Efficient Secure Group management for SSM

Ghassan Chaddoud, Vijay Varadharajan

Efficient Secure Group Management for SSM

Ghassan Chaddoud and Vijay Varadharajan

INRIA - LORIA
615, rue du Jardin Botanique - B.P. 101
54602 Vandoeuvre-Les-Nancy - FRANCE
{chaddoud}@loria.fr,
WWW home page: http://www.loria.fr

Abstract. The SSM model addresses some of the problems of deployment of IP multicast. However, a real commercial deployment of SSM requires security services. Our work proposes an architecture, called S-SSM, for securing the SSM model. S-SSM defines two mechanisms for access control and content protection. The first one is carried out through subscriber authentication and access permission. The second is realized through the management of a unique key, called the channel key, k_{ch}, shared among the sender and subscribers. The management of k_{ch} is based on a novel distributed encryption scheme that enables an entity to efficiently add and remove a subscriber *without affecting* other subscribers.

1 Introduction

The IP multicast model [5] is appeared as a way to optimize the communication used by multimedia applications (audio and video conferences, video diffusion) involving several participants. But today, we can see that commercial multicast deployment is not yet a reality yet. Moreover, the IP multicast model presents some limits on issues such as scalable routing, address allocation, and notably security. Indeed, any host can send data to any IP multicast address and any host can join any group.

In order to overcome these problems, some simplified approaches have been proposed. Express [10] has defined a point-to-multipoint diffusion, and presents a group as the tuple (S, E), where S is the unique sender and E the multicast destination address. The tuple (S, E) is called *channel*. Simple Multicast [2] is similar to Express but uses a birectional tree like CBT [1]. These different approaches, known as Source Specific Multicast (SSM) has been standardized in IETF with the proposal of PIM-SSM [8], a modified version of PIM-SM [6] which allows source-filtering. Subscription to channels is supported by IGMP Vers.3 [3], called IGMP specific-source [9]. The SSM scheme presents a solution for allocation, routing problems and partial access control.

But for an effective commercial deployment of SSM security services such as access control and content provider protection are required. In this paper, we propose a new architecture for securing SSM that can provide an environment for

multimedia diffusion; we refer to this as S-SSM. The diffusion environment is basically composed of SSM (PIM-SSM/IGMPv3) routing, video server and potential receivers.

Our S-SSM architecture addresses two security services and mechanisms to support them. These are access control and content protection. The access control service and mechanisms are an extension of the solution proposed in [4] which uses a signed token to control access to group communication. The aim of this solution is to authenticate members using their local routers and to protect membership demands against attacks. Content protection is achieved via sender authentication and data ciphering. This last one requires the management of a unique key, called the channel key, k_{ch}, shared among the sender and subscribers. The management of k_{ch} is achieved using a public-key system that enables the owner of the channel to robustly add or remove any subscriber by sending only one multicast message and without any additional computation for the subscribers.

In this paper, we present S-SSM, a secure architecture for an environment of 1-to-n multimedia diffusion. In Section 2, we define our general approach to secure an SSM environment. Section 3 describes in detail the S-SSM architecture. Section 4 outlines the novel distributed encryption scheme used to manage the channel key. Finaly, Section 5 concludes.

2 SSM Model

Our approach considers securing an environment for multimedia multicasting. A diagrammatic representation of such an environment is given in (Fig. 2) which is composed of the following:

- SSM (PIM-SSM/IGMPv3)[8, 9] routers. The IGMP Vers.3 offers support for "source filtering", i.e., the possibility for a system to report interest in receiving packets only from specific sources.
- Application Servers such as Internet TV and Video.
- Potential subscribers who have paid for a service.

