Key refreshment in overlay networks: a centralized secure multicast scheme proposal

J.A.M. Naranjo
Dpto. de Arquitectura de Computadores y Electrónica
Universidad de Almería
jmn843@ual.es

L.G. Casado
Dpto. de Arquitectura de Computadores y Electrónica
Universidad de Almería
leo@ual.es

J.A. López-Ramos
Dpto. de Álgebra y Análisis Matemático
Universidad de Almería
jlopez@ual.es

Abstract

This paper introduces a method for renewing secrets which are shared by a set of hosts. The method is centralized, secure, efficient and scalable to a reasonable size. It can be used to refresh cryptographic keys in a centralized multicast overlay. The method is compatible with any multicast topology used underneath. Execution times are shown and the scheme is compared to other existing alternatives.

1 Introduction

Application level multicast overlays have progressively filled the space left by IP multicast protocols, such as IGMP, since they stand as a cost-effective and easy-to-deploy alternative: they run on the OSI application layer, while IP multicast is hold on the network layer [15].

According to the source of the transmitted data, multicast schemes can be divided into *one-to-many* and *many-to-many*. In the first case, the source is one entity only. A typical scenario is an IPTV or P2PTV platform, in which clients receive a TV signal from a Contents Server via Internet. In the second case, many clients (or all) act both as source and receiver. Multiconferences are examples of this.

Still, there is an important issue that needs a definitive solution: privacy in multicast communications. The typical approach in order to maintain a continuous private communication with a group of hosts is to establish a common secret key to encrypt the information. The key is then refreshed periodically to prevent attacks from outsiders, or even insiders. This solution is named *secure multicast*. Huge efforts are still directed towards finding efficient, scalable and secure enough methods.

Depending on how key distribution and management are carried out, secure multicast schemes can be divided into *centralized* and *distributed*. Centralized schemes depend directly of a single entity, a key server, to distribute every cryptographic key. In a distributed approach the key distribution process is more complex, usually involving entities that act as subservers and manage subgroups of users. Re-encryption of the information is needed in some cases.

The heart of the matter in secure multicast is the necessity, in many cases, for perfect *backward* and *forward secrecy* (a client should not be able to decrypt any ciphered information transmitted before his/her arrival to the system and after his/her departure, respectively). This requirement forces to refresh all the keys used to encrypt the data whenever a client joins/leaves the system. If the members set is large and the joinings/leavings occur frequently then the refreshment operation may become an important bottleneck. Many schemes have been proposed as a solution. The following paragraphs summarize some of them.

The *Secure Lock* solution is proposed in [3]. It is based on the Chinese Remainder Theorem. The scheme does not impose any given topology. The problem relies on the ineffi-

cient computations required at the Key Server side on each key refreshment: the computation time needed quickly becomes excessive when the number of members grows [8].

RFC 2627 [14] presents some approaches to the problem. Among all, the *Hierarchical Tree Approach* (HTA) is the recommended option. It uses a logical tree arrangement of the members in order to facilitate key distribution. The benefit of this idea is that the storage requirement for each client and the number of transmissions required for key renewal are both logarithmic in the number of members.

In [12], a divide-and-conquer extension to Secure Lock is proposed. It combines the Hierarchical Tree Approach and the Secure Lock: members are arranged in a HTA fashion, but Secure Lock is used to refresh keys on each tree level. Therefore, the number of computations required by Secure Lock is reduced.

IOLUS [10] is a well known framework designed for the secure multicast problem. Nodes are physically distributed in subgroups, which are organized on a tree fashion. Some special trusted nodes handle the subgroups and serve as gateways among them. It supports huge sets of members, due to its distributed nature. Since IOLUS is a framework, not a protocol, the key refreshment scheme used within subgroups is not stated. Any scheme can be used.

An IETF Working Group, MSEC [1], is currently working in a set of protocols to standardize secure multicast. They are focusing, in an initial stage, in IP-layer centralized multicast, assuming the presence of groups and a single trusted entity in each one.

