Survey on the Hierarchical Key Distribution Algorithms for Secure Multicasting

Xu Yu

ABSTRACT

Security is always a challenging task due to the dynamic nature of multicasting model as compraing to that of the unicasting network. In the passed two or three decades, cryptography has become the well-established means to solve the security problems in networking. However, there are still a lot of difficulties for directly deploying cryptography algorithms into multicasting environment as what has been done for unicasting environment. Among all of the issues, *key distribution* is regarded as one of the most difficult problems so far. This paper addresses issues relevant to implementing key distribution algorithm for multicasting networks. More specifically, a survey on one particular family of architectures — the *hierarchical* key distribution algorithm(or, *scheme*), which is regarded as the most efficient category of key distribution architectures in term of efficiency and scalability is provided. We try to classify the current implementations, use specific algorithm(s) as the representative for each category. Based on a common criteria, those implementations are evaluated and compared to determine the strengths and weakness.

Keywords

Secure Multicast, Key Distribution, Rekey, Hierarchical Rekey

1. INTRODUCTION

Multicast communication is an efficient means of distributing data to a group of participants. Different from traditional unicast communications, multicast allows single IP datagram to be routed to multiple entities at the same time. Within a multicast group, the membership is dynamic — entities can join or leave at anytime. The inherent benefits of multicast routing also present huge challenges in term of security control, as the membership can change at any time.

The development in cryptography in the passed two or three decades has greatly aided the network communication in providing required protections. Through the use of encryptions and digital signatures, one can achieve desired levels of security requirements, such as *confidentiality*, *integrity* , and *authentication*, *etc.* This has been witnessed by the vast references of applying cryptography in networking domain, such as the *IPSec* protocol, or *TLS/SSL* protocols. Those ideas can also be applied to multicast communication. A generic outline for multicast registration and secure key distribution can be as following:

Assumption There is a trusted server which is used to store membership information (which will be used to exercise group access control or other more granular quality of protection), and also, there would be some trusted servers that are used to generate keys[1].

Step 1 When a client wants to join a particular group, the client would perform mutual authentication with server using certain authentication protocol;

Step 2 After the client has been authenticated and accepted into the group, server and client will share a common secret that will be denoted as the client's *individual key.*

Step 3 At the same time, the server distributes a *group key* to the client, which is shared by all members of the particular group[2].

Step 4 For the group communication, all the traffic would be encrypted with the *group key.* Therefore, only members of this particular group will be able to decrypt or encrypt the message.

Step 5 For any changes in the group dynamics (*i.e.*, any join/leave operation in the group), the server will generate a new key, update the group and distribute the new key to all the current members.

One of the most important aspect of securing multicast communications using cryptographical algorithms, or rather one of the most important aspect in securing any communication models using cryptography is to distribute the key information to *all*, and *only* the allowed recipients. However, mutlicast communication model introduces additional challenges.

[1]for the ease of discussion, we combine those two roles into one entity. While in the reality they can be assigned to different entities.

[2]This particular group could be the entire domain or subset of the domain

The most significant difference between unicast communication and multicast communication is the dynamic nature of multicast.

Generally speaking, key distribution includes these aspects:

- Initialization of the distribution domain.
- Joining and leaving semantics.
  This gives the so-called *handshake* protocol between the joining/ leaving entity and the server. In other words, this particular entity needs to send request to server, and server will validate the identity or the entity as well as the request, then perform registration/ de-registration.
- Refreshing of the distribution[3].
  It is sometimes being called as *re-keying* operation. Because of the change in group dynamics, certain keys will be obsoleted. Server needs regenerate keys and re-distribute them to all the current members.

Key distribution is required in secure multicast to ensure that only the *current* group members can send encrypted multicast datagram, and decrypt the received multicast datagram. In other words, the key distribution algorithm must ensure that an entity is only allowed to participate during those periods when it is authorized to do so. It means the securing multicast communication must provide the following two functionalities [8]:

Forward Message Securecy An entity should not be allowed to read multicast communication after it leaves the multicast group.

