

# On the Security of Group Communication Schemes based on Symmetric Key Cryptosystems\*

Shouhuai Xu

Department of Computer Science, University of Texas at San Antonio

shxu@cs.utsa.edu

## ABSTRACT

Many emerging applications in both wired and wireless networks, such as information dissemination and distributed collaboration in an adversarial environment, need support of secure group communications. There have been many such schemes in the setting of wired networks. These schemes can be directly adopted in, or appropriately adapted to, the setting of wireless networks such as mobile ad hoc networks (MANETs) and sensor networks. In this paper we show that the popular group communication schemes that we have examined are vulnerable to the following attack: an outsider adversary who compromises a legitimate group member could obtain some or all *past* group keys as well as the current group key; this is in sharp contrast to the widely-accepted belief that a such adversary can only obtain the current group key. This attack is very powerful also because it provides the adversary the following flexibility: since the adversary knows which members are the “most valuable” ones from its own perspective of view, compromise of any such member leads to the exposure of all the past and current group keys. This flexibility is particularly relevant in the setting of MANETs and sensor networks because they are typically deployed in a small area and the adversary can capture and compromise the easiest-to-obtain node. In order to deal with this powerful attack, we formalize two security models for stateful and stateless group communication schemes, respectively. We show that some practical methods can make a *subclass* of the group communication schemes immune to this attack at the following *extra* expense: at each rekeying event, a group member conducts logarithmically-many pseudorandom function evaluations.

## Categories and Subject Descriptors

C.2 [COMPUTER-COMMUNICATION NETWORKS]:  
Distributed Systems

\*Supported in part by the Army Research Office (ARO), NSF, and UTSA CIAS.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SASN'05, November 7, 2005, Alexandria, Virginia, USA.  
Copyright 2005 ACM 1-59593-227-5/05/0011 ...\$5.00.

## General Terms

Security

## Keywords

security, group communication, broadcast encryption, key management, forward-security, backward-security

## 1. INTRODUCTION

Secure group communications are useful in both wired and wireless networks. One important property of secure group communications is the assurance that only the legitimate members (or users, receivers) can have access to the multicast or broadcast data. For this purpose, there have been many secure group communication schemes in the setting of wired networks; popular ones include the stateful LKH [26, 25] and OFT [23, 2] as well as the stateless schemes (e.g., [20, 10]). These schemes can be directly adopted in, or appropriately adapted to, the setting of wireless networks such as mobile ad hoc networks (MANETs) and sensor networks.

In this paper we show that these group communication schemes are subject to the following attack: an outsider adversary who compromises a legitimate group member could obtain some or all past group keys as well as the current group key (and thus the multicast data encrypted using these keys). This is in sharp contrast to the widely-accepted belief in these schemes that a such adversary can only obtain the current group key. This attack is very powerful also because it provides the adversary the following flexibility: since the adversary knows which members are the “most valuable” ones from its own perspective of view (see examples in Section 1.1), compromise of any such member leads to the exposure of all the past and current group keys. This flexibility is particularly relevant in the setting of MANETs and sensor networks because they are typically deployed in a small area and the adversary can capture and compromise the easiest-to-obtain node.

### 1.1 Motivating Problems

Now we explore some attack scenarios against the stateful LKH [26, 25] and OFT [23, 2], and the stateless [20, 10].

**Vulnerability of the LKH and LKH+ Schemes:** Let's first briefly review the LKH (also called WGL in this paper) group communication scheme. Following the notations of [26], we let

$$x \rightarrow \{y_1, \dots, y_\ell\} : \{z\}_w$$

denote that  $x$  sends the users  $y_1, \dots, y_\ell$  (via multicast or

unicast) the encryption of plaintext  $z$  using key  $w$ , namely the ciphertext  $\{z\}_w$ .

Consider the simple scenario, as shown in Figure 1.(a), of a group consisting of a key server  $s$  and users  $u_1, \dots, u_8$ . The server is responsible for initiating and maintaining the group in the presence of user joining and leaving. The keys are organized as a *key tree*, where the leaves are the users and the inner nodes are the keys. Moreover, each user holds the keys corresponding to the inner nodes on the path starting from the parent of the user and ending at the root. For example, in Figure 1.(a), user  $u_1$  holds keys  $k_1$ ,  $k_{123}$ , and  $k_{1-8}$ , where  $k_{1-8}$  is the *group key* that can be used to encrypt the communications within the group.

