Scalable Multicast Security with Dynamic Recipient Groups.

Refik Molva and Alain Pannetrat
Institut Eurecom - Sophia Antipolis - FRANCE
{molva|pannetra}@eurecom.fr

In this paper we propose a new framework for multicast security based on distributed computation of security transforms by intermediate nodes. The involvement of intermediate nodes in the security process causes a new type of dependency between group membership and the topology of the multicast network. Thanks to this dependency, the containment of security exposures in large multicast groups is assured. The framework also assures both the scalability for large dynamic groups and the security of individual members. Two different key distribution protocols complying with the framework are introduced. The first protocol is an extension of the ElGamal encryption scheme whereas the second is based on a multi-exponent version of RSA.

Categories and Subject Descriptors: K [6]: 5—*Security and Protection*

General Terms: Multicast, Security

Additional Key Words and Phrases: Group communications, Confidentiality, Key distribution, Scalability, Diffie-Hellman, RSA

1. INTRODUCTION

Multi-party communications have recently become the focus of new developments in the area of applications and networking from group applications like video-conferencing to network layer multicast protocols. As part of the new issues involved with multi-party communications, security in terms of privacy and integrity has received particular attention due to the vulnerabilities inherent to multi-party architectures.

While several projects addressed the problem of key distribution [Steiner et al. 1996][Burmester and Desmedt 1995][Fiat and Naor 1993][Just and Vaudenay 1996] and digital signatures [Camenisch 1997][Chen and Pedersen 1995][Campbell and Wiener 1993][Chaum and Heyst 1991][Camenisch and Stadler 1997] among the participants of a group, the security issues related to multicast in large and dynamic groups remained comparatively unexplored. Like any other multi-party scheme, the inherent complexity of the underlying communication mechanisms exposes multicast protocols to vulnerabilities that have no counterpart in the unicast case as

depicted in [Ballardie and Crowcroft 1995][Mittra 1997]. Possible countermeasures for those vulnerabilities are cryptographic security services ranging from authentication of group members, data confidentiality and integrity, non-repudiation of origin to access control for group membership. Next to basic security services, automatic key management is necessary for the secure provision of large recipient groups with cryptographic keying material.

This paper is concerned with the security of large dynamic multicast groups, involving a one-to-many communication pattern, with a dynamic set of recipients. We present a framework for multicast security that focuses on two issues:

—scalability in large dynamic groups: the amount of processing of each individual component of the multicast security mechanism should be independent of the group size; changes to the group membership should only affect a small subset of the group.

—containment of security exposures through partitioning: each recipient group should be partitioned into sub-groups in order to assure that a security exposure in a sub-group does not endanger the security of other sub-groups.

In the proposed framework, conflicting security and scalability requirements are addressed through a distributed scheme whereby intermediate components take part in the security protocol. The intermediate components share the security processing load with the source and assure the containment of security exposures at various parts of the multicast tree. The framework defines basic properties of a set of cryptographic functions that assure data confidentiality. Depending on the performance of the underlying algorithm, implementations of the framework may be suitable either for the encryption of bulk data or only for the encryption of short messages as required by key distribution. The framework is first validated with respect to security and scalability requirements. Two different implementations of the framework are then discussed. Both solutions are based on asymmetric techniques: an extension of the ElGamal algorithm [Elgamal 1984] and a variation on RSA [Rivest et al. 1978]. These algorithms which offer strong protection are only suitable for key distribution since, due to their inherent complexity, bulk data encryption with these solutions seems prohibitive.

This paper is organized as follows. In section 2, we discuss multicast security in detail and use this analysis to motivate and establish a set of requirements for multicast security in a large dynamic group. Next, in section 3 we introduce a set of cryptographic sequences with special properties that are organized in a tree. In section 4, we apply this formal graph to a multicast tree. We show how the properties of our formal graph can be used to offer a multicast security framework that deals with confidentiality and scalability issues. In section 5 we consider our framework for key distribution and present two key distribution schemes based on asymmetric cryptography in section 6 and 7. Finally, we conclude this paper by a comparison with other key distribution schemes in section 8.

2. MULTICAST SECURITY

The goal of multicast security is to assure that the source of the multicast stream and the group of multicast recipients communicate securely. This can be achieved through the authentication of the message origin by the recipients and through

confidentiality and integrity preventing disclosure and modification of the messages by any party other than the members of the multicast group. These services typically require the establishment of a security association between the source and the recipients of the multicast channel. The security association defines the set of cryptographic keys and algorithms used for each service. While authentication, confidentiality and integrity of messages in the multicast stream can be assured by classical network security mechanisms akin to unicast, the establishment of a security association for a multicast channel is inherently more complex than with unicast. In the unicast case, a security association is static in that the source, the recipient and the the data flow do not vary during the association. In a dynamic multicast group, a session is an ever evolving entity as recipients can be added to or removed from the recipient group through join and leave operations, respectively. Ideally, the keying material shared by the members of the multicast security association should be updated in order to fulfill the following conditions:

(1) When a user JOINS the group he should not have access to past keying material.
(2) When a user LEAVES the group he should not have access to future keying material.

Hence, some keying material must change each time the set of users in a multicast group changes. It should be noted that the above conditions apply for the highest security requirements and they can be relaxed for multicast applications with less critical security requirements.

Moreover, as a group gets larger, it is not acceptable to share the same keying material between all users of the group. The security of a *large* group should not depend on its weakest member(s). If the keying material of a user is intentionally or unintentionally exposed, the security of the group should not be compromised in that only a small fraction of the recipient group should be affected by the exposure.

2.1 Scalability

Naturally, while offering security services, the multicast spirit needs to be preserved: the amount of multicast data sent by the source should be independent of the group size. This also means that the cost in space and processing of the security services at each receiver should be constant, and therefore not correlated to the group size.

Suivo Mittra has described [Mittra 1997] two main scalability pitfalls in multicast security:

(1) The "one affects all" failure which occurs when the action of a member affects the whole group.
(2) The "one does not equal n" failure which occurs when the group cannot be treated as a whole but instead as a set of individuals with competing demands.

As noted in [Mittra 1997], JOIN and LEAVE procedures are candidates for exhibiting such failures, if they are not carefully designed. In a straightforward multicast security protocol where each member M_i of the group has an individual key K_i used for the distribution of the group key K, at the departure of a member, the security conditions introduced in section 2 would require a new key K' to be distributed to all remaining members. This simple scenario illustrates both scalability failures: the departure of one member affects the entire group through the key update

procedure and the remaining members of the group must be treated individually during the key update, since the new group key K' must be placed in a separate envelope encrypted under the key distribution key (K_i) of each remaining member. As highlighted in this example, multicast security requirements naturally call for solutions that conflict with scalability.

2.2 Containment

We can classify the main proposals for secure multicast in large dynamic groups in 3 categories:

(1) The IOLUS approach: [Mittra 1997].

(2) The KEYGRAPH approach: [Wong et al. 1998][Wallner et al. 1998][Canetti et al. 1999][Caronni et al. 1998][McGrew and Sherman 1998][Chang et al. 1999].

(3) The MARKS approach: [Briscoe 1999].