Backward Message Securecy An entity should not be allowed to read multicast communication messages exchanged prior to the time when it joins the multicast group.

Kruus performed a survey on key distribution architectures for multicast in [3]. In his pioneer work, five different key distributions have been described and compared. They are 1. Manual Key Distribution, 2. Pairwise Keying, 3. Hierarchical Trees, 4. Secure Lock, and 5. Distributed Registration and Key Distribution. As notes in his paper, the *Hierarchical-Trees* methods provide linear initial keying performance and improved logarithmic rekey performance. The solution is the most scalable and efficient one because of the logarithmic performance.

In the recent years, a lot of hierarchical key distribution schemes have been proposed. Each of them tries to deploy the hierarchy into multicast diagram, though, each of them tries from different aspect. In this paper, we try to provide a comparison among those proposals. Based on a common simple criteria, different hierarchical key distribution algorithms are evaluated and compared to determine the strengths and weakness.

[3]For the first two aspects, the approaches used in unicast networking can also be applied easily in multicast. Therefore, the major work for secure multicast key distribution lies in this aspect. For all of the remaining discussion, we would focus on this area.

2. HIERARCHICAL KEY DISTRIBUTION SCHEMES

2.1 Criteria for evaluating different key distribution algorithm

Efficiency and scalability are the two fundamental criteria for designing a "good" key distribution algorithm for securing multicast communication. In Kruus' pioneer work [3], he uses 5 criteria as the baseline for comparing the performance of different key distribution scheme, as shown in Table 1:

However, one important issue has been missed out from his discussion, which we think are particular important in a scalable multicast system:

Stateness Whether the solution is stateful or stateless.
As shown later when we describe each algorithm, during the new key distribution process, certain algorithms would make use of the previous group key to encrypt the new generated group key. Therefore , the dependency of state is required for these schemes. In the case the group key for certain state is lost it is not possible for the participant to re-catch the session by all means.

In the following sections, we briefly describe several key distribution algorithms proposed so far [4, 9, 2, 6, 5], each of those represents a family of algorithms. After that, we will discuss the efficiency and the scalability issues of them.

A set of notations are introduced here, which will be used during following sections for describing the operations:

- K_{GRP} is used to denote the shared key for the whole group;
- $[\{msg\}_{Key}]$ denotes a message being encrypted using the key.

2.2 Hierarchical Key Distribution Algorithms

2.2.1 Group-based Hierarchy

Mittra *et al* propse a hierarchical keying scheme in [4], namely *Iolus*. This is one of the earliest efforts to solve the problem of securing multicast with hierarchical tree. In this scheme, all the entities participating the secure multicast session have been divided into different subgroups. Two types of new entities are introduced to manage and connect the various subgroups — *group security controller*(GSCs), which manages the top-level subgroup, and *group security intermediaries*(GSIs), which managed each of the subgroups — they are both called *group security agents*(GSAs). They form the bridges between subgroups by receiving datagram multicasted within their parent or child subgroups and re-multicasting to their child or parent subgroups respectively.

Criteria		Description
Efficiency	Initial Keying	The efficiency of the initial key distribution exchange at the start of the session
	Re-keying	The efficiency of re-key operations
Scalability		Whether the solution is scalable
Other Considerations	Computation Requirement	Levels of computational resources are required by both the key distributor and members to process
	Storage Requirement	The amount of storage required by participants for key storage key messages.

Table 1: Criteria for Key Distribution Scheme

In the *joining* operation, a sender or receiver locates its designated GSA and sends a `JOIN` request to it through a *secure unicast channel*. The GSA would create a secret $K_{GSA\text{-}MBR}$ to be shared only with the new member . After that, GSA would make a new group key $K^{'}_{GRP}$, multicast it encrypted with the old key K_{GRP} (*i.e.* $[\{K^{'}_{GRP}\}_{K_{GRP}}]$) to all the old group members, and send the new key to the joining party via the separated secure channel.