In SSM communication, a multicast *channel* is defined as a datagram delivery service [10]. A channel is identified by a tuple (S, E) where E is a *channel destination address* [1]. Only the source host S may send to (S, E). A *subscriber* host requests reception of data sent to a channel by explicitly specifying both S and E in a request, via an IGMPv3-report. The source S sends to a channel by simply transmitting datagrams addressed to E. The network layer guarantees that all datagrams sent by S to destination E are delivered to all subscribers of channel (S, E). The principal advantage of a such multicast diffusion service is a partial access control to the channel:

[1] L'IANA has allocated the adress suffix $232.*.*.*/8$, *i.e,* 2^{24} class D addresses, for experimental use by the singl-source multicast model

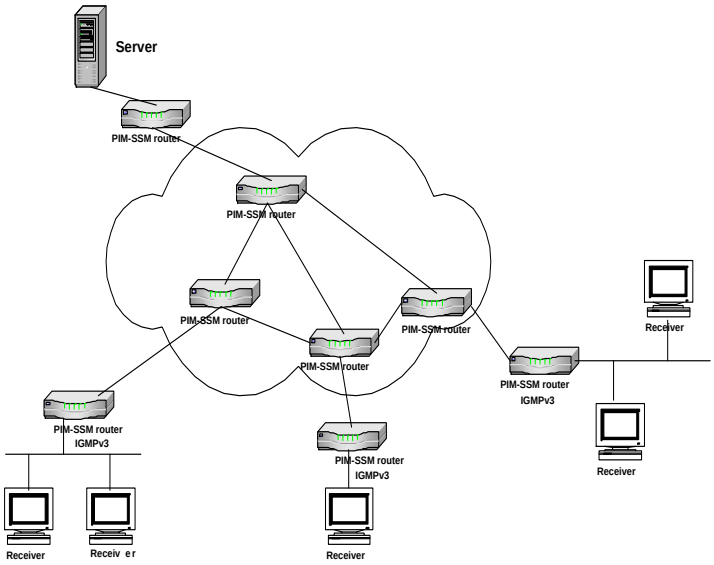

Fig. 1. SSM Enviornment

- A source can not send data to a channel owned by other source. Contrary to the classic IP multicast model, the two channels (S, E) and (S', E) which have the same destination address E are unrelated, despite the common destination address. This is guaranteed by `PIM-SSM` routing.
- A subscriber to (S, E) will not receive data sent to (S', E) unless he subscribes to (S', E). Filtering by source guarantees that a subscriber does not receive all data sent with the same destination address E.

The SSM model appears appealing, but it has some limits when it is used by Internet access providers; for instance, the contractors of multimedia diffusion services such as Internet TV require subscription fee for their services. An SSM model of security must allow only legitimate entities to access to channels for which they have paid subscription fee.

Let us now outline a more complete security architecture S-SSM that supports access control and content protection security services.

- **Access control**. This mechanism aims at protecting network resources against malicious and illegitimate subscription requests. It is realized through:
  - Authentication of new subscribers when they subscribe to a channel.
  - Checking access permission to channel provided by the contractor when s/he subscribes to channel.
- **Content protection**. It is achieved via data ciphering. It requires the management of a unique key, called the channel key k_{ch}, shared among the sender and subscribers. Before diffusion, channel source encrypts the data with the

key k_{ch}. Only subscribers having the key k_{ch} are capable of decrypting data. Thus, in addition to content secrecy, source authentication is ensured, because SSM routing forwards data coming only from the source.

We propose that the first mechanism would be an SSM extension by being integrated and inseparable part of subscription to channels. The second mechanism is independent from SSM and would be used by contractors who require secrecy for their content (according to the security policy of service diffusion).

3 S-SSM architecture

Let us now consider in detail the S-SSM architecture and explain how these offered services are applied at the time of subscription and content diffusion.

3.1 S-SSM overview

Our **S-SSM** proposes that the actors of the diffusion environment, notably the IGMPv3-capable routers, would be responsible for the security of the channel.