This paper presents a centralized secure multicast scheme with the following features:

- Suitable for all topologies. No need for node hierarchies, though they can be supported.
- No need for re-encryption.
- Only one secret piece of info is held by each client. We call this pieces *member tickets*.
- Cost-effective and easy to deploy.

The rest of the paper is organized as follows. Section 2 introduces the scheme and the theory underneath. Section 3 discusses some security and efficiency considerations and show the execution times we obtained. Section 4 compares the scheme with other well known alternatives mentioned above. Finally, Section 5 presents the conclusions of the paper.

2 Description of the scheme

The scenario assumed is a Key Server and a set of members (other hosts) that either send or receive multicast messages. Data communications can therefore be either one-to-many or many-to-many, and are encrypted with a symmetric key. Communications related to key refreshment are always one-to-many, from the Key Server to the members. Members can enter and leave the system at any time. The key must be refreshed upon member arrival or departure to achieve perfect backward and forward secrecy, respectively. In addition, a key refreshment must be performed periodically to prevent statistical or brute force attacks. All setup tasks are carried out by a Key Server. Any multicast topology can be used underneath.

Let us assume r is the symmetric encryption key to be multicast, and that there are n members at a given time. The following paragraphs explain how the scheme works.

When a member i joins, the Key Server assigns it a member ticket, x_i. Every ticket is a large prime[1] and is communicated to the corresponding member under a secure channel: SSL/TLS, for example. This communication is made once per member only, so it does not affect global efficiency. All tickets must be different from each other, at least during a relatively wide period of time. Note that x_i is known only by its owner (and the Key Server), and r is shared by all members (and the Key Server).

The distribution of r is done as follows.

[1] Strictly, it is sufficient that all x_i are coprime and greater than δ. In that case, however, it would be necessary that every x_i has a large prime factor in order to make the factorization of L harder (L will be introduced immediately).

1. The Key Server selects:
   - m and p, large prime numbers, such that p divides $m-1$.
   - k and δ, such that $\delta = k + p$ and $\delta < x_i$, for every $i = 1, \ldots, n$.
   - g that verifies $g^p = 1 \ mod \ m$ (such a value is easy to calculate[2]).

   The encryption key r consists of $r = g^k \ mod \ m$.
2. The Key Server calculates $L = \prod_{i=1}^{n} x_i$. L is kept private in the Key Server.
3. The Key Server finds u, v, by means of the Extended Euclidean Algorithm [9], such that
$$u \cdot \delta + v \cdot L = 1 \quad (1)$$
4. The Key Server multicasts (makes public) g, m and u on plain text.
5. Each member i calculates $u^{-1} \ mod \ x_i = \delta$ and $g^{\delta} \ mod \ m = g^k \ mod \ m = r$. The length of r, by definition, can not exceed that of m.

New values for m, g, p and/or k must be chosen for each refreshment of r. Note that δ, u and v depend on them and will change as they do.

In an event-driven refreshments scenario, the refreshment operation must be performed at least in the following two cases:

Member j joins: x_j is included in the product L.

Member j leaves: The leaving member should not be able to decrypt contents anymore. This is achieved by dividing L by x_j and refreshing r afterwards.

Note that it is not necessary to recompute L from scratch: only a single multiplication or division is required for a join or a leave, respectively. Finally, for security reasons, the Key Server might decide to refresh r after a given period of time with no members joining or leaving.

[2]Once the Key Server has chosen $m = p \cdot q + 1$, a value a is chosen satisfying that $m-1$ is the least integer such that $a^{m-1} \ mod \ m = 1$ (that is, a is a primitive value from $\mathbb{Z}_m$). Then $g = a^q \ mod \ m$.