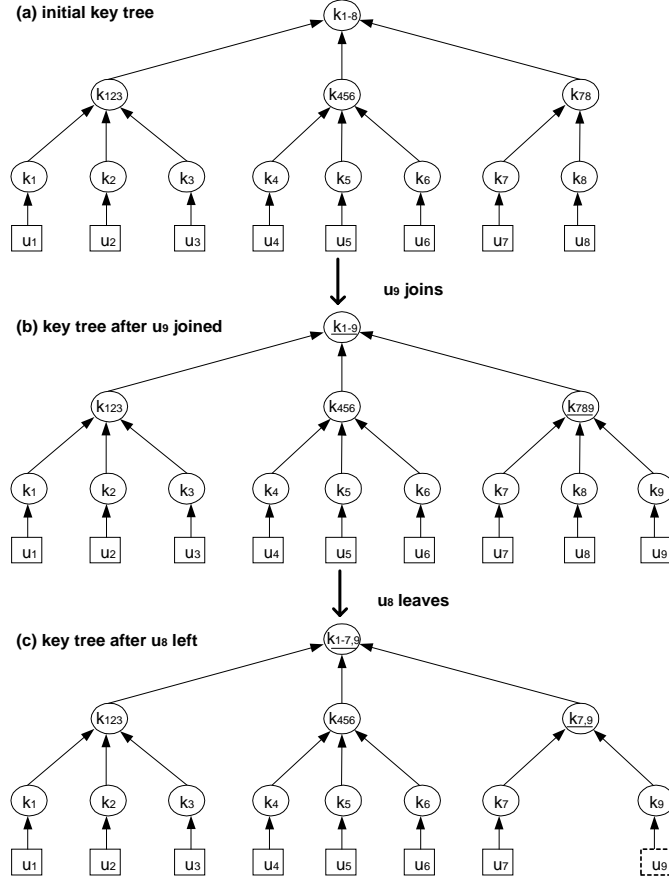


Figure 1: A simple scenario with a possible attack

Because the group is dynamic, meaning that some users join the group while some users leave it. In order to maintain secure communications, each join or leave would require the key server to change some keys that also need to be securely distributed to certain users (via some *rekeying messages*). Ignoring for a moment certain details such as authorization of joining the group and authentication of the messages sent by the key server, in what follows we explain how the key server responds to a group dynamics.

After granting a join request from user  $u_9$ , server  $s$  shares a key  $k_9$  with user  $u_9$ . Besides, certain keys need to be changed and sent to the corresponding users. As shown in Figure 1.(b), in order to prevent  $u_9$  from accessing past

communications, the key  $k_{78}$  and  $k_{1-8}$  are changed to  $k_{789}$  and  $k_{1-9}$ , respectively. Moreover,  $k_{789}$  and  $k_{1-9}$  need to be securely sent to users  $u_7$ ,  $u_8$ , and  $u_9$ . One efficient way to do this is the following algorithm (which corresponds to the so-called *group-oriented rekeying strategy*):

$$\begin{aligned} s \rightarrow \{u_1, \dots, u_8\} & : \{k_{1-9}\}_{k_{1-8}}, \{k_{789}\}_{k_{78}} \\ s \rightarrow \{u_9\} & : \{k_{1-9}, k_{789}\}_{k_9} \end{aligned}$$

Now, suppose  $u_8$  leaves. To prevent  $u_8$  from accessing future communications, as shown in Figure 1.(c), server  $s$  needs to change the keys  $k_{1-9}$  and  $k_{789}$  to new keys  $k_{1-7,9}$  and  $k_{7,9}$ , respectively. Moreover,  $k_{1-7,9}$  and  $k_{7,9}$  need to be securely sent to users  $u_7$  and  $u_9$ , and  $k_{1-7,9}$  needs to be securely sent to users  $u_1, \dots, u_6$ . One efficient way to do this is the following algorithm (which also corresponds to the *group-oriented rekeying strategy*):

$$\begin{aligned} s \rightarrow \{u_1, \dots, u_7, u_9\} & : \{k_{1-7,9}\}_{k_{123}}, \{k_{1-7,9}\}_{k_{456}}, \\ & \{k_{1-7,9}\}_{k_{7,9}}, \{k_{7,9}\}_{k_7}, \{k_{7,9}\}_{k_9} \end{aligned}$$