In IOLUS, the recipients are partitioned in subgroups attached to a local server. The set of servers is organized in a tree. The source sends data to the root of the tree and the data is decrypted and re-encrypted each time it goes through a server in the hierarchy until it finally reaches a recipient. Because each server uses a different symmetric encryption key, this introduces a dependency between the keying material used to access the multicast data and the location of the recipient in the network. An indirect benefit of this dependency is containment: if the symmetric key used by a recipient is compromised then it can only be exploited by an adversary in the same topological area in the network. The major drawback of IOLUS is that the intermediate elements access the clear-text multicast data because of the inherent requirement to decrypt and re-encrypt messages at the servers.

In the KEYGRAPH approach, there is a unique global key shared among all recipients in the group. A set of auxiliary keys is used to distinguish recipients. These auxiliary keys are used to selectively and efficiently send a new global key to the recipients in the group. Here, no provisions are made for containment: once this global key is compromised, it allows an adversary to access the multicast data anywhere in the network. Moreover, since the global key is updated each time a member leaves the group[1] this system introduces a "one affects all" scalability failure for leave operations. In the remainder of the paper, we will use the word KEYGRAPH as a generic name for all related work referenced above.

In the MARKS approach, a small set of keys is used to access the group during a limited amount of time. In other words, each set of keys represents a limited number of rights to access the multicast group. There is no containment in the MARKS approach and exposure of a set of keys allows anyone in the network to access the multicast group during the corresponding subscription. Recovering from a security exposure requires a complete re-initialization of the group.

Though we do not think that it's a good idea to let the intermediate elements access the clear-text multicast data, as in IOLUS, we believe that containment is a desirable property in large dynamic multicast groups.

[1] This requirement can be relaxed for join operations, see [Caronni et al. 1998].

2.3 Conflicting Requirements

Extending the security requirements of unicast with the ones due to scalability and group dynamics, the requirements of a security protocol for data confidentiality in a dynamic multicast group can be summarized as follows:

(1) Data confidentiality: the protocol should be immune to eavesdropping.
(2) JOIN and LEAVE security: a new (*resp. old*) member should not have access to past *(resp. future)* data exchanged by the group members.
(3) Containment: the compromise of one member should not cause the compromise of the entire group.
(4) Collusion resistance: a set of members which exchange their secrets cannot gain additional privileges.
(5) Processing scalability: the processing load supported by an individual component (be it the source, an intermediate forwarding component or a recipient) should be independent of the group size.
(6) Membership scalability: the actions performed by a member should not affect the group as a whole, as illustrated in the "one affects all" failure above.
(7) Groupwise scalability: the group should not require to be treated as a set of distinct individuals, as illustrated in the "one does not equal n" failure above.

At first glance these points seem to offer several contradictions. For example, points 3 and 6 call for the clustering of group members into different subgroups with different security parameters. However points 5 and 7 require the group to be treated as a whole. More generally, it's easy to see that the source and the recipients have opposite requirements. Scalability requires the source to consider the entire group as a single entity whereas security requires each recipient to be treated individually.

Many multicast or group security schemes have been proposed that all satisfy the first requirement. However they differ widely from one another on the remaining requirements: [Ballardie 1996] does not make provisions for containment nor join and leave security, [Steiner et al. 1996][Steiner et al. 1998][Burmester and Desmedt 1995] do not offer scalability in large groups. Only IOLUS, KEYGRAPH and MARKS approaches seem to address both security and scalability requirements in large dynamic groups.

2.4 Motivation for the Proposed Multicast Security Framework

The main motivation of the solution proposed in this paper is to solve the basic conflict between scalability and security akin to multicast security in order to come up with a solution that can scale up to large networks. We suggest that the conflict between scalability and security can be overcome by involving the intermediate components of the multicast communication in the security process.

Intermediate components, be they network nodes, routers, or application proxies, are inherent participants in the basic multicast transmission process. The key scalability factor in the basic multicast transmission schemes is the spread of the multicast routing and packet forwarding load over a network of intermediate nodes. Placing security mechanisms on existing intermediate components seems to be a natural extension of existing multicast protocols. Moreover, partitioning the cost of

security mechanisms over the intermediate components appears to be a good way of assuring scalability. When the multicast group grows, new intermediate components are added to support new group members and the cost of security mechanisms can still be equally distributed by placing the additional security processing load due to the new members on the new intermediate components.

The involvement of intermediate components in the security process is also a premise for meeting multicast security requirements. If the security mechanisms can be made dependent on the intermediate component in which they are implemented, group members attached to different intermediate components can be treated independently or with different keying material. In addition to its relationship with the group membership, the keying material can have a relationship with the topology of the multicast network. The keying material associated with each group member can thus be a function of the intermediate component to which the member is attached. This topological dependency assures the containment of security exposures: if some keying material belonging to a group member attached to an intermediate node is compromised, this keying material cannot be exploited by recipients attached to other intermediate nodes.

We introduce our solution in two steps. First, we define a general framework for multicast data confidentiality based on distributed mechanisms involving intermediate components and preserving the scalability and security properties. Then we propose actual solutions based on cryptographic functions that comply with the framework.

3. CRYPTOGRAPHIC FUNCTIONS

The building blocks we have chosen to use to design data confidentiality protocols over a multicast tree are called ***Cipher Sequences*** (CS). This section will give a formal definition of these sequences and associate them in trees. The resulting trees will then be used in further sections to describe our multicast security framework.

3.1 Cipher Sequences

Definition 1. Let $\mathcal{G} = \{g_i : \mathcal{N} \mapsto \mathcal{N}, i \in \mathcal{A}\}$ denote a set of permutations operating on a message space $\mathcal{N}$ and indexed in $\mathcal{A}$. $\mathcal{G}$ will be called a ***Cipher Group*** if it satisfies the following properties:

(i) $(\mathcal{G}, \circ)$ forms a group through the composition operation $\circ$. Specifically, if $g', g'' \in \mathcal{G}$ then $g'' \circ g' \in \mathcal{G}$ (closure). Moreover, $\forall g \in \mathcal{G}$, $\exists g^{-1}$ such that $g^{-1} \circ g = Id = g \circ g^{-1}$ (inverse).

(ii) $(\mathcal{G}, \circ)$ is indexable through composition: there exists a polynomial time algorithm $Comp : \mathcal{A}^2 \mapsto \mathcal{A}$, which given an index pair $(i, j) \in \mathcal{A}^2$, computes $k = Comp(i, j) \in \mathcal{A}$ such that $g_{(k)} = g_{(j)} \circ g_{(i)}$

Moreover:

—$\mathcal{G}$ will be called a ***Symmetric Cipher Group*** if the knowledge of $g \in \mathcal{G}$ allows the computation of g^{-1} in polynomial time.

—$\mathcal{G}$ will be called an ***Asymmetric Cipher Group*** if the computation of g^{-1} from $g \in \mathcal{G}$ is computationally infeasible without the knowledge of a trapdoor.

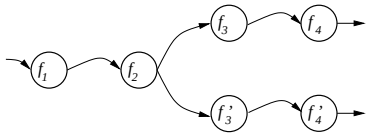

Fig. 1. Two $CS_{\mathcal{G}}$'s mapped over a tree.

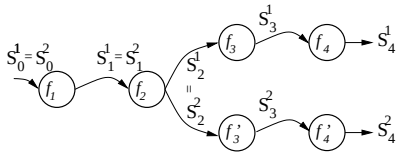

Fig. 2. Two instantiated $CS_{\mathcal{G}}$'s in a tree.

