change in group membership, viz; Member join, leave and group merge, partition. Every time a change occurs to group membership we have to;

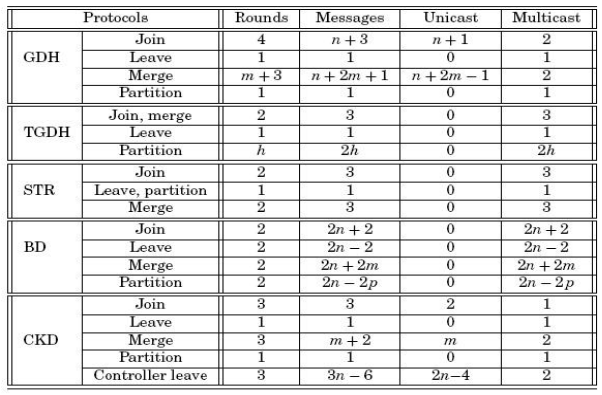
- Generate a new key
- Distribute the new key

Because group memberships may change frequently, the two important parameters which can impact overall performance are;

- Time required to generate a new key (Computation Cost)
- Time required to distribute the new key (Communication Cost)

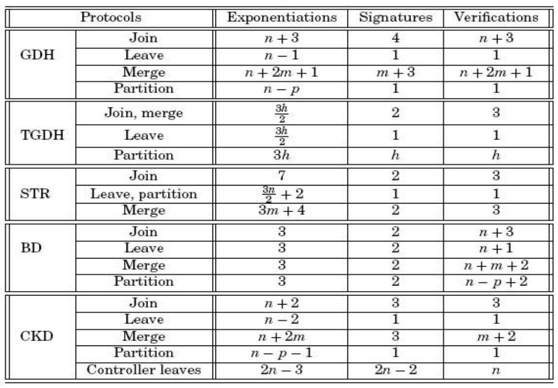
Now, in high-speed LAN's the communications costs may be negligible but becomes very impornat in high-delay environments. The tables I and II [3] below summarize the two types of costs for the five protocols we discusses.

Table I. Communication Cost



Protocols		Rounds	Messages	Unicast	Multicast
GDH	Join	4	$n+3$	$n+1$	2
	Leave	1	1	0	1
	Merge	$m+3$	$n+2m+1$	$n+2m-1$	2
	Partition	1	1	0	1
TGDH	Join, merge	2	3	0	3
	Leave	1	1	0	1
	Partition	h	$2h$	0	$2h$
STR	Join	2	3	0	3
	Leave, partition	1	1	0	1
	Merge	2	3	0	3
BD	Join	2	$2n+2$	0	$2n+2$
	Leave	2	$2n-2$	0	$2n-2$
	Merge	2	$2n+2m$	0	$2n+2m$
	Partition	2	$2n-2p$	0	$2n-2p$
CKD	Join	3	3	2	1
	Leave	1	1	0	1
	Merge	3	$m+2$	m	2
	Partition	1	1	0	1
	Controller leave	3	$3n-6$	$2n-4$	2

Table II. Computation Cost



Protocols		Exponentiations	Signatures	Verifications
GDH	Join	$n+3$	4	$n+3$
	Leave	$n-1$	1	1
	Merge	$n+2m+1$	$m+3$	$n+2m+1$
	Partition	$n-p$	1	1
TGDH	Join, merge	$\frac{3h}{2}$	2	3
	Leave	$\frac{3h}{2}$	1	1
	Partition	$3h$	h	h
STR	Join	7	2	3
	Leave, partition	$\frac{3n}{2}+2$	1	1
	Merge	$3m+4$	2	3
BD	Join	3	2	$n+3$
	Leave	3	2	$n+1$
	Merge	3	2	$n+m+2$
	Partition	3	2	$n-p+2$
CKD	Join	$n+2$	3	3
	Leave	$n-2$	1	1
	Merge	$n+2m$	3	$m+2$
	Partition	$n-p-1$	1	1
	Controller leaves	$2n-3$	$2n-2$	n

The numbers of current group members, merging members, and leaving members are denoted as n,m, and p, respectively. The height of the key tree constructed by the TGDH protocol is denoted by h.