2.1 Proof of correctness

Given that $\delta < x_i, \ i = 1 \ldots n$ and with every x_i prime (or coprime at least), it is clear that:

$$gcd(\delta, x_i) = 1, \text{for every } i = 1, \ldots, n. \quad (2)$$

and hence,

$$gcd(\delta, L) = 1 \quad (3)$$

Equation (3) ensures, by the Extended Euclidean Algorithm, the existence of $u, v \in \mathbb{Z}$ such that $\delta \cdot u + v \cdot L = 1$, from where it is deduced that $\delta \cdot u = 1 \ mod \ x_i$ and so $u^{-1} = \delta \ mod \ x_i$, for every $i = 1, \ldots, n$. The Chinese Remainder Theorem guarantees that the solution for $u^{-1} \ mod \ x_i = \delta$ and $\delta < x_i$, for every $i = 1, \ldots, n$ is unique.

The value $r = g^k \ mod \ m$ is obtained as shown next:

$$\begin{aligned} g^{\delta} &= g^{k+p} \ mod \ m \\ &= g^k \cdot g^p \ mod \ m \\ &= g^k \cdot 1 \ mod \ m \\ &= g^k \ mod \ m \end{aligned} \quad (4)$$

g is public, but the use of δ assures that an outsider will not be able to guess k and, therefore, r.

3 Security and efficiency considerations

Security in the distribution of r relies on the unfeasibility of calculating the right δ in a reasonable time if a valid x_i is not known by the attacker (recall that values for Eq.(1) are unique). The privacy of k and p is guaranteed if:

- a sufficiently large value is chosen for m,
- p and q have a similar bitlength (recall that $m - 1 = p \cdot q$).

In that case factorizing $m-1$ will be more difficult. Additionally, a strong prime can be chosen for m.

Note that the product L is not public in order to make attackers' work more difficult. In

case L was discovered and factorized by an attacker, she would gain access to every member ticket. But such a factorization is impractical by means of a brute force attack. A legal member, say i, might be tempted to factorize $\frac{u \cdot \delta - 1}{x_i}$. If she was successful she would obtain, again, every other ticket included in L. The problem of factorizing such a value, however, is equivalent to that of factorizing L (recall that x_i divides $u \cdot \delta - 1$, for every $i = 1, \ldots, n$, and so $\prod x_i = L$ is a factor of $u \cdot \delta - 1$).

Nevertheless, there is a security measure that must be taken when a new member joins: she should not be assigned a previously used ticket (at least recently). This is done to prevent the old owner to keep intercepting refreshment messages and using the old ticket to discover the secret.

Regarding efficiency, Kruus [7] suggests five issues that a multicast key management protocol must address. They are:

1. efficiency in initial keying,
2. efficiency in rekeying,
3. computational requirements,
4. storage requirements,
5. scalability.

There is no difference in our scheme between first time keying (requirement 1) and further rekeying operations. Rekeying operations are simple (requirement 2): the Key Server generates a single message which is injected into the multicast network on plain text, since only authorized members will be able to process it correctly.

Requirements 3, 4 and 5 are discussed in the next subsection.

3.1 Achieving scalability

We can observe that L will be large, given that $L = \prod_{i=1}^{n} x_i$. So will be u (recall Eq. (1)). To estimate it, assume that every x_i value is stored in an unsigned binary data type of b bits. The greatest value that can be represented is $2^b - 1$. Assume also there are n members. The maximum length of L is then $n \cdot b$ bits. That is also the maximum length of u.

As an example, for $b = 64$ and $n = 1000$ the maximum length of u is 64000 bits $\approx$ 8 KBs. Though that is an affordable message length for many devices (requirement 4), a shorter message would be desirable.

From previous consideration we assume that Kruus' requirements 3 and 5 are the weakest points of our scheme. The solution that allows to overcome these problems consists of dividing the audience into subgroups and delivering the same encryption key to all of them. For the rest of the paper we assume there are s subgroups, each one with a similar number of members. Still, the join and leave operations require the whole set of members to obtain a new key, therefore s refreshment messages (g, m and the corresponding u) must be computed and multicasted now; each one for a different subgroup (note that the final bandwidth requirement does not change). Figure 1 shows an example.