Suppose now an adversary compromises user  $u_9$ . It is of course true that the adversary is always able to obtain the *current* group key  $k_{1-7,9}$ , no matter how the group rekeying scheme works. However, the adversary who has recorded the network traffic is also able to obtain the group key  $k_{1-9}$ , because it can decrypt the message incurred by the event that  $u_9$  joins the group

$$s \rightarrow \{u_9\} : \{k_{1-9}, k_{789}\}_{k_9}.$$

As a consequence, the adversary can decrypt **both** the communications encrypted using group keys  $k_{1-9}$  and  $k_{1-7,9}$ . This is in sharp contrast to the desired property that the adversary can decrypt **only** the communications encrypted using group key  $k_{1-7,9}$ . We notice that the above attack is not fundamentally related to the group-oriented rekeying strategy, or to the fact that  $u_8$  – the sibling of the newly joined user  $u_9$  – leaves the group. We also notice that the above attack is not fundamentally related to the fact that  $u_9$  is a recently joined node. For example, suppose group dynamics is incurred by some users belonging to  $\{u_4, u_5, u_6, u_7, u_8, u_9\}$ , then it is possible that  $k_{123}$  is always used to encrypt the new group keys so that  $u_1, u_2$ , and  $u_3$  can obtain them. As a consequence,  $u_1, u_2$ , and  $u_3$  are the “most valuable” users from the adversary’s perspective of view, and compromising any of them will enable the adversary to recover all the past and current group keys.

The LKH+ [24], which was seemingly motivated from an efficiency perspective, resolves only piece of the problem because the above attack remains effective when the group dynamics are incurred by leaving events.

In summary, it is clear that the past group keys, which were ever encrypted using any of the keys held by a user that is being corrupt, are exposed. Under the extreme circumstance that the newly corrupt node holds the keys that have been used to encrypt all the past group keys, all the past and current group keys are compromised. This extreme scenario could occur because the adversary knows which nodes are “most valuable” ones from its own perspective of view.

**Vulnerability of the One-way Function Tree (OFT) Scheme:** The OFT [23, 2] is a stateful group communication scheme. The basic idea underlying the OFT scheme is the following. The center maintains a binary tree, each node  $x$  of which is associated with two cryptographic keys, a node key  $k_x$  and a blinded node key  $k'_x = g(k_x)$ , where  $g$

is an appropriate one-way function. Every leaf of the tree is associated with a group member, and the center assigns a randomly chosen key  $k_x$  to each member  $x$ . Let  $f$  be a “mixing” function (e.g.,  $\oplus$ ). The interior node keys are defined by the rule

$$k_u = f(g(k_{left(u)}), g(k_{right(u)}))$$

where  $left(u)$  and  $right(u)$  are the left and right children of the node  $u$ , respectively. This way, the node key associated with the root of the tree is the group key. In order for a member  $u$  to derive the group key, the center (or server, sender) sends the blinded node keys of nodes adjacent to the nodes on (i.e., of the nodes “hanging” off) the path from  $u$  to the root.

When a new member joins the group, an existing leaf node  $u$  is split, the member associated with  $u$  is now associated with  $left(u)$ , and the new member is associated with  $right(u)$ . Both members are given new keys. The new blinded node keys that have been changed are *securely sent* to the appropriate subgroups of members.

When the member associated with a node  $u$  is evicted from the group, the member assigned to the sibling of  $u$  is reassigned to the parent  $p$  of  $u$  and given a new leaf key. If the sibling  $s$  of  $u$  is the root of a subtree, then  $p$  becomes  $s$ , moving the subtree closer to the root, and one of the leaves of this subtree is given a new key. The new blinded node keys are *securely sent* to the appropriate subgroups of members.