The reasons for this are as follows:

- The first one is to control some events that could compromise the network resources, in particular, the multicast routers. When one entity sends a subscription request, i.e, an IGMPv3-report, the first router receiving the request must be capable of sealing the request's fate. Indeed, if the entity meets certain conditions imposed by the channel security, the router can follow up the subscription request. In this way, we limit as much as possible any network resource waste. Moreover, every malicious request is prevented from illegal access to channel flow.
- The second aspect is a scalability concern. The decentralized management of channel security is more flexible than the centralized one. In other words, instead of having only one entity controlling all the security operations within the channel, entities distributed within domains where we have subscribers, can be delegated to do some security tasks. There are many advantages for this choice. The propagation of control messages will be limited to only domains where the subscription requests are issued. Moreover, the subscription latency to a channel is minimized and the bottleneck for the entity responsible for the security management is avoided.

So, the *channel manager* delegates the security tasks to the actors of the diffusion environnment.

The architecture S-SSM defines four security actors (cf. figure 2):

- **Global controller, GC**. It is an entity delegated by an authority responsible for data distribution such as content providers. According to information stored in a server, the global controller grants access to new subscribers and

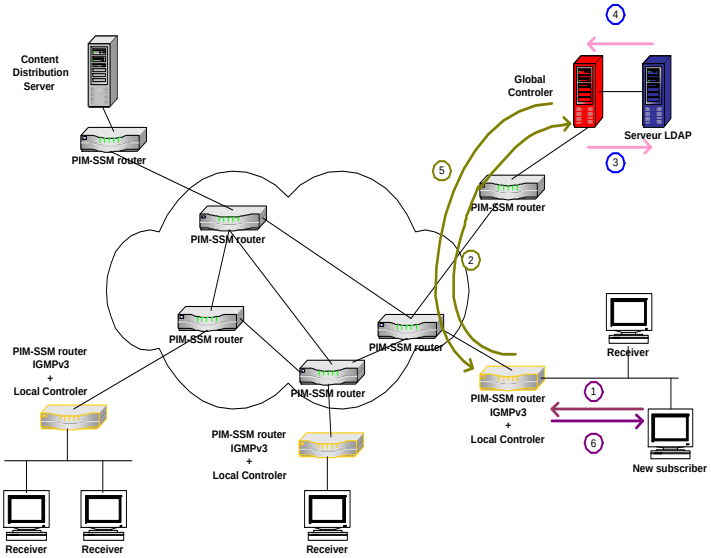

Fig. 2. S-SSM Environment

revokes channel subscribers. It coordinates with *local controllers* in to manage the channel key k_{ch}. In addition, the global controller informs the other controllers if there are any compromised subscribers in their domains. It rekeys the subscriber channel periodically.

- **Local controller, LC**. It is delegated by the global controller within its domain. It authenticates new subscribers and distributes k_{ch} to subscribers within its domains. It can, with the GC authorization, periodically rekey channel. A local controller can be implemented in an IGMPv3-capable router or an IGMP-proxy [7], which allows to extend delegation of the secured environment to a domain larger than a local area network. *Channel controller* can be any local or global controller.
- **Directory Server, DS**. The service provider memorizes all information related to his clients or subscribers and needed to the management of the channel in a server, called *Directory Server*, DS. DS has an entry per subscriber. An entry can be formed of several fields such as *Validity* to prove that the entry is valid, *Start_date* and *End_date* indicating the beginning and ending dates of a subscription, and a specific nonce, Ns, attributed randomly to the subscriber. GC has only read access to DS. An example of a DS could be a LDAP server.
- **Subscriber**. Any entity which can receive channel flow and has paid a subscription fee, according to the service agreement with the service provider. Prior to subscription, the new subscriber must have a valid entry in the DS server. The valid entry defines the *access permission*.