Definition 2. Let $\mathcal{F}$ be a random sequence $(f_{0<i\leq n})$ of n elements in a cipher group $\mathcal{G}$. By definition, there exists a function $g \in \mathcal{G}$ such that $g = (f_n \circ f_{n-1} \circ \cdots \circ f_1)^{-1}$. We will call this function the ***Reversing Function*** of $\mathcal{F}$ and denote it as $h_{\mathcal{F}}$ or, more simply h. The sequence $\mathcal{F}$ will be called a ***Cipher Sequence*** in $\mathcal{G}$, or $\boldsymbol{CS}_{\mathcal{G}}$.

Consequently, if $\mathcal{G}$ is a symmetric cipher group, we will say that $\mathcal{F}$ is a ***Symmetric Cipher Sequence*** in $\mathcal{G}$ or $\boldsymbol{SCS}_{\mathcal{G}}$. Similarly, if $\mathcal{G}$ is a asymmetric cipher group, we will say that $\mathcal{F}$ is an ***Asymmetric Cipher Sequence*** in $\mathcal{G}$ or $\boldsymbol{ACS}_{\mathcal{G}}$.

Definition 3. Let $\mathcal{F}$ define a Cipher sequence of n functions in the cipher group $\mathcal{G}$. From this sequence $\mathcal{F}$ we can derive a sequence $(S_{0\leq i\leq n})$ of elements in $\mathcal{N}$ as follows:

$S_i = f_i(S_{i-1})$, for $n \geq i > 0$.

S_0, the initial value of the sequence.

We will call $(S_{0\leq i\leq n})$ an ***instance*** of $\mathcal{F}$ and we will denote it as $\mathcal{F}(S_0)$.

3.2 $\mathrm{CS}_{\mathcal{G}}$'s over a General Tree

A tree can map a family of $CS_{\mathcal{G}}$'s which have terms that differ only after a certain rank, greater than 1. For example, if $\mathcal{F}_1$ (resp. $\mathcal{F}_2$) is a $CS_{\mathcal{G}}$ defined as $\mathcal{F}_1 = \{f_1, f_2, f_3, f_4\}$ (resp. $\mathcal{F}_2 = \{f_1, f_2, f'_3, f'_4\}$), a simple tree that maps $\mathcal{F}_1$ and $\mathcal{F}_2$ can be constructed as in figure 1. This tree illustrates the fact that $\mathcal{F}_1$ and $\mathcal{F}_2$ differ after rank 2.

This property can be extended from 2 to n different $CS_{\mathcal{G}}$'s, $\mathcal{F}_{0<i\leq n}$, with a tree that branches each time at least two sequences differ. If we instantiate each $\mathcal{F}_k$ in the tree with the same value S_0 we can label the edges of the tree with the elements S^i_k of each $\mathcal{F}_i(S_0) = \left(S^i_{0<k\leq n_i}\right)$. We give a small example in figure 2. Note that all the instantiations $\mathcal{F}_i(S_0)$ share at least two elements, namely S_0 and S_1. These two values are the input and the output, respectively, of the root node of the instantiated tree.

Remark 1. We implicitly require that for all $\mathcal{F}_i$ and $\mathcal{F}_j$ where $i \neq j$, there exists a function f such that $f \in \mathcal{F}_i$ and $f \notin \mathcal{F}_j$. In our trees, this translates to the fact that the number of leaves is equal to the number of cipher sequences mapped over the tree.

4. MULTICAST SECURITY FRAMEWORK

The proposed multicast security framework consists of a model that is an abstract definition of the components involved in the security mechanisms and the relationship between them which is an application of the functions we defined in section 3.

4.1 Model

In the abstract definition of the framework, the components of the multicast security framework form a ***tree***. The ***root*** of the tree is the multicast source and the members of the multicast group form the ***leaves*** of the tree. The intermediate nodes of the tree -referred to as ***inner nodes***- correspond to the intermediate components of the multicast communication. Like the multicast scheme itself, inner nodes can be implemented at the application layer or at the network layer. In the case of application layer multicast, inner nodes can be application proxies, such as those in a hierarchical web caching structure. In the case of network layer multicast, inner nodes can be intelligent routers capable of performing security operations in addition to multicast packet forwarding functions.

In further abstraction, each leaf of the tree will represent the set of group members attached to the same terminal inner node. In the application layer case, a leaf will delimit a sub-group of members attached to a proxy. In the network layer case, a leaf will delimit a sub-network of recipient stations attached to a router. Hence a leaf will refer to a set of multicast group members with a common attachment node in the tree.

If a set of users represented by a leaf becomes too large, the leaf can easily be subdivided into several "sub-leaves" by adding new inner nodes. Hence the leaf size in terms of the group members it represents is not a scalability issue for algorithms that treat a leaf as a single entity.

4.2 $CS_{\mathcal{G}}$'s over a Multicast Tree

Next, we turn to multicast by applying the previous concept of $CS_{\mathcal{G}}$ from section 3 over a tree as a means of performing secret transforms in multicast communications. Let S_0 be an encoding of the information to be transmitted over the multicast channel by the source under confidentiality.

As part of the setup for a series of secure multicast transmissions, each inner node N_i is assigned a secret function $f_{i>0} \in \mathcal{G}$. We require that each inner node is capable of performing this function $f_{i>0}$ as defined in the previous section. During secure multicast transmission, upon receipt of multicast data S_j from its parent node N_j, node N_i computes $f_i(S_j)$, and forwards the resulting value S_i as the secure multicast data to the child inner nodes or the leaves.

Assuming $\mathcal{F}_i(S_0)$ is an instance of a $CS_{\mathcal{G}}$ mapped over a path from the root to a leaf on the multicast tree, the leaf will eventually receive S_n^i, which is the final term of $\mathcal{F}_i(S_0)$. The leaves in the multicast tree bear a special role in that they are able to recover the original message S_0. Each leaf is assigned a reversing function h^i that allows it to compute $S_0^i = S_0$ from $S_{n_i}^i$ since $S_0 = h^i(S_{n_i}^i)$. The leaves don't use any other function in $\mathcal{G}$.

The distribution of the secret f_i functions to the inner nodes and the reversing functions to the leaves can be assured by a central server using classical unicast

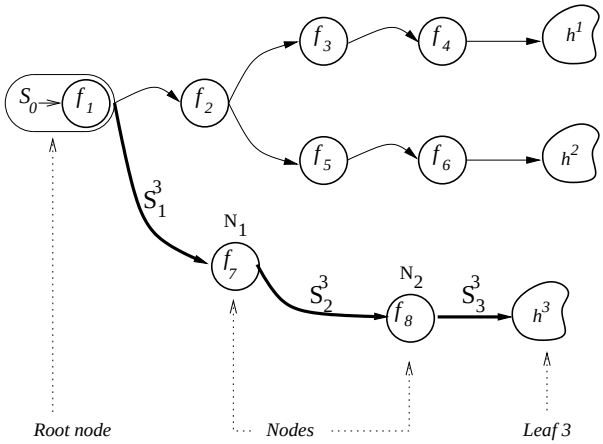

Fig. 3. A simple 3 $CS_{\mathcal{G}}$ tree.

security mechanisms. Because of the structure of the algorithm, the central server will need to have a precise image of the tree structure. This doesn't mean, however, that the functionality of this server cannot be distributed over several network entities.