In the case of *leaving* operation, GSA would also create a new group key $K^{'}_{GRP}$ and multicast one `LEAVE` message to the group with following format:

$$[\{K^{'}_{GRP}\}_{K_{GSA\text{-}MBR_1}}, \quad \{K^{'}_{GRP}\}_{K_{GSA\text{-}MBR_2}}, \quad \ldots, \quad \ldots, \quad \{K^{'}_{GRP}\}_{K_{GSA\text{-}MBR_n}}]$$

.

During the data transmission phase, instead of senders muliticasting directly to the group, each sender unicasts the data to the GSA encrypted with its unique key shared with the GSA, $K_{GSA\text{-}MBR}$. The GSA will be responsible for decrypting the data, re-encrypt it with K_{GRP}, and then multicast it to the group as well as to its parent subgroup (if any). In this way, it will eliminate the possibility of using an outdated group key.

Iolus is a typical *group-based* hierarchical key-distribution algorithm. The re-key operation only takes effect in a subgroup scope instead of changing the whole global group. At the same time, by using specific defined `LEAVE` message, the number of messages required to rekey the sub-group (or subtree) is only one – with the length of $O(n)$. For the storage requirement, all the leaf entities in the graph need to store two keys — one for individual, and one for group; while the intermediate CSIs need to store three — one for individual, one for the subgroup it controls, and one for the group it belongs to. Therefore, Iolus is quite efficient in terms of key distribution and storage requirements. Also, it is straight forward to map the real-life infrastructure to group-based hierarchical tree. For example, one can easily make use of different departments in one organization as the sub-group, or based on geographical region, etc.

On the other hand, the scheme does introduce additional computations at the inter-domain GSA, as for each GSA transmitting multicast message from one domain to another, a decryption (using the key for the incoming domain) and an encryption (using the key for the outgoing domain) are required. Also, the algorithm is state-sensitive or stateful as the new key will be encrypted by the old key.

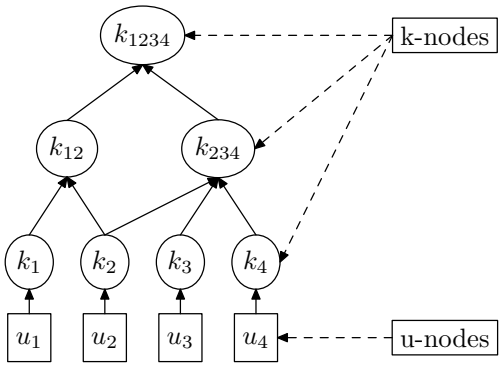

Figure 1: A Key Graph in LKH

2.2.2 Key-based Hierarchy

C.K. Wong, *et al* proposed *logical key hierarchy*(LKH) in [9], which is a typical *key-based* hierarchical system. In LKH, the tree is based on the keys instead of the entities — a key graph is built to represent the *user-key* relationship.

Key graph(as shown in Figure 1, cited from [9].) is a directed acyclic group with two types of nodes: *u-nodes*, which represents users, and *k-nodes*, which represents keys. U-nodes stay at the bottom of the group, each has one or more outgoing edges but no incoming else; while the k-nodes stay in the intermediate and the top of the group, each has one or more incoming edges. The hierarchy of the keys are composed by the round nodes in the key group. In the key hierarchy, the group key is the root, individual keys are leaves, and subgroup keys stay in the middle.

When an entity is trying to join/leave a secure group, the key server performs the authentication, generates the individual key for the entity, and finds an existing k-node in the key tree to attach (in the case of joining, and the node is called as *joining point*) or detach (in the case of leaving, and the the node is named as *leaving point*) the entity into the tree. Based on this principal, for the structure change operations, the affected nodes will be the path from the joining/leaving point along the tree, and up to the root.