According to the server DS, the GC and local controllers create a global *recovery_list*, `RECOV_List`, composed of entities which have compromised or tried to compromise the channel security. A member of this list has no more access to channel data. In addition, every LC has a list of local subscribers, `LOCALSUB_List`. This list is updated after every join or eviction of a subscriber. A LC distributes the key k_{ch} only to members of the list `LOCALSUB_SUB`.

In the remainder of this section, we discuss how the access control is performed. Then we describe the management of the channel key.

3.2 Access control

In the following, we assume that a local controller is an SSM-capable router (IGMPv3/PIM-SSM) and that IGMPv3 defines a type of message which recognizes the definition of a signed *token*. The signed token forms an essential part of the authentication process of exchanged messages. A token is composed of:

* Ns specific number attributed by the service provider at the time of subscription and payment to the channel service,
* The tuple (S, ch), where S is the address of the source and ch is the channel destination address;
* Timestamp,
* The IP address of the subscriber.

Remark For a signed token of a local controller, the field Ns will be ignored or set to zero. It should be noted that the role of a token is to protect a non-secure subscription request against attacks.

As mentioned in Section 2, the subscription phase is carried out in two steps:

- Authentication carried out by a local controller.
- Access permission check done by the global controller.

In the following paragraphs, we present the access control service using the scenario depicted in figure 2, which shows the subscription of the host hh to a channel (S, ch). We assume that a local controller is on the SSM tree.

Authentication The entity, h, which wants to subscribe to the channel, sends a subscription message, $n°$ 1 as shown in figure 2. It is composed of an IGMPv3-report with ch as the address SSM channel and with S as the address of the channel source. This message is protected by a signed token of h. h sends the message to its local controller. On receipt of this message, the LC authenticates the sender. We assume that the LC knows the public keys of hosts within its domain. The subscription message is:

$h \rightarrow LC$: *IGMP-report*(S, ch), *[token_h]^ pK_h*
[token_h]^ pK_h is the h's token signed with its private key

If the authentication process succeeds, the LC checks whether h is not in the RECOV_List. If not, it must verify with the GC whether h has the right to access the channel (S, ch). If the authentication process fails or if h is in the list RECOV_List, then the message will be ignored. This step triggers the next step, which is the validity check.

Validity check The purpose of this step is to verify with the GC whether the new subscriber has a valid entry in the DS server. The LC sends a message to GC, $n°$ 2 in figure 2. This message contains the tokens of LC and of the new subscriber. The message is:

$LC \rightarrow GC$: *[token_h]^ pk_h, [token_lc]^ pk_lc*
[token_lc]^ pK_lc is the lc's token signed with its private key.

By receiving this message, the GC authenticates first the LC. Then, if successful, it authenticates the host and checks with the DS server its Ns for the required channel and its period validity. This is done with the message $n°$ 3 of figure 2. This message is acknowledged by the DS with message $n°$ 4. If the answer is positive, the GC confirms the validity of the subscription by sending a message $n°$ 5 to the LC:

$CG \rightarrow CL$: *[token_h]^ pK_h, [token_gc]^ pk_gc, Start_date, End_date.*

This message contains the signed token of the host and the token of the GC and *Start_date, End_date* which indicate the beginning and the expiration validity of the subscription. In fact, these fields together with the subscriber's Ns form the *access permission* to the channel.

At the reception of this message, the LC interprets this message as follows:

* If *End_date* > *start_date* AND *TIME* < *End_date* THEN the host can have access to channel data. The LC must start the phase of distribution of the key k_{ch}.
* If *End_date* > *Start_date* AND *TIME* > *End_date* THEN the subscription demand is ignored.