Working example . Figure 3 depicts a simple tree with three $CS_{\mathcal{F}}$'s. Looking at the path from the root to leaf 3 on figure 3, we have:

—The root computes $f_1(S_0)$ and sends the result to its children inner nodes.
—N_1 receives $S_1^3 = f_1(S_0)$, computes and sends $f_7(S_1^3)$ to N_2.
—N_2 receives $S_2^3 = f_7(S_1^3)$ and sends $f_8(S_2^3)$ to leaf 3.
—Leaf 3 receives $S_3^3 = f_8(S_2^3)$ and recovers the original multicast data by computing $S_0 = h^3\left(S_3^3\right)$.

4.2.1 *The Join Procedure.* When a user joins a group by contacting a node, two situations can arise:

(1) a leaf (sub-group) attached to this node already exists.
(2) there is no leaf attached to this node prior to the current join operation.

In the former situation, a cascade sequence $\mathcal{F} = (f_{0 \leq i \leq n})$ is already mapped between the source and the members in the existing leaf. The last inner node on the path which holds function f_n will be assigned a new value $\widetilde{f}_n$, updating the last transformation in the sequence. Hence, the corresponding new reversing function $\widetilde{h}$ will be distributed to all the members in the leaf including the new member.

In the example of figure 4 where C wishes to join the leaf including existing members A and B, the join operation will perform as follows:

(1) $\widetilde{f}_8$ will be substituted to f_8 in the last inner node.
(2) $\widetilde{h}^3$ will be sent to A, B, C.

If M is the upper bound on the number of members in a leaf, a join operation requires the exchange of the following messages:

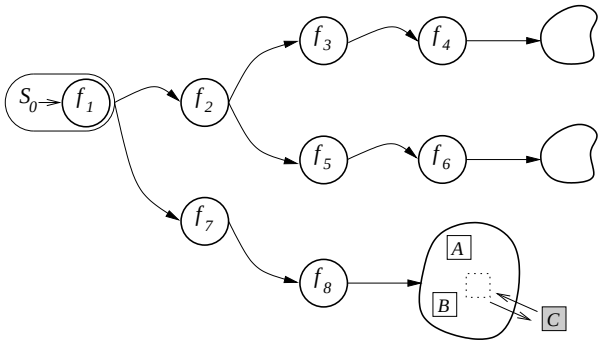

Fig. 4. User C joins/leaves.

—1 message sent to update the value in the last inner node on the path,

—at most $M - 1$ messages sent to the current members in the leaf,

—1 message sent to the new member.

A join operation thus requires at most $M + 1$ message exchanges.

In fact, it's possible to reduce the number of messages to 3, by slightly changing the order of operations in the join procedure. Instead of changing the value f_n in the node right away, it's possible to use the secure sequence to vehicle the new $\widetilde{h}$ function to the current members in the leaf, thus reducing the update to one message (versus an upper bound of $M - 1$ messages). Then the value f_n in the node can be changed and the new $\widetilde{h}$ function transmitted to the joining member. However this approach has a drawback: it creates a chain between the different values of $\widetilde{h}$ which potentially weakens the security of the scheme. Unless the cost of individually sending a message to each member in the leaf is more important than the security of the group, such an option should be avoided.

In the second case, the authority which receives a join request has to figure out the path from the new member to the closest inner node in the active tree. The path establishment method used in this case depends on the layer (application/network) at which the multicast security scheme is implemented. A similar decision has to be taken by IP multicast routing algorithms when a new router needs to be included in a multicast routing tree. Once the path to the new member is selected, the authority will assign values to the newly added inner nodes on the path, thus extending the $CS_{\mathcal{G}}$ mapping. Finally the new member will receive the h function needed to recover the original multicast data in the newly created $CS_{\mathcal{G}}$. Hence, he's the only member of the new leaf in the tree.

The number of messages exchanged here depends on the algorithm used to set the path between the new member and the tree. Consequently, as stated in section 2.4, this security framework would be a natural extension of multicast routing schemes. The number of messages exchanged here to create a new leaf can be assumed to be proportional to the number of messages exchanged by the multicast routing protocols when adding a new element in the multicast tree.

In many cases, it will be possible to perform the node setup ahead of time, leaving only the h function to be distributed when the member effectively joins.

The authority that manages the group does not need to be the root itself and its functionality can be distributed in a tree hierarchy, where each sub-authority manages a multicast subtree.

4.2.2 *The Leave Procedure.* The leave procedure is similar to the join procedure. When a user leaves a leaf in the tree, the function in the terminal inner node is changed from f_n to $\widetilde{f}_n$ and the new reversing function $\widetilde{h}$ is distributed to the remaining inner nodes in the tree. In effect, the associated $CS_{\mathcal{G}}$ has its last term changed.

4.3 Evaluation of the Framework

The previous discussion has focused on the use of $CS_{\mathcal{G}}$ to achieve data confidentiality over a multicast tree. This section will show how the $CS_{\mathcal{G}}$ construct meets the requirements established in 2.3, assuming that the intermediate nodes are trusted and secure. The implications of node compromise will be discussed in the next section.

Data Confidentiality. The security of the scheme depends on the strength of the ciphers that are used to implement it. The functions in a cipher group should be viewed as building blocks for confidentiality services. A possible approach, which is beyond the scope of this work, would be to consider these function as pseudo-random permutations. (As a side effect, it would require the cipher groups to be of large order: a small order would provide a method to build a polynomial time distinguisher between cipher group elements and pseudo-random permutations).

JOIN and LEAVE Security. A new member joining a leaf gets a new reversing function $\widetilde{h}$ that cannot be used to recover the old reversing function h . As a consequence, past data is not accessible to a new member. Similarly, a former member using an old reversing function cannot access data that is transmitted subsequently to its departure.

Containment. Because of the topological dependency introduced by the model, the reversing function h used in a leaf of the tree will be useless outside that leaf. An intruder will only benefit from an attack if he is located in the same leaf as the victim. This greatly reduces the impact of member compromise.

Collusion Resistance. Let $(G, \circ)$ define a group. For a random $(a_0, a_1, ..., a_n) \in G^{n+1}$, define $b_i = a_0 \circ a_i$ for all $1 \leq i \leq n$. An adversary observing $(b_1, b_2, ..., b_n)$ does not gain any knowledge about a_0.

PROOF. For all $a_0 \in G$ we can write each $b_i \in G$ as $b_i = a_0 \circ a_i$ where $a_i = (a_0^{-1} \circ b_i)$. Thus, a_0 can be any of $|G|$ possible values. □

This means that in a fully balanced tree of depth 1, members who collude and exchange their reversing functions gain no knowledge of the intermediate functions. This can be easily generalized to any tree, provided that no node is both an inner node and a leaf, as already highlighted in remark 1 of section 3.

Processing Scalability. The amount of processing per component is independent of the group size. First, in our framework, the size of messages transmitted by a node (be it the source or an intermediate component) does not depend on the

number of group members but it depends only on the size of the original secret message. Second, the number of messages transmitted by a node does not depend on the number of group members but it depends only on the number of child nodes attached to this node.

Membership Scalability. There are three basic actions a group member can perform, namely join, leave and receive data. The model is designed so that none of these actions affects the whole group. In fact these actions have an impact that is limited to the leaf containing the member performing these actions as shown in section 4.2.1. The "one affect all" type failure never appears.

Groupwise Scalability. The "1 does not equal n" type of failure never appears over the group as a whole, instead, it is confined to the leaves in which join or leave operations occur. Since the leaves have a maximum size, this is not a scalability issue. All other operations, including re-key, address the group as a whole.