Following the hierarchical key distribution approach, LKH also attempts to change a $O(n)$ problem into a $O(\log n)$ problem, in which n gives the group size. In fact, for join/leave operations, LKH costs $O(\log n)$ in the server side, and $O(1)$ at the client side. For the storage cost, the server needs to store all the keys in the graph, which is approximate to $2n$; while each client needs to store all the keys from its joining point up to the root, which is actually the height of the key tree and costs $O(\log n)$. However, in contrast to Iolus, when broadcasting the message to the group, only one pair of encryption/decryption operations is required, as all the members share the global group key. The proposed algorithm for re-key in LKH is also stateful, as the new key will be encrypted with the old one before broadcasting.

2.2.3 Periodical Group Rekey

Setia, *et al* study the dynamic orientation of the multicasting, and its effect on the efficiency of the key distribution. It has been found out that for a group with large number of participants and significant dynamic orientation (*i.e.*, the rate at which members leave and join the group is very high), re-keying on each membership change will becomes the primary bottleneck for scalability. Therefore, Setia proposes a new schema based on the idea of *periodic* re-keying, namely, *Kronos*[6].

The operation of the Kronos protocol is similar to that of IGKMP [1] and [7]. The whole domain is divided into different "area", and there is a domain-wide key distributor (DKD) and an area key distributor (AKD) corresponding to each area. The major differences between Kronos and that of IGKMP lie in two aspects: 1. the DKD is not directly involved in generating the new group key, instead, each AKD independent generates the same group-wide key and transmits it to the members in its area; 2. all the AKDs *periodically* update the key instead of updating based on membership variations — this is the most important contribution of Kronos.

Setia does a lot of simulations, and compares the results. It has been found that comparing with *Iolus*, Kronos gives higher join/leave latencies, but lower data packet delivery latencies. At the same, Kronos shows some attractive properties for real-time application:

- the data latencies are independent of the location of the AKDs, and
- the group re-keying rate is independent of group size and membership dynamics.

These properties give Kronos great scalability as the overhead of re-keying is really predictable and bounded.

However, since the key distribution operations in Kronos is highly synchronized, additional resources support (such as Network Time Protocol for normal situation, and Coordinated Universal Time via Global Positioning Server receiver for highly demanding network) is vital for the protocol.

The original Kronos protocol as defined in the paper is also a *stateful* scheme – the new group key derived from the old key:

$$K_{i+1} = E_{k_{common}}(K_i), i \geq 0$$

, in which K_{common} is the common key for the whole network, and E_k is the encryption algorithm used to generate key. However, since this protocol makes use of the timing factor that is independent of the group, it is possible to make the protocol *stateful* with a key generation algorithm that only depends on time. In other words, $K_{i+1} = E_{k_{common}}(T_{Current\ Time})$.

2.2.4 Hybrid Hierarchies and Batch Re-keying

Mykil, or **M**ulti-Hierarch**y** **B**ased **K**ey Di**s**tribution Protoco**l** proposed by J.H. Huang combines both hierarchical key (as in Iolus and LKH) and batch re-keying (as in Kronos) [2].

Mykil tries to combine both *group-based* and *key-based* hierarchical schemes — it uses group-based hierarchy to divide a multicast group into subgroups (which is called *area*) with designated area controller (AC); at the same time, Mykil follows key-based hierarchical key distribution to build a tree-structured hierarchy of keys called *auxiliary-key tree* in each area to facilitate key distribution to the area members. Therefore, the whole structure of Mykil is a hybrid hierarchical system with *Iolus*-like tree as the higher level, and *LKH*-like tree for each subgroup of Iolus graph. Since the join/leave operations in Iolus are always scoped within subgroup, the operations are exactly the same in LKH with the two modifications:

1. the auxiliary key tree for each area is maintained to be a balanced tree with up to 4 leaves for each node (except root node) — this is based on the observation from LKH that the optimal key tree degree is 4;
2. when an entity leaves the group the leaf for the entity is not pruned , but only all keys along the path from this node to the root are changed — thus the operational cost for maintaining a balance tree is reduced.

The batch idea is used by Mykil in two ways: 1. all the join/leave requests arrive at area controller will be aggregated, and the rekey operation will be postponed until a multicast data packet is received by the area controller; 2. there is also a clock to counter the elapsed time since last rekey operation, and when the interval has reached to a particular value, a rekey operation is forced.