4 Channel key management

In the case of channel diffusion, 1-to-n multicast, where there is only one sender, the channel key is used to ensure confidentiality and sender authentication. The channel rekeying should take place after the following:

– reception of a new subscription request in order to avoid the subscriber from accessing an old channel traffic ;

- access permission expiration or eviction of a subscriber in order to avoid access to future channel flow;
- the periodicity of channel rekeying in order to avoid key forging

In this section we explain how GC distributes securely k_{ch} to subscribers via the intermediate of LCs. The distribution of k_{ch} is based on the public-key distribution encryption scheme proposed in [11]. A distributed encryption system consists of a manager and several users who form a group. The group manager has two major tasks: constructing a unique group public key and performing revocation. In the group, each member has a private decryption key that can be used for decryption. The group possesses a unique public key or a group public key that maps to all private keys owned by its members. A message encrypted using the group public key can be decrypted using any one of these private keys.

In our architecture, the GC and any LC act as a group manager. First, the GC uses encryption key to encrypt the channel key k_{ch} that is then transmitted to LCs. Secondly, every LC acts as group manager within its domain. A LC uses the encryption key to encrypt k_{ch} received from GC and then transmits it to its subscribers, who possess a private decryption key to decrypt k_{ch}.

The major task of constructing such a public-key distribution system is to find an algorithm where a public key maps to several associated private decryption keys. In the following, we describe such a distributed encryption scheme.

4.1 Distributed Encryption Algorithm

The security of this system relies on difficulty of computing discrete logarithm. The protocols are based on a polynomial function and a set of exponentials. Let p be a large prime, Z_p^* be a multiplicative group of order q where $q|p-1$, and $g \in Z_p^*$ be a generator. Let $x_i \in_R Zq$ for $i = 0, 1, 2..., n$ be a set of integers. The polynomial function of order n is constructed as follows.

$$f(x) = \prod_{i=1}^{n}(x - x_i) \equiv \sum_{i=0}^{n} a_i x^i \ (mod q),$$

where $\{a_i\}$ are coefficients: $a_0 = \prod_{j=1}^{n}(-x_j)$, $a_1 = \sum_{i=1}^{n}\prod_{i\neq j}^{n}(-x_j)$, $\cdots$, $a_{n-2} = \sum_{i\neq j}^{n}(-x_i)(-x_j)$, $a_{n-1} = \sum_{i=1}^{n}(-x_j)$, $a_n = 1$.

Note that $\sum_{i=0}^{n} a_i x_j^i = 0$. This property is important for us to construct the distributed encryption system. Having the set $\{a_i\}$, we can then construct the corresponding exponential functions, $\{g^{a_0}, g^{a_1}, \cdots, g^{a_n}\} \equiv \{g_0, g_1, \cdots, g_n\}$. All elements are computed under modulo p. For convenience, we will omit modulo p in the rest of this paper. Note that $\prod_{i=0}^{n} g_i^{x_i} = 1$.

Now we are ready to construct an asymmetric-key system where the encryption key is the tuple $\{g_0, g_1, \cdots, g_n\}$ mapping to n decryption keys $\{x_i\}$. Let $\mathcal{H}$ be a strong one-way hash function and M be the message to be sent to a recipient by the sender who possesses the encryption key. The idea of this protocol is for the sender to encrypt a message M using the encryption key and produce

the corresponding ciphertext that can be decrypted by anyone who has a private key, $x_i \in \{x_1, x_2, \cdots, x_n\}$

To encrypt a message M, the sender picks a random number $k' \in_R Zq$, computes $k = \mathcal{H}(M)$ and encrypts M using the encryption key to obtain the ciphertext $c = (c_1, c_2)$, where $c_1 = (g^{k'} g_0^k, g_i^k)$, $i = 1, \cdots, n$, and $c_2 = Mg^{k'}$. c is sent to recipients.

Since each recipient has his/her own private decryption key and each of these can be used to decrypt the ciphertext, each recipient can obtain M. The process is as follows. For recipient j,

$$d \leftarrow g^{k'} g_0^k \prod_{i=1}^{n} g_i^{kx_j^i} = g^{k'} \prod_{i=0}^{n} g_i^{kx_j^i} = g^{k'} \prod_{i=0}^{n} g^{a_i' k x_j^i} = g^{k'} g^{k \sum_{i=0}^{n} a_i x_j^i} = g^{k'}.$$

The last equality holds because $\sum_{i=0}^{n} a_i x_j^i = 0$. The message can be recovered by computing $M = c_2/d$.