4.4 Node Compromise.

The previous section assumed that the inner nodes of the tree were completely secure. This has to be true for the root node of the tree but it might not be possible to make such an assumption about the intermediate nodes in the network. Hence the following section will focus on the impact of intermediate node compromise. Two type of attacks that derive from node compromise are highlighted in this section: unauthorized membership extension and node compromise by external users. Unauthorized membership extension happens when a former member of the secure group is able to maintain access to the data even though he has not received the new reversing function. Node compromise by external users more generally describes unauthorized access to the group by users that never became group members.

4.4.1 *Unauthorized Membership Extension.* If a member *Eve* in a leaf controls the last inner node on the path from the source to the leaf i, she can intercept changes in the last element of the $CS_{\mathcal{G}}$.

Let N be the last inner node on the path from the root to leaf i of the tree and f the secret function held by N. N receives S^i_{j-1} from its parent node and sends $S^i_j = f(S^i_{j-1})$ to the leaf elements which will use a h^i reversing function to recover S_0. If the group membership manager decides that *Eve* must leave the group, the function f in N will be changed to a new function $\widetilde{f}$ and the corresponding reversing function $\widetilde{h}^i$ will be send to all leaf members except *Eve*.

Despite its formal exclusion from the group, *Eve* can ignore the change in the router and compute S_0 from S^i_{j-1} using f and the old reversing function h^i obtained through the compromise of node N, simulating the older sequence, where $S_0 = h^i\left(f(S^i_{j-1})\right)$.

This attack succeeds whatever the nature of the sequence, APS_f or SPS_f, but requires several conditions to be met:

(1) *Eve* should be a former member of the group.

(2) *Eve* should be able to access the secret functions f and $\widetilde{f}$ held by node N.

(3) *Eve* should have access to the data transmitted to node N by its parent node

(i.e. S_{j-1}).

Moreover, updates of f_i functions in inner nodes at a higher level will limit the scope of this attack because the resulting reversing functions cannot be retrieved based on the information gathered in a leaf or from the compromise of the last inner node.

A $SCS_{\mathcal{G}}$ specific attack. Condition 3 described in the previous paragraph is not required if the sequence is an instance of a $SCS_{\mathcal{G}}$. To work around condition 3, the intruder first computes:

$$\widetilde{f}^{-1} \text{ from } \widetilde{f}$$

thanks to the fact that the sequence is symmetric.
Now, using $\widetilde{f}^{-1}$ and the new sequence value $\widetilde{S}_j^i$ received in the leaf, the former member computes:

$$S_{j-1}^i = \widetilde{f}^{-1}(\widetilde{S}_j^i)$$

Next, the intruder uses the value f obtained through the compromise of N to compute:

$$S_j^i = f(S_{j-1}^i)$$

Finally, using the old reversing function h^i and applying it to S_j^i, we have:

$$S_0 = h^i(S_j^i)$$

where S_0 is the original multicast data.

This attack doesn't apply in an APS_f tree because by definition $\widetilde{h}^i$ cannot be derived from $\widetilde{f}$.

4.4.2 *Node Compromise by External Users.* If the intruder *Eve* is not even a former member of the group, an attack is still possible if the sequence is symmetric provided that:

(1) *Eve* has access to the value of the reversing function used by a legitimate member .
(2) *Eve* controls all the inner nodes on the path between her[2]and the legitimate member except the first common ancestor they have in the tree.

If these conditions are met, *Eve* will be able to forge a reversing function he can use to access the group.

Instead of a lengthy formal discussion, we chose to illustrate the attack with the example scenario depicted on figure 5, where the malicious user *Eve* gets multicast data S_3^2 from node N_5. If *Eve* knows h^3 from a compromised user and $\{f_2, f_5, f_7, f_8\}$, she can compute:

$$S_2^2 = f_5^{-1}(S_3^2)$$

because f_5^{-1} can be derived from f_5. Similarly,

$$S_1^3 = S_1^2 = f_2^{-1}(S_2^2)$$

because f_2^{-1} can be computed from f_2. Then

$$S_2^3 = f_{a_7}(S_1^3)$$

[2] *Eve* does not have to be in a real leaf, she can simply intercept traffic somewhere in the tree.

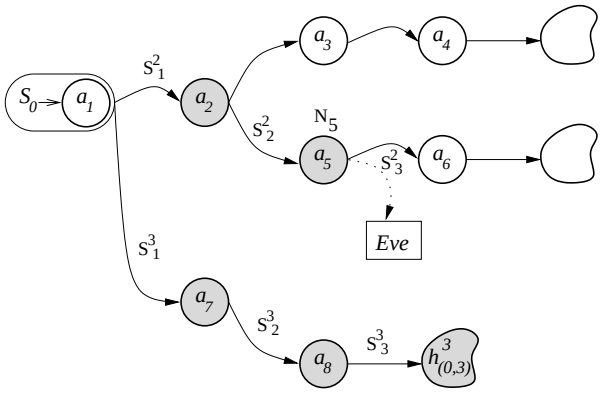

Fig. 5. Multiple node compromise attack.

and

$$S_3^3 = f_{a_8}(S_1^3)$$

yielding

$$S_0 = h^3(S_3^3)$$

Again, this attack doesn't apply to an $ACS_{\mathcal{G}}$ based tree because reversing functions associated with an $ACS_{\mathcal{G}}$ cannot be derived from the parameters used in the intermediate nodes.

4.4.3 *Node Compromise Summary.* The distinction between an $ACS_{\mathcal{G}}$ and a $SCS_{\mathcal{G}}$ is germane to node compromise scenarios. Unlike IOLUS, when using $ACS_{\mathcal{G}}$'s our framework is immune to node compromise by external users. The framework does not however dictate the choice of an $ACS_{\mathcal{G}}$ over $SCS_{\mathcal{G}}$ as one could expect because $SCS_{\mathcal{G}}$ are likely to be easier to design than $ACS_{\mathcal{G}}$.

It should be noted that the security containment property is also effective in case of node compromise. Hence, previously described node compromise scenarios don't allow the intruder to provide unauthorized access to just any other recipient in the network but only to those attached to the same leaf.

This section concludes the formal presentation of our secure multicast framework. The next sections present two implementations of this framework based on extensions of public key cryptographic schemes. The first scheme is an $SCS_{\mathcal{G}}$ and will therefore lend itself to further description of a concrete node compromise scenario.

5. KEY DISTRIBUTION

Depending on the performance of functions in $\mathcal{G}$ our framework can be used either for bulk data confidentiality or only for key distribution. Current symmetric cryptographic systems provide sufficient encryption speed but they don't exhibit the mathematical properties[Kaliski et al. 1985][Campbell and Wiener 1993] required to create a $CS_{\mathcal{G}}$. On the other hand asymmetric cryptography offers suitable properties to build a solution compliant with the framework but it doesn't offer yet the necessary performance for bulk data confidentiality. Consequently, the next

two sections will describe the framework based on asymmetric cryptography for multicast key distribution. The first scheme, derived from the ElGamal encryption algorithm, allows the creation of a $SCS_{\mathcal{G}}$ key distribution tree, whereas the second scheme, based on RSA, effectively creates an $ACS_{\mathcal{G}}$ key distribution tree.

Using a $CS_{\mathcal{G}}$ tree, the source can distribute a secret key k by instantiating the tree with $S_0 = k$ (or otherwise a function of k). The data confidentiality mechanism of the secure multicast framework will allow to securely transmit k to the members of the group. Unlike the reversing function that is different in each leaf, k is shared among all members of the group so the exposure of k affects the group as a whole. However, unlike the reversing function that enables each member to access the multicast group, the shared key k is a short term value that can be frequently updated by the source using the secure multicast framework. Using a secure key generation technique, it can be assured that subsequent values of k are independent.