Mykil is a combination of hybrid hierarchical schemes and batch rekeying scheme. As for the hierarchical structure, it is a compromise of group-based hierarchy and key-based hierarchy. By adopting batch rekey, it also reduces the overhead cost in large groups with frequent membership changes. In fact, the robust nature of the scheme makes it particular useful in mobile computing environment.

2.2.5 Stateless Key Distribution

In a stateless key distribution protocol there is no dependency between the key used in different rounds of key distribution operations, *i.e.*, for any legitimate users, the current group key can be deduced from the information broadcasted

in the current key distribution operation. One of such protocols is Subset difference rekey method (SDR) proposed by D. Naor in [5]. SDR uses key-based hierarchy to partition users into different domains. Following standard key-based tree, all the leaves denote the users, while the vertices represent the keys. SDR defines a framework for algorithms called *Subset-Cover*, and the principal is to divide the whole entity domain into two subset — *current* and *revoked*. The basic concepts in SDR are:

Definition 1. Let $\aleph$ be the set of all users, $|\aleph| = N$ and $\Re \subset \aleph$ be the set of revoked users with $|\Re| = r$.

Definition 2. Let w be the number of nodes (leaves and vertices), and V_i, in which $1 \leq i \leq w$ be a node in the tree, assume V_i is an ancestor of V_j, then a *subset* S_{ij} is the set of users in the subtree rooted at node V_i minus the set of users in the subtree rooted at node V_j.

Definition 3. Subset Cover is a collection of disjoint subsets S_{i_1,j_1}, S_{i_2,j_2} , ..., S_{i_m,j_m} which partitions $\aleph \setminus \Re$.

Therefore, we have following relationship:$\aleph \setminus \Re = \bigcup S_{ij}$ for any i,j such that V_i is an ancestor of V_j. At the same time, each subset S_{ij} is associated with a unique key only known by the users in S_{ij}, which is referred to as L_{ij}.

For each joining operation, the key server assigns to the entity the *individual key* as well as a number of *group keys* for all the subsets it might belong to at this time. On the other hand, when the keys in the tree needs to change according to some leaving requests, the key server would calculate the *subset cover* at this moment, re-calculate the new key $K^{'}$, and broadcast following message to the affect subtree:

$$< [\{K^{'}\}_{L_{i_1,j_1}}, \{K^{'}\}_{L_{i_2,j_2}}, \ldots, \{K^{'}\}_{L_{i_m,j_m}}], \{M\}_{K'} >$$

, in which M is the message to be broadcasted, and (i_k, j_k) holds for all V_{i_k} is an ancestor of V_{j_k} and S_{ij} are the minimal number of subsets get affected. Since the new key is encrypted with the keys from the *subset cover*, which none of the revoked entities has. On the other hand, all the entities that are belonging to S_{ij} have the corresponding key. As shown in the description, because each entity is provided with the keys for all the subsets it might belong to at the time it joins the group, the key encryption keys used in each rekey operation are independent of each other, which leads to the stateless nature of the protocol.

Naor *et al* prove that the average number of subsets in the *subset cover* is approximate to $1.25r$ in which r represents the number of revoked entities in $\Re$ [5]. Therefore, the communication complexity, or the number of subsets is independent of the group size. On the other hand, the storage requirement for each entity is pretty high, but still be bound by the fact of $0.5 \log^2 |N|$.

3. SUMMARY OF KEY DISTRIBUTION ALGORITHMS

Each of the solutions presented in [4, 9, 2, 6, 5] tries to efficiently solve multicast key distribution problem in a slightly different fashion. The common factor for all the algorithms are the use of hierarchical tree structure to divide the problem domain, which leads to the logarithmic performance.

