Multicast Key Distribution

***B. Ramesh Kumar**
****D.Madhavi**

*****Gokul Institute of Technology and Sciences, Piridi, Andhra Pradesh, India.
******Gokul Institute of Technology and Sciences, Piridi, Andhra Pradesh, India.

Abstract

Multicast key distribution is an important problem for secure group communications. In many applications, multicast is an efficient means of distributing data in terms of resources usage. The privacy of a multicast communication session is usually ensured using symmetric encryption. All the designated receivers or members in a multicast group share a session encryption key. In many applications, the multicast group membership changes dynamically. Thus, session keys shall change dynamically to ensure both *forward secrecy* and *backward secrecy* of multicast sessions. Each session thus needs a new key that is only known to the current session members i.e., session keys need to be dynamically distributed to authorized session members.

Keywords: Key distribution, multicast, MDS codes, erasure decoding, computation complicity.

I. Introduction

IN many applications, multicast is an efficient means of distributing data in terms of resources (such as network bandwidth, server computation, and I/O load) usage. The privacy of a multicast communication session is usually ensured using (symmetric) encryption. All the designated receivers or members in a multicast group share a session (encryption) key. In many applications, however, the multicast group membership changes dynamically, i.e., some new members are authorized to join a new multicast session, whereas some old members should be excluded. Thus, session keys shall change dynamically to ensure both forward secrecy and backward secrecy of multicast sessions. The forward secrecy is maintained if an old member who has been excluded from the current and future sessions cannot access the communication of the current and future sessions, and the backward secrecy is guaranteed if a new member of the current session cannot recover the communication data of past sessions. Each session thus needs a new key that is only known to the current session members, i.e., session keys need to be dynamically distributed to authorized session members. In this paper, we study how a multicast group key can efficiently be distributed in computation. We adopt a common model where session keys are issued and distributed by a central group controller (GC), as it has much less communication complexity, as compared to distributed key exchange protocols, which is a very desired property in most wireless applications [35], [36], [37], [22], [38], [39]. The resources needed for the GC to distribute session keys to group members include communication, storage, and computation resources. The communication complexity is usually measured by the number of data bits that need to be transmitted from the GC to group members to convey information of session keys, whereas the storage complexity is measured by the number of data bits that the GC and group members need to store to obtain session keys. Another similarly important but usually under noticed, if not ignored,

factor is the computation complexity, which can be measured by the number of computation operations (or the computation time on a given computing platform)

That the GC and group members need to distribute and extract session keys. Hereafter, the problem of how resources can effectively be used to distribute session keys is referred to as the group key distribution problem. The group key distribution problem has been studied extensively in the larger context of key management for secure group communications [26], [27], mainly on balancing the storage complexity and the communication complexity. There are two trivial schemes for distributing a session key to a group of n members. The first one is that the GC shares an individual key with each group member, which can be used to encrypt a new group session key. In this scheme, the communication complexity is one, whereas the GC needs to store on key information, each member stores Own key information, and on encryption and decryption operations are needed. In the second scheme, the GC shares an individual key with each subset of the group, which can then be used to multicast a session key to a designated subset of group members. Now, both the communication complexity and the computation complexity reduce to Oð1Þ, but at the cost of increasing the storage complexity to Oð2n Þ for both the GC and each group member. It is easy to see that neither scheme works for practical applications with a reasonable group size n. Thus, research efforts have been made to achieve low communication and storage complexity for group key distribution. Static secret sharing via broadcast channel was studied in [32] and [20]. However, this threshold-based scheme can only distribute a session key to a designated group of members for one-time use. Once a session key is distributed to the group, any member can calculate the secret information that other members in the same group hold. Thus, the scheme does not provide forward or backward secrecy. A secure lock method based on the Chinese Remainder Theorem was proposed in [11]. However, its prohibitively high communication complexity and computation complexity make it only practical for a very small group with limited number of members. Various theoretical measures and schemes for group key distribution were introduced in [14]. Along the same line, many research efforts have been made on balancing communication complexity and storage complexity of the group key distribution problems, for example, [8], [18], [6], [7], [35], and [36]. For a multicast group with a large number of members, key-tree-based schemes were introduced to decompose a large group into multiple layers of subgroups with smaller sizes 37], [22], [38], [39]. Using these schemes, a group membership change can be effectively handled in the corresponding subgroups without affecting other ones. Thus, the communication complexity is reduced, but at the cost of increase in storage and computation complexity together with extra communication delays. For a group of n members, key-tree based schemes have a communication complexity of OðlognÞ and a storage complexity of one for the GC and Oolong for each group member. It has been shown that if a group member can store atmost OðlognÞ keys, then the lower bound of the communication complexity is OðlognÞ if a structure preserving protocol is used for group key distribution [10].Thus, the key-tree-based scheme shares of practical interest for variety applications because of its balance between communication complexity and storage complexity.