6. KEY DISTRIBUTION USING ELGAMAL.

The ElGamal cryptosystem can be extended to create a $SCS_{\mathcal{G}}$.

PROPOSITION 1. *Let p be a large random prime. Define $\mathcal{N}$ as the set of all primitive elements of $\mathbb{Z}_p^*$. The set $\mathcal{G} = \left\{ f_a(x) = x^a \bmod p; a \in \mathbb{Z}^*_{(p-1)} \right\}$ forms a symmetric cipher group for the message space $\mathcal{N}$.*

Properties (i) and (ii) in definition 1 are easily verified. Given an index $i \in \mathbb{Z}^*_{(p-1)}$, we can compute $f_{(i)}(x) = x^i \bmod p$ as well as $f^{-1}_{(i)}(x) = x^{1/i} \bmod p$. Given $(i,j) \in \mathbb{Z}^*_{(p-1)}$, we can compute $k \in \mathbb{Z}^*_{(p-1)}$ such that $f_{(k)}(x) = f_{(i)} \circ f_{(j)}(x) = f_{(j)} \circ f_{(i)}(x) = x^{i.j} \bmod p = x^k \bmod p$.

By definition, all the elements in $\mathcal{N}$ are of order $p-1$. Reasonable assumptions about the security of these functions are:

(1) For random primitive elements $x_i \in \mathcal{N}$ and a random $f \in \mathcal{G}$, an adversary has a negligible chance to recover any x_i without knowing f.
(2) For random primitive elements $x_i \in \mathcal{N}$ and a random $f \in \mathcal{G}$, an adversary has a negligible chance to recover f by observing pairs of the form $(x_i; f(x_i))$.

6.1 Setup

The source chooses a generator g of the cyclic group $\mathbb{Z}_p^*$ and a secret random value r in $\mathbb{Z}^*_{(p-1)}$. The inner nodes and the root are assigned f_{a_i} values in $\mathcal{G}$ to form a $SCS_{\mathcal{G}}$ tree. The tree is instantiated with $S_0 = g^r \bmod p$.

Let $\{S_{i_k > 0}\}$ denote the instantiated sequence elements. The reversing function distributed to the leaves is defined as:

$$h^k(x) = x^{(a_{i_1}.a_{i_2}.a_{i_3}\ldots a_{i_{n_k}})^{-1}} \bmod p$$

6.2 Key Distribution

The source wishing to distribute a key K sends the following initial data to its children in the tree:

$$S_1 = (S_0)^{a_1} \bmod p$$

$$T = K \oplus S_0$$

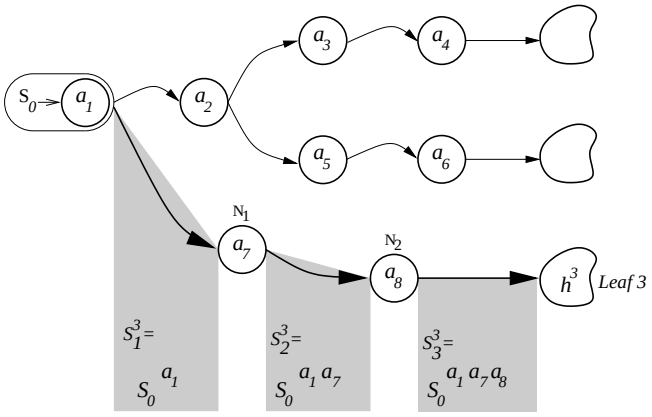

Fig. 6. A discrete log tree.

Remark 2. Issues such as a proper padding of K, have been omitted here for the sake of simplicity. Depending on the particular security requirements, we can substitute T with a better encoding of K as suggested, for example in [Abdalla et al. 1998].

The intermediate elements in the tree perform $f_{a_{i_k}}$ on their input S_{i_k-1} and send the resulting S_{i_k} value to their children, along with T, where:

$$S_{i_k} = f_{a_{i_k}}(S_{(i_k-1)}) = \left(S_{(i_k-1)}\right)^{a_{i_k}} \ mod\, p$$

An example of this scheme is illustrated on the path from the root to the leaf 3 of the tree as depicted in figure 6:

—The source sends $S_1^3 = (S_0)^{a_1} \ mod\, p$ and $T = K \oplus S_0$ to its children.
—N_1 receives (S_1^3, T) and sends $S_2^3 = (S_0)^{a_1 a_7} mod p$ and $T = K \oplus S_0$ to its children.
—N_2 receives (S_2^3, T) and sends $S_3^3 = (S_0)^{a_1 a_7 a_8} \ mod\, p$ and $T = K \oplus S_0$ to leaf 3.

6.2.1 *Decryption.* The decryption process is straightforward, the reverse function h is simply applied to the received value, and the result is used to extract K from T:

$$h(S_{i_k}) = S_0 \ mod\, p$$

$$K = T \oplus S_0$$

Recalling the previous example, where $h^3(x) = x^{\frac{1}{a_1 a_7 a_8}} \ mod\, p$, the value of K is computed simply:

$$K = T \oplus S_0$$

where

$$S_0 = h^3\left(S_3^3\right) = \left((S_0)^{a_1 a_7 a_8}\right)^{\frac{1}{a_1 a_7 a_8}} \ mod\, p$$

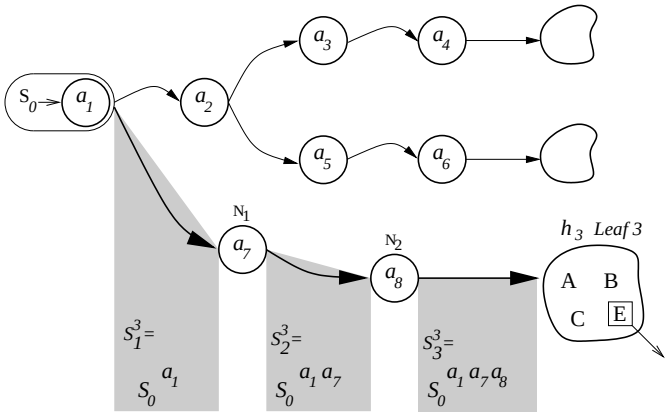

Fig. 7. Node Compromise.

6.2.2 *The Next Key.* Sending the next key $\widetilde{K}$ only requires $S_0 = g^r \ mod\, p$ to be updated as $\widetilde{S}_0 = g^{\widetilde{r}} \ mod\, p$ where $\widetilde{r}$ is a random element in $\mathbb{Z}^*_{(p-1)}$.

6.3 Node Compromise and Member Collusion

Many of the requirements established in section 2.3 are naturally fulfilled by implementing the framework as described above. However member collusion and node compromise need to be considered on a per-algorithm basis.

6.3.1 *Node Compromise.* The previously described sequence is clearly a $SCS_{\mathcal{G}}$ because the reversing functions can be computed with the knowledge of the secret parameters in the inner nodes. Hence, compromise of the inner nodes offers some potential for unauthorized membership extension as described in 4.4.

Figure 7 illustrates the node compromise scenario. The hypothesis here will be that a malicious member E of leaf 3 wishes to maintain membership in the group using the information of the terminal inner node N_2 he has compromised.