4.2 System Setup

As mentioned before, the GC or any LC can act as a group manager. A group manager needs to prepare the encryption key and all decryption keys for its subscribers. The subscribers for the GC are the LCs and those for a LC are the subscribers of the diffusion channel. We assume that the channel controllers have sufficient computational power, whereas subscribers have limited computational power. In the following we explain the system set up for a group manager (ie GC and any LC).

We assume that the system has the upper limit, n, for the maximum number of subscribers. The actual number of subscribers is say m for $0 < m \leq n$. The difference between n and m will be used for adding new subscribers to the system.

The construction of the encryption key and decryption keys follow the steps given below:

- Select n distinct random numbers $x_i \in_R Zq$ for $i = 1, 2, \cdots, n$, which form a set X_n and a subset $X_m \subset X_n$.
- Compute $A = \prod_{j=1}^{n} (\prod_{i=0}^{n-1} g_i^{x_j^i})$. We will see later that the group manager (broadcaster) needs only to compute A *once*.
- Select an integer $b \in_R Zq$ and compute its multiplicative inverse b^{-1} such that $bb^{-1} = 1 mod q$.
- Compute $\bar{x}_j = b^{-1} \sum_{i \neq j}^{n} x_i^n mod q$, for $j = 1, 2, \cdots, n$.
- Compute $\hat{x}_j = s_j x_j^n mod q$, where $s = s_1 s_2 \cdots s_n$, and $s_i s_i mod q = 1$, $\forall s_i \in Zq$, i.e., the multiplicative inverse is itself. It is easy to see that it can be realised by simply setting $q = s_i(s_i - 1)/k$ for an integer k such that $q + 1$ is still a prime. The solutions of s_i can be found if $1 + 4kq = X^2$ where X is an odd number. It is not difficult to see that there are infinite solutions when we let k be $k'(1 + k'q)$ for an integer k'.

These values satisfy the equality:

$$A^s g^{sb\bar{x}_j} g^{s\hat{x}_j} = 1, \quad \forall j \in \{1, 2, \cdots, n\}.$$

A is kept by the group manager and will be used as the encryption key for broadcasting the channel key K_{ch}. Since the encryption key is not public, there is no need for us to protect it against any illegal modification.

$\bar{x}_j$ and $\hat{x}_j$ are given to subscriber j as its secret decryption key during the process of its subscription to the channel.

4.3 Broadcasting Protocol

We show in the following how the channel key is distributed securely to subscribers by the intermediate of LCs, using the distributed encryption system. The unique encryption key A of GC is used for the encryption of K_{ch} . Any LC who has a legitimate private decryption key can decrypt them. The broadcasting protocol is given as follows:

- 1) GC Select an integer $k \in_R Zq$.
- 2) Compute $\bar{g} = g^{sk}$ and $\hat{g} = g^{sbk}$.
- 3) Compute the ciphertext $c = K_{ch} A^{sk}$.
- 4) Broadcast the triplet $(\bar{g}, \hat{g}, c)$ to all LCs.

To decrypt it, the LC j computes

$$c\hat{g}^{\bar{x}_j} \bar{g}^{\hat{x}_j} = K_{ch}.$$

Next, every LC uses its unique encryption key A for the encryption of K_{ch}. Then, it follows the step from 1) to 4) accomplished by the GC. To decrypt it, the subscriber j computes

$$c\hat{g}^{\bar{x}_j} \bar{g}^{\hat{x}_j} = K_{ch}.$$

The completeness of this protocol is clear:

Lemma 1. *For a given ciphertext c, if the GC and the LCs follow the correct encryption procedure, any registered subscriber can correctly decrypt the ciphertext to obtain* K_{ch}.