II Related Work

2.1 The Basic Scheme

Maximum Distance Separable Codes Maximum Distance Separable (MDS) codes are a class of error control codes that meet the Singleton bound. Letting $GF(q)$ be a finite field with q elements, an (n,k) (block) error control code is then a mapping from $GF(q)^k$ to $GF(q)^n : E(m) = c$, where $m = m_1 m_2 \cdots m_k$ is the original message block, $c = c_1 c_2 \cdots c_n$ is its code word block, and $E(\cdot)$ is an encoding function, with $k \leq n$. If a decoding function $D(\cdot)$ exists such that $D(c_{i_1} c_{i_2} \cdots c_{i_k}, i_1, i_2, \cdots, i_k) = m$ for $1 \leq i_j \leq n$ and $1 \leq j \leq k$, then this code is called an (n,k)MDS code. For an (n,k) MDS code, the k original message symbols can be recovered from any k symbols of its code word block. The process of recovering the k message symbols is called erasure decoding. All the symbols are defined over $GF(q)$, and usually, $q = 2^m$. The well-known Reed-Solomon (RS) codes are a class of widely used MDS codes. Notably, the RS codes and other MDS codes can be used to construct secret-sharing and threshold schemes.

Description of the Basic Scheme

For a dynamic multicast group, a session key is issued by a GC. Using this session key, the GC can establish a secure multicast channel with the authorized group members. Every time group memberships change because of the join or leave of some group members, the GC reissues a new session key, which is independent of all the old session keys. This rekeying procedure ensures the security of the current session and that of the old sessions, i.e., the newly joined members cannot recover the communications of the old sessions, and those old members who left the group cannot access the current session. Thus, both the backward secrecy and the forward secrecy of group communication are maintained. The complexity of the rekeying operation is asymmetric between a new member's join and an old member's leave. When a new member joins, the GC can easily multicast the new session key encrypted by the current session key to all the current members, followed by a unicast to the new member to send the new session key encrypted by a predetermined encryption key.Shared between the GC and the new member. Thus, join is easy, with low communication and computation cost.

However, when an old member leaves, the current session key cannot be used to convey the new session key information securely, since it is also known to the old member. Thus, hereafter, we will focus on the rekeying operation for a single member leave. The same idea can easily be extended to other rekeying operations in any key distribution schemes; a basic operation is needed to distribute a piece of secret data to a small group of n members, where each member shares a different individual key with the GC. In all current existing schemes, this operation is fulfilled by the GC using n encryptions, followed by n unicasts.Now, we describe a new scheme that realizes this operation by using one erasure decoding of certain MD Scode, followed by one multicast to all the n members. We call this scheme the basic scheme of key distribution. We will then show that this basic scheme can be easily integrated into any key distribution scheme, especially the schemes based on key trees, to reduce computation cost.
The basic scheme consists of three phases:

1) The initialization of the GC,
2) The join of a new member, and
3) The rekeying procedure whenever a group member leaves.
Here again, a targeted multicast group has n members.

2.2 Group Controller Initialization

Initially, theGCconstructs a nonsystematic (L, n) MDS code C over $GF(q)$ and a secure one-way hash function $H(\cdot)$ whose codomain is $GF(q)$. (For a nonsystematic code, none of the original message block symbols directly appears in the corresponding code word block [19].) The domain of $H(\cdot)$ can be an arbitrary space F that is large enough so that $H(\cdot)$ has a secure one-way property: given any arbitrary $y \in GF(q)$, it is impossible or computationally hard to derive $x \in F$ such that $H(x) = y$. Since other strong properties such as second-preimage resistance [21, chapter 9.2] are not necessary, the hash function H can be implemented more efficiently. The GC then makes both the MDS code C and the one-way hash function H public.