In a normal scenario where node compromise is not taken into account, in a leaf consisting of members $\{A, B, C, E\}$, when E leaves, the following actions take place:

—In N_2, f_{a_8} is changed to $\widetilde{f}_{a_8}$.

—The newly computed reverse function $\widetilde{h}^3$ is sent to $\{A, B, C\}$ but not E.

Once the leave procedure is complete, E cannot access further keys distributed to $\{A, B, C\}$.

However, in the case of inner node compromise, if E controls the last inner node, he can monitor the change from f_{a_8} to $\widetilde{f}_{a_8}$. E can then derive $\widetilde{h}^3$ from h^3, because if $h^3(x) = x^{(a_0 a_1 a_7 a_8)^{-1}} \ mod\, p$ then:

$$\widetilde{h}^3(x) = x^{\frac{1}{a_0 a_1 a_7 a_8} \times \frac{a_8}{\widetilde{a}_8}} \ mod\, p$$

In summary, even if E doesn't receive the new reversing function, he will be able to compute it and thus access the keys distributed subsequently to the leave operation.

This attack can be extended to allow a malicious user to derive a reversing function from another one even if the reverse function comes from another leaf. It requires the attacker to compromise nearly all the inner nodes on the graph between him and the compromised member.

Figure 5 will serve as an example where user *Eve* -not a member of the group- listens to traffic coming out of N_5. The malicious user is assumed to know the following node functions $\{f_{a_2}, f_{a_5}, f_{a_7}, f_{a_8}\}$ as well as h^3 from a compromised member in leaf 3. With these conditions together, *Eve* can compute a new local reverse function h^2 from h^3 thus violating the confinement property of the model:

$$h_3^3(x) = x^{\frac{1}{a_1 a_7 a_8}} \; mod\, p$$

which allows to compute:

$$h^2(x) = x^{\frac{1}{a_1 a_2 a_5}} \; mod\, p = \left(h_3^3(x)\right)^{\frac{a_7 a_8}{a_2 a_5}} \; mod\, p$$

This second attack assumes that the inner nodes are easy to compromise, and the first one makes strong assumptions about the compromise power of the attacker. While these attacks on $SCS_{\mathcal{G}}$'s might be considered hard to carry out in most cases, the very possibility of such attacks motivated further analysis to study a second and stronger construction based on $ACS_{\mathcal{G}}$'s, as described in section 7.

7. KEY DISTRIBUTION USING RSA.

Extending RSA to use multiple keys as in [Harn and Kiesler 1989] allows the creation of an $ACS_{\mathcal{G}}$ scheme.

PROPOSITION 2. *Let $n = pq$ be the product of two carefully chosen large primes, as in the RSA cryptosystem. Define $\mathcal{A} = \mathcal{Z}^*_{\varphi(n)}$ and $\mathcal{N} = \mathcal{Z}_n$. The set $\mathcal{G} = \{f_a(x) = x^a \, mod\, n; a \in \mathcal{A}\}$ is a cipher group for messages in $\mathcal{N}$.*

This construction is quite similar to the one from the previous section, except that we are working with a composite modulus and a different message space.

7.1 Setup

The setup is even simpler here than in the ElGamal case. Each inner node in the tree is assigned a value $a_{i>1}$ and the root uses a_1 where $\gcd(a_{i\geq 0}, \varphi(n)) = 1$. This assures that the product A of any subset of these a_i values also verifies $\gcd(A, \varphi(n)) = 1$. The multiplicative inverse B of A defined as $AB \equiv 1 \, (mod\, \varphi(n))$ can be computed using the extended Euclidean algorithm.

Let $\{a_{k_i>0}\}$ denote the set of parameters used in the inner nodes between the source and leaf k, plus $a_{k_1} = a_1$ in the root. The reversing function distributed to the leaves is defined as:

$$h^k(x) = x^{D_k} \; mod\, n$$

where,

$$(a_{k_1}.a_{k_2}...a_{k_m}).D_k \equiv 1 \, (mod\, \varphi(n))$$

Like the basic RSA algorithm, the asymmetric property of this scheme relies on the difficulty of computing D_k from the reversing function without the knowledge of $\varphi(n)$ (which currently seems to be only derivable from the factors of $n = pq$).

7.2 Key Distribution

The source wishing to distribute a key K, sends the following value to its children in the tree:

$$S_1 = K^{a_1} = (S_0)^{a_1} \ (mod\ n)$$

Remark 3. Here we have $S_0 = K$, and for simplicity we have omitted issues such as padding or semantic security. A proper (and probabilistic) encoding of K is suggested in [RSA Security Inc. 1999] or better in [Bellare and Rogaway 1995].

Each inner node N_i in the secure multicast tree processes the S_{i-1} value received from its parent node and sends S_i to its children inner nodes where:

$$S_i = f_{a_i}(S_{i-1}) = (S_{i-1})^{a_i} \ (mod\ n)$$

Recalling figure 7 while assuming an RSA like $ACS_{\mathcal{G}}$ sets the following scenario on the path from the root to leaf 3 of the tree:

—The root send $S_1^3 = (S_0)^{a_1}\ mod\ n$ to its children.
—N_1 receives S_1^3 and sends $S_2^3 = (S_0)^{a_1 a_7}\ mod\ n$ to its children.
—N_2 receives S_2^3 and sends $S_3^3 = (S_0)^{a_1 a_7 a_8}\ mod\ n$ to leaf 3.

7.2.1 *Decryption.* The decryption process is also simpler than in the ElGamal case. The decryption function h is applied to the received value in the leaf to recover K. For example, on figure 5:

$$K = S_0 = h^3(S_3^3) = ((S_0)^{a_1 a_7 a_8})^{D_3}\ mod\ n$$

assuming

$$a_1 a_7 a_8 . D_3 \equiv 1 \ (mod\ \varphi(n))$$

7.2.2 *The Next Key..* Sending a new key $\widetilde{K}$ only requires S_0 to be changed in the preceding description. Nothing else needs to be done.

7.3 Node Compromise

The node compromise attack previously described in section 6 regarding the discrete log case does not apply here essentially because the RSA-based sequences are asymmetric: to compute the inverse of any function in $\{f_{a_1}, f_{a_2}..., f_{a_k}\}$, the knowledge of the intermediate parameters $\{a_1, a_2..., a_k\}$ wouldn't be sufficient as the knowledge of $\varphi(n)$ is also required. However the node compromise attack based on membership extension as depicted in section 4.4 is still possible with the RSA-based scheme.

8. RELATED WORK

Other papers have presented schemes that address some of the requirements highlighted in section 2.3. However, only the IOLUS, the KEYGRAPH and the MARKS approaches seem to address both scalability and security. Hence we will focus our comparison on those three approaches, which differ from ours in mainly three areas: scalability, containment and trust. Moreover we will look at the particular implication of using our scheme for key distribution.

Scalability. We believe that MARKS is the most scalable scheme since the leave operation introduces no side effects. The tradeoff to this enficiency is the difficulty of revoking a member's rights. Containment oriented schemes such as IOLUS and ours, come next since the side effects are limited to a small subgroup. KEYGRAPHS require a global update each time a member leaves the group. Current research about KEYGRAPHS aims at reducing the cost of this operation through various optimizations.

Trust. Although our solution uses intermediate components, it has a major difference with IOLUS: our framework puts limited trust in the intermediate components, whereas in IOLUS each intermediate component has access to the multicast data. This problem does not appear in KEYGRAPHS or MARKS since no intermediate elements are involved.