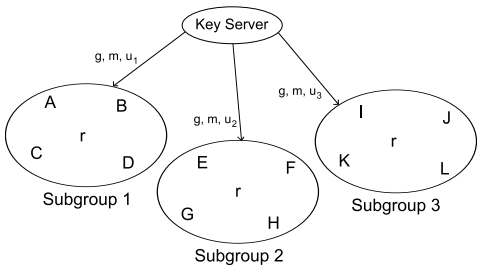

Figure 1: The subgroup extension to the scheme. Capital letters denote members. r is the multicasted encryption key.

Adopting this approach brings many benefits. First, it is obvious that, for a fixed number of members, the length of u values decreases linearly as the number of subgroups increases. In the previous example, arranging the same audience in 20 groups of 50 members would yield 20 messages of 3200 bits = 400 Bs maximum, each one shorter than a typical X.509 certificate. Shorter messages will be handled more easily and quickly by the recipients. This means less hardware requirements.

Second, the message generation process that takes place at the Key Server can be sped up. Every different u can now be computed by a separate process, which may run concurrently

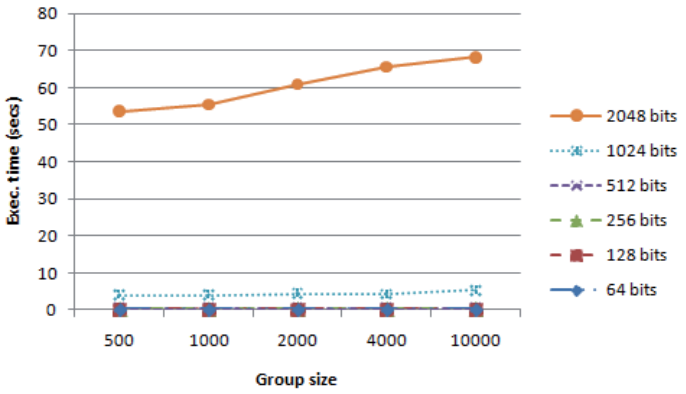

Figure 2: Key Server execution times for different ticket lengths and group sizes.

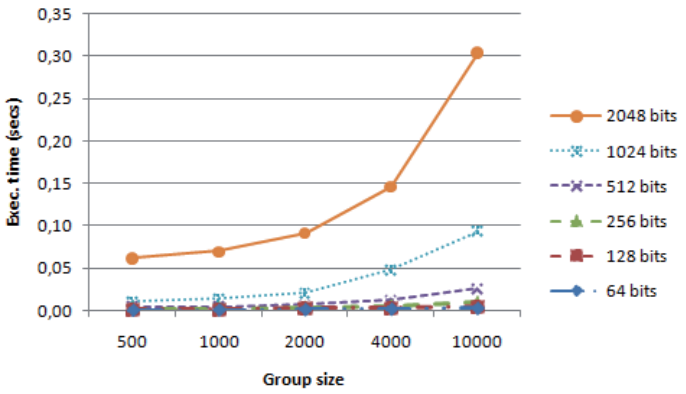

Figure 3: Member execution times for different ticket lengths and group sizes.

with the others. This is specially appropriate for current multi-core processors. The whole process can be sped up by almost s times if the software is properly tuned.

This subgroup approach provides a better scalability, allowing to increase the maximum number of clients that can be handled. As a remark, users should be assigned to subgroups in a balanced way, in order to keep refreshment messages as short as possible. This raises other issues, such as the problem of rebalancing subgroups after a leave avalanche, that are not considered here.