Now we show why the OFT scheme is also vulnerable to a similar attack. The key observation is that whenever there is a change to any blinded node key, the center needs to *securely send* the new blinded node keys to certain other legitimate nodes. It seems any reasonable method would be based on the keys possessed by the respective nodes (e.g.,  $u$ ). Since  $u$  can derive the new group key after receiving the rekeying message, an *outsider* adversary could use the following strategy to recover this group key: it first records the rekeying message, and then breaks into  $u$ ’s computer *after* the next rekeying event (assuming that  $u$  is still legitimate). Moreover, the “most valuable” members, from the adversary’s perspective of view, are those who have never been evicted.

**Vulnerability of the stateless subset-cover framework:** Naor et al. [20] presented the first *practical* stateless group communication scheme, which has its roots in broadcast encryption [8]. Compared with the stateful group communication schemes discussed above, stateless schemes have the nice feature that they do not assume the receivers (or users, members) being always on-line. Since the receivers do not necessarily update their state from session to session, stateless schemes are especially good for applications over lossy channels (e.g., MANETs and sensor networks).

Let  $\mathcal{N}$  be the set of all users,  $\mathcal{R} \subset \mathcal{N}$  be a group of  $|\mathcal{R}| = r$  users whose decryption privileges should be revoked,  $E_L$  and  $F_K$  be two appropriate symmetric key cryptosystems. The goal of a stateless group communication scheme is to allow a center to transmit a message  $M$  to all users such that any user  $u \in \mathcal{N} \setminus \mathcal{R}$  can decrypt the message correctly, while even a coalition consisting of all members of  $\mathcal{R}$  cannot decrypt it. Suppose  $S_1, \dots, S_w$  are a collection of subsets of users, where  $S_j \subseteq \mathcal{N}$  for  $1 \leq j \leq w$ , and each  $S_j$  is assigned a long-lived key  $L_j$  such that each  $u \in S_j$  should be able to deduce  $L_j$  from its secret information  $I_u$ . Given a revoked set  $\mathcal{R}$ , if one

can partition  $\mathcal{N} \setminus \mathcal{R}$  into (ideally disjoint) sets  $S_{i_1}, \dots, S_{i_m}$  such that  $\mathcal{N} \setminus \mathcal{R} \subseteq \cup_{\ell=1}^m S_{i_\ell}$ , then a message-encryption key  $K$  can be encrypted  $m$  times with  $L_{i_1}, \dots, L_{i_m}$ , and each user  $u \in \mathcal{N} \setminus \mathcal{R}$  can obtain  $K$  and thus  $M$ . The subset-cover framework of [20] is reviewed below.

**Initialization:** Every receiver  $u$  is assigned private information  $I_u$ . For all  $1 \leq i \leq w$  such that  $u \in S_i$ ,  $I_u$  allows  $u$  to deduce  $L_i$  corresponding to the set  $S_i$ .

**Broadcasting:** Given a set  $\mathcal{R}$  of revoked receivers, the center (or server, sender) executes the following:

1. Choose a session encryption key  $K$ .
2. Find a partition of the users in  $\mathcal{N} \setminus \mathcal{R}$  into disjoint subsets  $S_{i_1}, \dots, S_{i_m}$ . Let  $L_{i_1}, \dots, L_{i_m}$  be the keys associated with the above subsets.
3. Encrypt  $K$  with keys  $L_{i_1}, \dots, L_{i_m}$  and sends the ciphertext

$$\langle [i_1, \dots, i_m, E_{L_{i_1}}(K), \dots, E_{L_{i_m}}(K)], F_K(M) \rangle.$$

**Decryption:** A receiver  $u$ , upon receiving a broadcast message  $\langle [i_1, \dots, i_m, C_1, \dots, C_m], C \rangle$ , executes as follows.

1. Find  $i_j$  such that  $u \in S_{i_j}$  (in the case  $u \in \mathcal{R}$  the result is NULL).
2. Extract the corresponding key  $L_{i_j}$  from  $I_u$ .
3. Decrypt  $C_j$  using key  $L_{i_j}$  to obtain  $K$ .
4. Decrypt  $C$  using key  $K$  to obtain the message  $M$ .