Containment. In terms of containment, our scheme is equivalent to IOLUS, where each subgroup uses a different key to access the multicast data. On the other hand KEYGRAPS and MARKS do not address containment issues even though they use a computational tree structure. In those schemes, the keys held by any user can be used to access the multicast group regardless of the access location in the network and all users are equivalently trusted with the security parameters of the group.

Key distribution. Even though we offer higher security in terms of trust and containment, this comes with a cost. Indeed, IOLUS, KEYGRAPHS and MARKS have a clear advantage over our scheme in terms of performance. This has lead us to consider our scheme for key distribution and not bulk data encryption. In that respect the framework is used to distribute a short term data encryption key k. As this short term key is common to all recipients, it may look as our scheme looses its containment advantage over KEYGRAPHS or MARKS. However, the short term key can be frequently updated and its disclosure does not provide a means of long term group access to intruders. This is because in our scheme the group membership is represented by the long term reversing functions that are different in each leaf of the multicast tree as opposed to the shared secret group membership key(s) of KEYGRAPHS and MARKS.

Conclusion

This paper has presented a framework designed to support data confidentiality in a large dynamic multicast group. The framework meets a set of requirements wider than the previous work. While covering scalability, the new concept of containment was introduced as we believe the latter is a key requirement in very large groups.

The introduction of Cipher Sequences, or $CS_{\mathcal{G}}$, provides a formal but yet practical description of the framework elements, with a voluntary distinction between symmetric and asymmetric behaviors. The mapping of these sequences over a multicast tree is the core mechanism that allows this framework to meet the previously described requirements.

Two key distribution schemes have been presented as implementations of the framework. Further detailed studies of the those two schemes, that would each deserve a complete article, will be necessary before a concrete implementation. Nevertheless these two schemes have served as a proof of concept for the framework

and they have allowed us to discuss the implication of various node compromise scenarios, as the possibility of node compromise cannot be neglected in a large multicast network.

The next major step would be the design of efficient functions that could be used to build $CS_{\mathcal{G}}$'s that operate on bulk data, in order to fully capitalize on this framework. Beyond just multicast, such functions will have applications in many group security problems.

REFERENCES

Abdalla, M., Bellare, M., and Rogaway, P. 1998. DHAES: An encryption scheme based on the diffie-hellman problem. *Submited to IEEE P1363a*.

Ballardie, T. 1996. Scalable multicast key distribution. RFC 1949, may 1996.

Ballardie, T. and Crowcroft, J. 1995. Multicast-specific security threats and countermeasures. In *The Internet Soc. Symposium on Network and Distributed System Security, (February 16-17, 1995, San Diego, California)* (1995).

Bellare, M. and Rogaway, P. 1995. Optimal asymmetric encryption. In A. D. Santis Ed., *Advances in Cryptology - EuroCrypt '94* (Berlin, 1995), pp. 92–111. Springer-Verlag. Lecture Notes in Computer Science Volume 950.

Briscoe, B. 1999. MARKS: Zero side-effect multicast key management using arbitrarily revealed key sequences. In *First International Workshop on Networked Group Communication* (Nov. 1999).

Burmester, M. V. D. and Desmedt, Y. 1995. A secure and efficient conference key distribution system. In A. D. Santis Ed., *Advances in Cryptology - EuroCrypt '94* (Berlin, 1995), pp. 275–286. Springer-Verlag. Lecture Notes in Computer Science Volume 950.

Camenisch, J. and Stadler, M. 1997. Efficient group signature schemes for large groups. In *Advances in Cryptology - CRYPTO'97, 1997* (1997).

Camenisch, J. L. 1997. Efficient and generalized group signatures. In W. Fumy Ed., *Advances in Cryptology - EuroCrypt '97* (Berlin, 1997), pp. 465–479. Springer-Verlag. Lecture Notes in Computer Science Volume 1233.

Campbell, K. and Wiener, M. 1993. DES is not a group. In *In Advances in Cryptology - Crypto '92* (1993), pp. 512–520. Springer-Verlag.

Canetti, R., Garay, J., Itkis, G., Micciancio, D., Naor, M., and Pinkas, B. 1999. Multicast security: A taxonomy and some efficient constructions. In *Proceedings of IEEE Infocom'99* (1999).

Caronni, G., Valdvogel, M., Sun, D., and Plattner, B. 1998. Efficient security for large and dynamic multicast groups. In *Proceedings of IEEE WETICE'98* (1998).

Chang, I., Engel, R., Kandlur, D., Pendarakis, D., and Saha, D. 1999. Key management for secure internet multicast using boolean function minimization techniques. In *Proceedings IEEE Infocomm'99*, Volume 2 (March 1999), pp. 689–698.

Chaum, D. and Heyst, E. v. 1991. Group signatures. In D. W. Davies Ed., *Advances in Cryptology - EuroCrypt '91* (Berlin, 1991), pp. 257–265. Springer-Verlag. Lecture Notes in Computer Science Volume 547.

Chen, L. and Pedersen, T. P. 1995. New group signature schemes. In A. D. Santis Ed., *Advances in Cryptology - EuroCrypt '94* (Berlin, 1995), pp. 171–181. Springer-Verlag. Lecture Notes in Computer Science Volume 950.

Elgamal, T. 1984. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Crytology - CRYPTO'84, Santa Barbara, California, USA* (1984).

Fiat, A. and Naor, M. 1993. Broadcast encryption. In D. R. Stinson Ed., *Advances in Cryptology - Crypto '93* (Berlin, 1993), pp. 480–491. Springer-Verlag. Lecture Notes in Computer Science Volume 773.

Harn, L. and Kiesler, T. 1989. Authenticated group key distribution scheme for a large distributed network. In *Symposium on Security and Privacy* (1989).

Just, M. and Vaudenay, S. 1996. Authenticated multi-party key agreement. In *In Advances in Cryptology - Asiacrypt'96* (1996), pp. 36–49. Springer-Verlag.

Kaliski, B. S., Rivest, R. L., and Sherman, A. T. 1985. Is the Data Encryption Standard a group? In *Advances in Crytology - CRYPTO'85, Santa Barbara, California, USA* (1985).

McGrew, D. A. and Sherman, A. T. 1998. Key establishment in large dynamic groups using one-way function trees. Technical report, TIS Labs at Network Associates, Inc., Glenwood, MD.

Mittra, S. 1997. Iolus: A framework for scalable secure multicasting. In *Proceedings of the ACM SIGCOMM'97 (September 14-18, 1997, Cannes, France)* (1997).

Rivest, R. L., Shamir, A., and Adleman, L. M. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 21(2):120-126 21(2)*, 120–126.

RSA Security Inc. 1999. PKCS-1 v2.1: RSA cryptography standard. Technical report, RSA Security Inc.

Steiner, M., Tsudik, G., and Waidner, M. 1996. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Communications Security (March 14-16, 1996, New Delhi, India)* (1996).

Steiner, M., Tsudik, G., and Waidner, M. 1998. CLIQUES: A new approach to group key agreement. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)* (Amsterdam, May 1998), pp. 380–387. IEEECSP.

Wallner, D. M., Harder, E. J., and Agee, R. C. 1998. Key management for multicast: Issues and architectures. Internet draft, Network working group, september 1998.

Wong, C. K., Gouda, M., and Lam, S. S. 1998. Secure group communications using key graphs. In *ACM SIGCOMM 1998* (1998), pp. 68–79.