3.2 Simulation

We have developed a Java implementation of the scheme in order to perform simulations and obtain execution times. The BigInteger Java class was used for handling large numbers, and the Miller-Rabin test was employed for primality tests. Figures 2 and 3 show execution times of the algorithm in Section 2, both in the Key Server and in a member, for

different group sizes and ticket lengths. They were obtained in a Intel Core 2 Duo processor at 2,26 GHz with 3 MB of L2 cache and 2 GB of RAM. Despite using a bi-core processor, the application was not parallelzed. Better times are therefore expected for a parallel version of the application.

Two main conclusions can be extracted from the Key Server times. First, key pair refreshment messages are computed very fast, excepting in the case of 2048 bits. This means that the scheme can be applied to a wide variety of scenarios. Second, execution times are mainly affected by ticket length and not by the number of members considered. That is good news when large audiences are addressed. However, remember that the length of the refreshment message might force the audience to be split into several subgroups (see Section 3.1).

Member times show that retrieving the secret is a very fast process. The main problem at the member side is the need to store and handle long refreshment messages when large groups are used. This can be an issue if small devices are used as members: the hardware used must have a sufficiently large memory as well as arbitrary-precision arithmetic capabilities.

4 Comparison with other multicast schemes

Table 1 compares our scheme with other well known alternatives, which were briefly presented in Section 1. More concretely, we consider the Hierarchical Tree Approach (HTA) in its "multiple keys per message" version [14], the Secure Lock extended with HTA [12] and IOLUS [10]. Our scheme is analyzed in its subgroups version (see Section 3.1). Three assumptions have been made:

- both backward and forward secrecy are provided,
- trees (where applicable) are balanced and full,
- members (where applicable) are fairly distributed into subgroups.

Some of the information shown in the Table was taken from [2]. "# keys in Key Server" and "# keys in a member" denote the number of secret keys (on tickets in our case) the Key Server and each member must store, respectively. "# messages on Join" and "# messages on Leave" denote the number of messages that must be sent on a Join and Leave event, respectively. Data re-encryption means the necessity of decrypting and encrypting again information messages at some point in the network, like IOLUS does.

Results depend on several variables: n and s for IOLUS and our scheme, d and h for HTA and Secure Lock + HTA (obviously, h depends on n and d).

Regarding storage, IOLUS has the lowest requirements. On the other hand, HTA and Secure Lock + HTA need the Key Server to maintain a set of keys for the intermediate logical nodes of the tree. Some of these keys must be held by the corresponding users, too. The amount of info to be stored by the Key Server in our scheme is directly proportional to the number of users. However, members themselves only need to know their ticket.

The join operation is very efficient in IOLUS, since only one multicast message within the joined subgroup is needed. Our scheme requires a key refreshment for every subgroup in the system, while Secure Lock + HTA needs a refreshment message per tree level. The HTA solution multicasts more than one message per tree level. Initial communications via a secure channel between the joining member and the server (or trusted entity in IOLUS) are not considered here, since they do not affect global efficiency.

In our scheme, as well as in HTA and Secure Lock + HTA, leave operations are similar to joins. In IOLUS, instead, the trusted entity that manages the affected subgroup must inform all the members that remain in it, and other subgroups' managing entities (all of them in the worst case).

Finally, there is no necessity to re-encrypt multicasted data (this means actual encrypted information), except for IOLUS. However, an indirection mechanism was proposed to over-

	HTA	Secure Lock+ HTA	IOLUS	Ours
# keys in Key Server	$d^h - 1$	$d^h - 1$	$\frac{n}{s} + s$	n
# keys in a member	$h + 1$	$h + 1$	1	1
# messages on Join	$\frac{d-1}{h}$	h	1	s
# messages on Leave	$\frac{d-1}{h}$	h	$\frac{n}{s} + s$	s
Data re-encryption	No	No	Yes	No

Table 1: Secure multicast schemes comparison (n is the number of members, s is the number of subgroups, d is the tree degree and h is the tree depth in HTA and Secure Lock + HTA).

come this problem [10]. Still, the trusted entity in every subgroup must handle every multicasted message.