The subset-cover framework has a vulnerability similar to the one against the stateful group schemes. Specifically, suppose an adversary  $\mathcal{A} \notin \mathcal{N}$  records all the encrypted communications over the channels. If  $\mathcal{A}$  breaks into a legitimate user  $u \in \mathcal{N}$  at a later point of time, then  $\mathcal{A}$  obtains  $I_u$ , which allows it to recover the  $L_{i_j}$  (and thus the encrypted  $M$ ) that  $u$  was entitled to obtain. In the extreme case that  $u$  has never been revoked (i.e.,  $u$  is one of the “most valuable” users),  $\mathcal{A}$  can recover all the past and current keys that have been used to encrypt messages.

## 1.2 Our Contributions

We trace the above vulnerability of group communication schemes back to that their security models (if any) are not sufficient. We formalize two adversarial models. One is called the *passive attack model*, in which the adversary is passive in the sense that it is only allowed to join and leave the group in an arbitrary fashion, but not allowed to corrupt any legitimate members. This model has seemingly been implicitly adopted in the existing group communication literature. The other more realistic one is called the *active outsider attack model*, in which the adversary is further allowed to corrupt legitimate members. This model helps understand and deal with the aforementioned attack. In each of the two models, we define two security notions, namely *forward-security* meaning that the revoked or evicted members, even if they collude, cannot obtain the future group keys, and *backward-security* meaning that a newly admitted member cannot obtain the past group keys.<sup>1</sup> This allows

<sup>1</sup>The terms follow the group communication literature (see, e.g., [26, 25]). Their meanings are indeed different from the ones adopted in the cryptographic literature [1, 3, 4].

us to obtain the following interesting results about the relationships between these security notions, which are equally applicable to both stateful and stateless group communication schemes.

1. Backward-security in the active outsider attack model (also called **strong-security** for short) is *strictly* stronger than forward-security in the active outsider attack model (also called **security** for short). This means that in the active outsider attack model one only needs to prove the backward-security property.
2. Backward-security in the passive attack model is equivalent to forward-security in the passive attack model.
3. Backward-security in the active outsider attack model (i.e., **strong-security**) is *strictly* stronger than backward-security in the passive attack model. However, we do **not** know whether forward-security in active outsider attack model is also *strictly* stronger than its counterpart in the passive attack model (we only know that when the adversary is *static* they are equivalent).
4. The security achieved in existing group communication schemes (e.g., [26, 23, 20, 10]) is indeed, as we will show, forward-security in the active outsider attack model (i.e., **security**). This has become clear only until now because there were no formal models specified before (in spite of the fact that the passive attack model has seemingly been implicitly adopted in the literature). The achieved **security** property is at least as strong as what we call backward-security in the passive attack model, but *strictly* weaker than what we call backward-security in the active outsider attack model (i.e., **strong-security**) — a property that dismisses the attack discussed above.

Besides the above general results, we show that some practical methods can transform a *subclass* of the group communication schemes (including LKH [26, 25], LKH+ [24], and the complete subtree method [20]) into ones that achieve the desired **strong-security** (i.e., backward-security in the active outsider attack model). The methods are based on two general compilers. The extra complexity imposed by the compilers is that at each rekeying event a group member conducts logarithmically-many pseudorandom function evaluations. This should not jeopardize their utility even in the setting of MANETs and sensor networks, as pseudorandom functions may be implemented with block ciphers in practice. We also present instantiations of the compilers, which lead to concrete schemes that achieve the desired **strong-security**.

Although the technical means underlying the transformation is to evolve the keys based on a secure pseudorandom function family — an idea inspired by [4], there are some subtle issues in our security models. First, we must allow the adversary to corrupt some group keys that are used to encrypt the communications *before* the rekeying message of interest. Of course, the corrupt members must have been revoked before this rekeying message. On the other hand, in [4] no such corruption is allowed before the event of interest. Second, from an adversary’s perspective of view, there could be many “most valuable” users (i.e., an intelligent adversary only needs to select the easiest-to-obtain one) to compromise in our setting. Whereas, no such flexibility is given to the adversary in the setting of [4].