2.3 Rekeying

Whenever some new members join or some old members leave a multicast group, the GC needs to distribute a new session key to all the current members. As already discussed, we will focus on the rekeying operation when an old member leaves. After an old member leaves, the GC needs to distribute a new key to n remaining members to achieve both forward and backward secrecy of the session key.

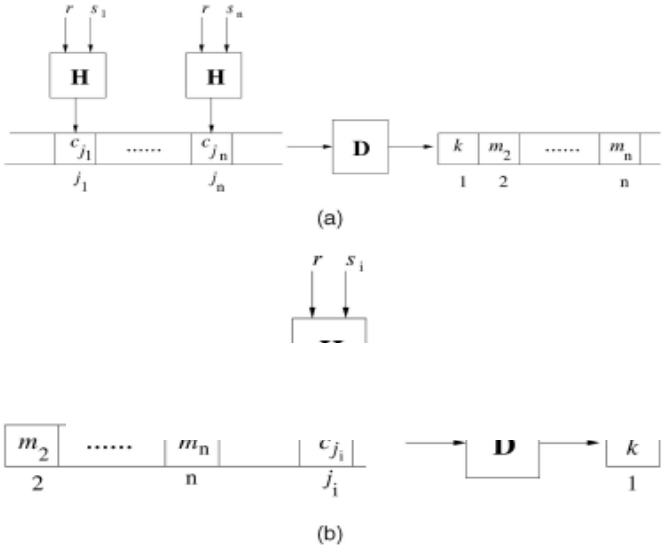
The GC executes the rekeying process in the following steps:
1. The GC randomly chooses a fresh element r in F, which has not been used to generate previous keys.
2. In the remaining group of n members, for each member i of the current group with its seed key (j_i, s_i), the GC constructs an element c_{j_i} in $GF(q) : c_{j_i} = H(s_i + r)$, where $+$ is a simple combining operation in F, for example, string concatenation.
3. Using all the c_{j_i}'s in the above step, the GC constructs a code word c of the (L, n) MDS code C: set the (j_i)th symbol of the code word c to be c_{j_i}. Since C is an (L, n) MDS code, the code word c is uniquely determined by its n symbols. Using an efficient erasure decoding algorithm for C, the GC can easily calculate the n corresponding message symbols $m_1 m_2 \cdots m_n$.
4. The GC sets the new session key k to be the first message symbol $m_1 : k = m_1$. .
5. The GC multicasts r and m2 _ _ _mn.

Upon receiving r and m2m3 _ _ _mn from the GC, an authorized member i of the current group execute the following steps to obtain the new session key:

1. Calculate $c_{j_i} = H(s_i + r)$ with its seed key (j_i, s_i). .

2. Decode the first message symbol m1 from the $(n - 1)$ message symbols m2 _ _ _mn, together with its code word symbol c_{j_i}. .

3. Recover the new session key k, i.e. $k = m_1$.



III. Evaluation of literature survey

As can be seen from the above basic scheme, for all the authorized group members, the GC generates a new session key by generating a common new message word, the first symbol of which is used as the new session key. The new session key is decided by a random element r in F,as well as all the seed keys of the current authorized group members. The random element r and the $(n-1)$ message symbols are multicast in plaintext to all the group members, and the computation cost is thus much lower than using encrypted point-to-point communications for the rekeying process in all existing schemes. The computation operations needed for this new scheme are erasure decoding of a chosen MDS code, a one-way hash function, and some simple combining functions, all of which can be implemented efficiently and are computationally much cheaper than traditional encryptions. As already noted, the purpose of this basic scheme is to replace separate encryptions, followed by unicasts, and be used as a building block for any key distribution schemes whenever applicable. Thus, we examine the communication, computation, and storage costs of the basic scheme and compare them with those of conventional rekeying schemes using point-to-point unicasts secured by separate encryptions. For simplicity, hereafter, we assume that $q = 2^m$, and the size of a new session key is l bits.

Security

Since r and m2 _ _ _mn are multicast in plaintext and are thus known to all interested parties, including unauthorized receivers who attempt to access the current session, the security of the new session key relies on the secrecy of a code word symbol cji that authorized member i of the current multicast group has. The secrecy of cji , in turn, depends on the secrecy that the seed key member i has. Thus, an attacker who attempts to deduce the new session key k has three potential options.