Proof: Proof is straight forward. It is based on the following: when a registered subscriber j decrypts the channel key, s/he does not change the value of s, i.e. $ss_j = s \mod q$. Therefore, in the decryption, A can be removed from c.

$$\begin{aligned} c\hat{g}^{\bar{x}_j} \bar{g}^{\hat{x}_j} &= K_{ch} A^{sk} (g^{sbk})^{b^{-1} \sum_{i=1, i\neq j}^{n} x_i^n} (g^{sk})^{s_j x_j^n} \\ &= K_{ch} A^{sk} (g^{sk})^{\sum_{i=1}^{n} x_i^n} = K_{ch} \prod_{j=1}^{n} (\prod_{i=0}^{n} g_i^{x_j^i}) = K_{ch} \end{aligned}$$

The last equality is based on $\prod_{j=1}^{n} (\prod_{i=0}^{n} g_i^{x_j^i}) = 1$.

4.4 Removing a subscriber

There are two schemes for removing a subscriber.

4.5 Scheme 1

To remove a subscriber γ from the list or x_γ from X_m, its LC needs to

- Recompute A:
$$A' = \prod_{j=1, j\neq\gamma}^{n} (\prod_{i=0}^{n-1} g_i^{x_j^i}).$$
- Compute a new parameter $d = g^{-\hat{x}_\gamma}$.

To broadcast the channel key, the controller now needs to compute the ciphertext in terms of
$$c' = K_{ch} A'^{s'k} d^{s'k},$$
where $s' = \sum_{i=1, i\neq\gamma}^{n} s_i mod q$. The broadcasted token consists of $(\bar{g}, \hat{g}, c')$, where $\bar{g} = g^{s'k}$ and $\hat{g} = g^{s'bk}$.

To decrypt it, the subscriber j computes
$$c' \hat{g}^{\bar{x}_j} \bar{g}^{\hat{x}_j} = K_{ch}.$$

4.6 Scheme 2

This scheme is much simpler than the first scheme. The channel controller does not need to reconstruct the encryption key A. Instead, the broadcaster just recomputes s such that the s_γ for the subscriber to be removed is moved from the computation, i.e., $s = \sum_{i=1, i\neq\gamma}^{n} s_i$. Under this scheme, the broadcasting protocol given before can still be used without any modification. The removed subscriber cannot decrypt K_{ch} since $s_\gamma s \neq s mod q$, while other subscribers can still decrypt K_{ch} as usual.

Remark The GC can do the same thing in order to remove a LC.

4.7 Adding a Subscriber

There are two methods for a LC to add a new subscriber to the system.

1. The first approach makes use of an element in the spare set $X_n - X_m$. Recall that we have assumed that the actual number of subscribers is less than n, i.e., $m < n$ or $X_m \subset X_n$. To add a new subscriber, the LC just simply moves one unused element from $X_n - X_m$ to X_m. For convenience, we still denote by X_m the new set. Suppose that the new subscriber is denoted by subindex ℓ and is given $\bar{x}_\ell$ and $\hat{x}_\ell$ as his/her decryption key doublet.

2. The second approach reuses x_γ, after $x_\gamma \in X_m$ has been removed from the system. Once x_γ has been removed from the system, the corresponding party γ is no longer able to decrypt K_{ch}. However, because x_γ still exists in X_m, it is perfectly legal for the channel controller to reuse it, provided that the removed subscriber γ cannot gain anything from it. The algorithm for the channel controller to reuse x_γ is actually simple; for a new subscriber r the LC needs only to generate a new "s_r" and compute a new "s" by replacing s_γ with s_r, i.e., $s = s_1 s_2 \cdots s_r \cdots s_n$. The new decryption key is then constructed in the same manner: $\bar{x}_r = b^{-1} \sum_{i \neq r}^{n} x_r^n mod q$ and $\hat{x}_r = s_r x_r^n mod q$. The soundness of this scheme is based on the following: $s\hat{x}_r = ss_r x_r^n = sx_r^n mod q$ and $s\hat{x}_\gamma = ss_\gamma x_\gamma^n \neq sx_\gamma^n mod q$.