### 1.3 Related Work

The WGL scheme was independently invented in [25]. Both schemes are sometimes called Logical Key Hierarchy (LKH). Although these schemes are mainly invented for secure multicast applications, we believe that many other applications can utilize such a scheme; we refer the reader to [21, 6] for a survey, including the relationship between the schemes of [26, 25] and the schemes of [8, 20]. We notice that the stateless schemes (e.g., [20, 10]) are perhaps more useful in an environment of lossy channels. Although the LKH scheme has been extended in several directions, these extensions are motivated to improve performance rather than to achieve strictly stronger security. For example, performance can be improved by periodic group rekeying [22] or batch rekeying [17], and improved trade-offs between storage and communication are available in [7, 6, 19]. Nevertheless, these techniques may also be utilized in our **strongly-secure** group communication schemes. To the best of our knowledge, our work is the first one that identifies a new and realistic attack, and presents solutions to (a subclass of) the popular group communication schemes. The variant presented in [24] (which is also known as LKH+) is similar to our performance optimization that the communication complexity incurred by joining events can be substantially reduced. However, there was no formal treatment of the utilized key evolution, nor was their scheme resistant against the attack introduced in Section 1.1.

While secure group-oriented communications have been intensively investigated in the setting of wired networks, their counterparts in the setting of wireless networks are yet to be thoroughly explored. Although the aforementioned schemes can be directly deployed in wireless settings, a simple-minded adoption may not lead to the desired performance (see, e.g., [28, 15]). Fortunately, there have been some interesting investigations that show that these schemes can be adapted (e.g., by taking into account some physical characteristics of ad hoc networks [16, 14, 13, 15]) so that better performance can be achieved. One of the practical values of the present paper is that the significantly improved security guarantee in the popular group communication schemes can be seamlessly integrated into the methods for improving performance [16, 14, 13, 15]. Indeed, our schemes can be easily integrated into any other methods for improving performance to achieve better security, as long as the methods assume “black-box” access to an underlying security group communication scheme. There have been few other attempts at securing group communications in such settings. [12] presented a scheme for secure multicast communications in MANETs based on public key cryptosystems. In contrast, we consider only symmetric key cryptosystems based group communication schemes. Yet another approach to secure group communication scheme was initiated in [28], which is partially stateless.

**Outline.** The rest of the paper is organized as follows. In Section 2 we briefly review some cryptographic preliminaries. In Section 3 we present formal models and security definitions of stateful group communication schemes, as well as the relationships between the security notions. In Section 4 we present a compiler for stateful group communication schemes and investigate its properties. The compiler is utilized in Section 5 to derive a concrete **strongly-secure** stateful group communication scheme out of the **secure** WGL scheme, which is reviewed in Appendix A for com-

pletteness. We conclude the paper in Section 6. Due to space limitation, we leave most of the details as well as the full exploration of the stateless case to the full version of the present paper [27].

## 2. CRYPTOGRAPHIC PRELIMINARIES

A function  $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$  is *negligible* if  $\forall c \exists \kappa_c$  s.t.  $\forall \kappa > \kappa_c$ , we have  $\epsilon(\kappa) < 1/\kappa^c$ .

We will base security of group communication schemes on the security of pseudorandom function families. A pseudorandom function (PRF) family  $\{f_k\}$  parameterized by a secret value  $k \in_R \{0, 1\}^\kappa$  has the following property [9]: A probabilistic polynomial-time adversary  $\mathcal{A}$  has only a negligible (in  $\kappa$ ) advantage in distinguishing  $f_k$  from a perfect random function (with the same domain and range). It is well-known that pseudorandom functions can be naturally used to construct symmetric key encryption schemes that are secure against chosen-plaintext attacks (which suffices our treatment of the WGL scheme). Informally, this means that no adversary is able to learn any significant information about the encrypted content. We refer the reader to [11] for a thorough treatment on this subject.

**DEFINITION 2.1.** (computational independence) *Consider a set  $S = \{s_1, \dots, s_\ell\}$  of secret binary strings of length  $\kappa$ , where  $\ell = \text{poly}(\kappa)$  for some polynomial  $\text{poly}$ . We say  $s_1, \dots, s_\ell$  are computationally independent of each other if for any probabilistic polynomial-time algorithm  $\mathcal{A}$  and any  $i \in \{1, \dots, \ell\}$ ,*

$$\Pr[\text{partialInfo}(s_i) \leftarrow \mathcal{A}(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_\ell)] = \epsilon(\kappa)$$

where  $\text{partialInfo}$  is any non-trivial polynomial-time computable predicate with respect to  $s_i$  and  $\epsilon$  is a negligible function. This is equivalent to that  $\mathcal{A}$  cannot distinguish  $s_i$  from a random string of the same length.