4.1 IMPORTANT RESULTS

Live tests were conducted in [3] using the five protocols mentioned above over LAN's and WAN's. The important inferences that can be drawn are as follows: [3]

a. communication cost for group-oriented cryptographic protocols over long delay network can dominate the computational cost.

b. simultaneous *n* broadcast message for relatively large *n* is also very expensive in practice, and it, therefore, is recommended to be avoided

c. Cost of BD: The cost of BD roughly doubles as the group size grows in increment of the total number of machines

d. Cost of TGDH: its worst case communication cost is quite expensive compared to STR. The results show that TGDH is the best overall protocol in practice, if only one protocol has to be selected.

In short, GDH is fairly computation-intensive requiring O(n) cryptographic operations upon each key change. It is, however, bandwidth-efficient. CKD is comparable to GDH in terms of both computation and bandwidth costs. TGDH is more efficient than the above in terms of computation as most operations require O(log n) cryptographic operations. BD is computation-efficient requiring constant number of exponentiations upon any key change. However, communication costs are significant with two rounds of n-to-n broadcasts.

4.2 APPLICATIONS CLASSES

Y. Amir et al. discusses major application classes and the most suited group key agreemtent protocol for each.

Application	Characteristics	Best Suited Protocol
Conferencing	Most member will join and leave at the same time. Single member operations.	BD: small n STR: large n
One to many broadcast	One sender, many receiver. Receivers join and leave at will	GDH CKD: strong security
Distributed Logging	Several logging server get updates from many senders which may join or leave the group frequently. Requires efficiency.	TGDH
Mobile State Transfer	Few participants that share soft state. May join frequently and leave temperarorily.	TGDH

The above applications represent a broader class of applications with common characteristics. Using the overlapping characteristics of a particular application we can use the above table as a ball-mark reference to quickly decide on which protocol would work best for our application.

5. CONCLUSION

GKM is particularly challenging because of its added requirement to support group membership dynamics and at the same time provide secure communication to all its members. A number of GKM protocols supporting abstract peer groups have been developed in the last decade. Most of them extend the well-known Diffie-Hellman key exchange [3] method to group of n parties. These protocols vary in degrees of protection and in their performance characteristics.

Understanding the basic framework required for GKM, gives a good insight into this problem. The cost of group key management is determined by two dominating factors: communication and computation. Typically, efficiency in one comes at the expense of the other. Protocols that distribute computation usually require more communication rounds, while protocols minimizing communication require more computational effort. From this study, it can be seen that right now there is no clear winner i.e. a universally acceptable solution is not yet found. Every Application has its own set of characteristics (group dynamics) which arbitrates the use of a particular GKM protocol. Ongoing research based on different protocols as directed by several researchers can help make clear the right direction for solving this problem. Furthermore, learning from them, one can base future GKM protocols to live with the inherent complexities of group dynamics and yet achieve an acceptable level of security.

ACKNOWLEDGEMENTS

I would like to thank Dr. Debanik Roy who made numerous useful suggestions throughout the preparation of this paper.

REFERENCES

[1]. M. Steiner, G. Tsudik and M. Waidner. Key Agreement in Dynamic Peer Groups. IEEE Transactions on Parallel and Distributed Systems, August 2000.
[2] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In ACM CCS 2000, November 2000
[3]. Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik. On the Performance of Group Key Agreement Protocols. IEEE ICDCS'2002, July 2002.
[4]. W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. 22, pp. 644–654, Nov. 1976.
[5]. H. Harney and C. Muckenhirn, "Group key management protocol (gkmp) specification," Tech. Rep. RFC 2093, IETF, July 1997.
[6]. C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure group communications using key graphs," in P*roceedings of the ACM SIGCOMM '98*, pp. 68–79, 1998.
[7]. Giuseppe Ateniese, Michael Steiner, and Gene Tsudik, "New Multiparty Authentication Services and Key Agreement Protocols", IEEE Journal of Selected Areas in Communications, VOL 18, NO. 4, April 2000
[8]. M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," *Advances in Cryptology – EUROCRYPT'94*,May 1994.
[9]. Yongdae Kim, Adrian Perrig, Gene Tsudik: Group Key Agreement Efficient in Communication. IEEE Trans. Computers 53(7): 905-921 (2004)