4.8 Analysis

In this section, we outline the computations involved in the setup and operation of the system. We derive the number of operations involved in computing A, performing encryptions and decryptions as well as for adding and removing subscribers.

In terms of performance,the main objective of the proposed distributed encryption scheme is to reduce the amount of computations that need to be done during system operation. This resulted in a trade off and the bulk of computations being done at system set up time. This is advantageous as it can be done before system operation and more importantly by the broadcaster rather than the service subscribers.

Here are the rough estimates of exponentiation operations that need to be done in the various computations. Note these operations are modulo operations. As far as the computation of A is concerned at system setup time, it requires of the order of n^2 exponentiations where n is the number of subscribers. Recall that in our scheme, n is greater than the actual number of subscribers at a given time m and the difference is being used to add new subscribers. The computation required to perform decryption is of the order of 2 exponentiations. With encryption, in general it is 1 exponentiation, though after removal of a subscriber, there will be a need to perform 2 more exponentiation operations. The removal operation with scheme 2 requires 2 additional exponentiations from the broadcaster side and addition does not involve any exponentiation operation. Hence it can be seen that the proposed scheme is very efficient when it comes to normal system operation and updating of subscribers.

5 Concluding Remarks

In this paper, we have proposed a security architecture for source specific multicast (SSM) communications. The architecture enables legitimate subscribers to have secure access to channel flow in a dynamic environment. The access control service protects the network resources against illegitimate subscription requests. It is realized through checking and enforcing access permissions to a channel

after authentication. The confidentiality and sender authentication are achieved through the establishment and management of the channel key shared between the source and the subscribers of the channels. Then we have proposed the application of a novel distributed encryption scheme for dynamic group management. The presented scheme is efficient when it comes to addition and removal of subscribers. These are done without affecting the rest of the subscribers. We have also briefly outlined the amount of computations involved in secure communications as well as addition and deletion of members. We are currently exploring a larger scale implementation of the proposed scheme.

References

1. A. Ballardie. Core Based Trees (CBT version 2) Multicast Routing, September 1997. RFC-2189.
2. T. Ballardie, R. Perlman, C. Lee, and J. Crowcroft. Simple scalable internet multicast, April 1999. tech. rep., University College London.
3. B. Cain, S. Deering, W. Kouvelas, and A. Thyagarajan. Internet group management protocole, version 3, January 2001. Intenet draft.
4. G. Chaddoud, I. Chrisment, and A. Schaff. Dynamic group communication security, May 2001. MMM-ACNS 2001, SaintPetersburg, Russia.
5. S. Deering. Multicast routing in a datagram internetwork, Octobre 1991. Ph.D. thesis, Stanford University.
6. D. Estrin, D. Farinacci, and A. et al. Helmy. Protocol independent multicast sparse mode (pim-sm): Protocol specification, Juin 1997. RFC 2117.
7. B. Fenner, H. He, B. Haberman, and Sandick H. Igmp-based multicast forwarding (igmp proxying), November 2001. Intenet draft.
8. H. Holbrook and B. Cain. Source-specific multicast for ip, May 2000. Intenet draft.
9. H Holbrook and B. Cain. Using IGMPv3 For Source-Specific Multicast, november 2001. draft-holbrook-idmr-igmpv3-ssm-02.txt, work in progress, IETF.
10. H. Holbrook and D. Cheriton. Ip multicast channels: Express support for large-scale single-source applications., November 1999. ACM SIGCOMM.
11. M. Yi and V. Varadharajan. Robust and Secure Broadcasting, December 2001. The second international conference on cryptology in India (INDOCRYPT 2001),Chennai, India.