1. brute-force attack, i.e., guessing the session key k itself,
2. Guessing a symbol ck of the code word, or
3. Guessing a current member's seed key.

The effort that an attacker makes to deduce a session key depends on the parameters of the scheme, namely, the size of finite field $GF(2^m)$, the size t of member i's seed key component s_{iv}, and the size of a random number r. Intuitively, when proper hash function $H(\cdot)$ and a random number generator are chosen, the larger these parameters are, the more the effort that an attacker needs to make, and thus, the more secure this scheme is. The following theorem states the exact sizes of the parameters to ensure the security of this scheme:

IV Practical Key Distribution Applying the Basic Scheme to Key Trees

The basic key distribution scheme reduces computation complexity by replacing computationally expensive encryption and decryption operations with more efficient erasure decoding operations of MDS codes. This basic scheme has the same communication complexity as conventional key distribution schemes using secure unicasts. Thus, the basic scheme can be readily used as a building block to replace encrypted unicasts in any key distribution schemes, particularly schemes with low communication complexity.

To reduce the communication complexity of rekeying operations, a key – tree based scheme and many of its variations have been proposed [37], [22], [38], [39], [34]. This scheme reduces the communication complexity of rekeing operations to O(logn), where as each member needs to store O(logn) keys, and the GC needs to store O(nlogn) keys, where n is the multicast group size. This is the most practical key distribution scheme, which balances the communication and storage complexity for dynamic multicast key distribution.

Here, we briefly describe a basic key – tree – based scheme for the rekeying operation. The main idea of reducing the rekeying communication complexity of this scheme is to have the GC distribute subgroup keys in addition to individual member keys and the group session key. These keys are arranged in a logical tree group session key. These key are arranged in a logical tree hierarchy, where the group session key serves as the root, the individual member keys are the leaves, and the subgroup keys correspond to intermediate nodes. Each member stores all the keys along the path from the corresponding leaf to the root in the root in the tree. Then , each sub group key can be used to securely multicast to the members that are leaves of the corresponding sub tree. During the rekeying process, the GC can thus securely multicast to a subgroup of members using their shared subgroup key instead of individual member keys.

This rekeying procedure ensures the security of the current session and that of the old sessions, i.e., the newly joined members cannot recover the communications of the old sessions, and those old members who left the group cannot access the current session. Thus, both the backward secrecy and the forward secrecy of group communication are maintained. The complexity of the rekeying operation is asymmetric between a new member's join and an old member's leave. When a new member joins, the GC can easily multicast the new session key encrypted by the current session key to all the current members, followed by a unicast to the new member to send the new session key encrypted by a predetermined encryption key.

V. Conclusion

We have presented a dynamic multicast key distribution scheme using MDS codes. The compution complexity of key distribution is greatly reduced by employing only erasure decoding of MDS codes instead of more expensive encryption and decryption computions.Easily combined with key trees or other rekeying protocols that need encryption and decryption operations, this scheme provides much lower compution complexity while maintaining low and balanced communication complexity and storage complexity for dynamic group key distribution.

VI .References

[1] AES Algorithm (Rijndael) Information, http://csrc.nist.gov/Crypto Toolkit/aes/rijndael/, 2007.
[2] M. Abdalla, Y. Shavitt, and A. Wool,"Towards Making Broadcast Encryption Practical," ACM Trans. Networking, vol. 8, no. 4, pp. 443-454, Aug. 2000.
[3] M. Blaum, J. Bruck, and A. Vardy, "MDS Array Codes with Independent Parity Symbols," Trans. Information Theory, vol. 42, no. 2, pp. 529-542, Mar. 1996.
[4] R. Blom, " An Optimal Class of Symmetric key Generation Systems," Advances in Cryptology – Proc. Workshop Theory and Application of Cryptographic Techniques (EUROCRYPT 84), pp. 335 – 338, 1984
[5] J. Bloemer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, and D. Zuckerman, "An XOR – Based Erasure – Resilient coding Scheme," Technical Report TR – 95 – 048, Int'l Computer Science Inst., Aug. 1995.
[6] R.E. Bryant and D.R. O'Hallaron, Computer Systems: programmer's Perspective. Prentice Hall, 2002,