Group-based hierarchy organize a multicast group into a hierarchy of subgroup. By delegating the key management services to subgroup controller, the scheme thereby achieves decentralization and scalability. The scheme can tolerate network partition and the actual network infrastructure can be easily mapped with this scheme. Storage requirement in a group-based hierarchy is very low (two or three keys). However, this scheme relies on subgroup controller to send control messages to all subgroup members, which results in a possible performance and scalability bottleneck on the subgroup controller. At the same time subgroup controllers also act as the bridge for crossing domains, and for each message broadcasted, it is required to be decrypted and encrypted again at the subgroup controller.

Key-based hierarchy address the scalability by organizing a hierarchy of cryptographic keys, and scalability is achieved by reducing the number of messages exchanged during a re-key operation. It reduces the computation cost for securely mutlicasting datagram — as every member has the group key, only one pair of encryption/decryption operations is required. On the other hand, it requires servers and participants storing additional information, thus introduces greater storage costs.

Periodical re-key schemes are trying to achieve scalability and efficiency from another angle. It is really orthogonal to the hierarchical architectures. However, as seen in earlier session, when it is combined with hierarchical key distribution algorithm, a greater performance and scalability can be achieved.

Hybrid schemes, such as Mykil tries to combines the advantages of both group-based hierarchy (for subdividing the global user space) and key-based hierarchy (for the efficiencies in multicasting messages within the domain). At the same time the periodical scheme is used to further reduces the overhead introduced by the highly dynamic orientation of the network. All of these features make it well suitable for mobile network.

Stateless hierarchical algorithms, such as SDR described previously fix the state-dependent problem in stateful counterparts. Because of the catastrophic effects of losing state information in stateful schemes, as well as the connectionless nature of the multicasting network, stateless schemes provide additional flexibility and scalability. However, it is at the cost of significant storage requirements.

One thing needs to be highlighted is that "best" solution is really application dependent — the best solution for one particular application or one particular problem domain might not be well suited for another. At the same time, all the solutions proposed so far have close relationship with the cryptography as per current state of arts. As mentioned in the introduction, the development in cryptography will significant influence the solutions in secure multicast domain.

4. CONCLUSION

We have presented a review on some of the proposed hierarchical key distribution algorithms for secure multicasting. By using a set of criteria, this paper evaluated and compared the strengths and weakness. Similar with other areas, there is no single "best" solution to cover all the issues. *Solution* is really application or domain dependent. In general, it has been observed that by combining some different algorithms, defining the scopes or use cases for each and applying each in corresponding scenario, the overall efficiency of the keying scheme can be improved. Therefore, future work should focus on achieving a truly integrated security solution that functions together with the existing multiast protocol.

5. REFERENCES

[1] T. Hardjono, B. Cain, and I. Monga. Intra-domain group key management protocol. *Internet-Draft, draft-ietf-ipsec-intragjm-00.txt*, November 1998.

[2] J.-H. Huang and S. Mishra. Mykil: A highly scalable key distribution protocol for large group multicast. In *IEEE 2003 Global Communicatons Conference (GLOBECOM 2003)*, 2003. To appear.

[3] P. S. Krus. A survey of multicast security issues and architectures. In *21st National Information Systems Security Conference, Arlington, VA*, October 1998.

[4] S. Mittra. Iolus: A framework for scalable secure multicasting. In *ACM SIGCOMM'97*, pages 277–288, 1997.

[5] D. Naor, M. Naor, and J. Lotspiech. *Revocation and Tracing Schemes for Stateless Receivers*, pages 41–62. Advances in Cryptology – CRYPTO 2001. Springer-Verlag Inc., 2001.

[6] S. Setia, S. Koussih, and S. Jajodia. Kronos: A scalable group re-keying approach for secure multicast. *IEEE Symposium on Security and Privacy*, 2002.

[7] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architecture. *Internet-Draft, draft-waqllner-key-arch-00.txt*, 1997.

[8] D. M. Wallner, E. Harder, and R. C. Agee. Key management for multicast: Issues and architecture. *RFC 2627, IETF*, June 1998.

[9] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. In *ACM SIGCOMM'98*, 1998.