Choosing a scheme or another will depend on the scenario considered. HTA, Secure Lock + HTA and ours will probably be able to handle medium sized audiences at a low implantation cost, such as Internet radio, multiconferences and stock quote services. On the other hand, IOLUS can be used with huge audiences at a higher deployment cost. Typical scenarios for IOLUS would be pay-per-view platforms and the broadcasting of great interest events.

An interesting point is that IOLUS is simply a framework: it does not specify which secure multicast protocol should be used within every subgroup. We believe that joining IOLUS and our scheme would result in a efficient and secure solution: while IOLUS would provide great scalability, local instances of our scheme would efficiently and securely handle every subgroup.

5 Conclusions

This paper introduces a novel secure multicast rekeying scheme. It is centralized, secure and reasonably efficient. Any overlay topology is supported, and there is no need for intermediate trusted nodes. A proof of correctness is provided, and some security and efficiency considerations are discussed.

In order to assure security, members must compute a modular inverse, being the modulus the corresponding member ticket, which is different for every member. It is computationally unfeasible to find the correct result without knowing a valid ticket. Regarding efficiency, execution times from a simulation were presented, showing that the scheme can be applied to a real scenario. Finally, a comparison with other well known alternatives was discussed.

Our plans for the future include addressing authentication, both of the Key Server and the members. Other possible applications, such as mobile networking, ubiquitous computing and related technologies will be considered too.

6 Acknowledgements

J. A. M. Naranjo and L. G. Casado are supported by the Spanish Ministry of Science and Innovation (TIN2008-01117). L. G. Casado is also supported by funds of Junta de Andalucía (P08-TIC-3518). J. A. López-Ramos is supported by the Spanish Ministry of Science and Innovation (TEC2009-13763-C02-02) and Junta de Andalucía (FQM 0211).

References

[1] Msec working group. `http://www.ietf.org/dyn/wg/charter/`

`msec-charter.html`.

[2] K.-C. Chan and S.-H. Chan. Key management approaches to offer data confidentiality for secure multicast. *Network, IEEE*, 17(5):30–39, Sept.-Oct. 2003.

[3] G.-h. Chiou and W.-T. Chen. Secure broadcasting using the secure lock. *IEEE Trans. Softw. Eng.*, 15(8):929–934, 1989.

[4] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[5] Y.-L. Huang, S. Shieh, F.-S. Ho, and J.-C. Wang. Efficient key distribution schemes for secure media delivery in pay-tv systems. *IEEE Transactions on Multimedia*, 6(5), 2004.

[6] International Teleommunication Union. ITU-R Rec. BT.810. Conditional-Access broadcasting systems. 1992.

[7] P. S. Kruus. A survey of multicast security issues and architectures. In *Proceedings of 21st National Information Systems Security Conference*, pages 5–8, 1998.

[8] P. S. Kruus and J. P. Macker. Techniques and issues in multicast security. In *Proceedings of Military Communications Conference, MILCOM, 1998*, pages 1028–1032, 1998.

[9] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. CRC Press, 1996.

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

[11] J. A. M. Naranjo, J. A. López-Ramos, and L. G. Casado. Key management schemes for peer-to-peer multimedia streaming overlay networks. In *WISTP '09: Proceedings of the 3rd IFIP WG 11.2 International Workshop on Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks*, pages 128–142, Brussels, Belgium, 2009. Springer-Verlag.

[12] O. Scheikl, J. Lane, R. Boyer, and M. Eltoweissy. Multi-level secure multicast: the rethinking of secure locks. In *Parallel Processing Workshops, 2002. Proceedings. International Conference on*, pages 17–24, 2002.

[13] F. Tu, C. S. Laih, and H. H. Tung. On key distribution management for conditional access system on pay-tv system. *IEEE Transactions on Consumer Electronics*, 45:151–158, 1999.

[14] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. RFC 2627, 1999.

[15] H. Zimmermann. OSI reference model—the ISO model of architecture for open systems interconnection. pages 2–9, 1988.