## 3. MODEL AND DEFINITION OF STATEFUL GROUP COMMUNICATION SCHEMES

### 3.1 Model

Let  $\kappa$  be a security parameter, and  $\mathbb{ID}$  be the set of possible group members (i.e., users, receivers, or principals) such that  $|\mathbb{ID}|$  is polynomially-bounded in  $\kappa$ . There is a special entity called a *Group Controller* (i.e., key server, center, server, or sender), denoted by  $\mathcal{GC}$ , such that  $\mathcal{GC} \notin \mathbb{ID}$ .

Since a stateful group communication scheme is driven by “rekeying” events (because of joining or leaving operations below), it is convenient to treat the events occur at “virtual time”  $t = 0, 1, 2, \dots$ , because the group controller is able to maintain such an execution history. This indeed accommodates the following important two cases: (1) all the parties periodically update their keys, even if there are no joining or leaving operations — this is relevant when a scheme achieves what we call **strong-security** specified below; (2) the lengths of the time periods do not have to be the same — this is the case when the rekeying events occur in an arbitrary fashion. At time  $t$ , let  $\Delta^{(t)}$  denote the set of legitimate group members,  $k^{(t)} = k_{\mathcal{GC}}^{(t)} = k_{U_1}^{(t)} = \dots$  the

group key,<sup>2</sup>  $K_{\mathcal{GC}}^{(t)}$  the set of keys held by  $\mathcal{GC}$ ,  $K_U^{(t)}$  the set of keys held by  $U \in \Delta^{(t)}$ ,  $\text{acc}_U^{(t)}$  the state indicating whether  $U \in \Delta^{(t)}$  has successfully received the rekeying message. Initially,  $\forall U \in \mathbb{ID}, t \in \mathbb{N}$ , set  $\text{acc}_U^{(t)} \leftarrow \text{FALSE}$ . We assume that  $\mathcal{GC}$  treat joining and leaving operation separately (e.g., first fulfilling the leaving operation and then immediately the joining one), even if the requests are made simultaneously. This strategy has indeed been adopted in the group communication literature.

To simplify the presentation, we assume that during the system initialization (i.e., **Setup** below)  $\mathcal{GC}$  can communicate with each legitimate member  $U$  through an *authenticated private* channel. In practice, this assumption can be implemented with a two-party authenticated key-exchange protocol. Further, we assume that  $\mathcal{GC}$  can establish a common secret, if needed, with a joining user, and that after the system initialization  $\mathcal{GC}$  can communicate with any  $U \in \mathbb{ID}$  through an *authenticated* channel. These assumptions can be implemented with a digital signature scheme [26], and digital signatures are sometimes necessary [5].

Although we will not make any *synchronization* assumption about the underlying communication model, which could therefore be asynchronous [18]. However, known practical schemes (e.g., [26, 25, 7]) assume *reliable* delivery, which would require some (loose) clock synchronization.

A group communication scheme has the following components:

**Setup:** The group controller  $\mathcal{GC}$  generates a set of keys  $K_{\mathcal{GC}}^{(0)}$ , and distributes them to the current group members (that may be determined by the adversary),  $\Delta^{(0)} \subseteq \mathbb{ID}$ , through the authenticated private channels. (In the case that some users were corrupted before this setup procedure, we may let the adversary select the keys held by the corrupt users.) Each member  $U_i \in \Delta^{(0)}$  holds a set of keys denoted by  $K_{U_i}^{(0)} \subset K_{\mathcal{GC}}^{(0)}$ , and there is a key,  $k^{(0)}$  that is common to all the current members, namely  $k^{(0)} \in K_{\mathcal{GC}}^{(0)} \cap K_{U_1}^{(0)} \cap \dots \cap K_{U_{|\Delta^{(0)}|}}^{(0)}$ .

**Join:** This algorithm is executed by group controller  $\mathcal{GC}$  at time, say,  $t$  due to some join request(s) (we abstract away the out-of-band authentication and establishment of an individual key for each of the new members). It takes as input: (1) identities of previous group members,  $\Delta^{(t-1)}$ , (2) identities of newly admitted group members,  $\Delta' \subseteq \mathbb{ID} \setminus \Delta^{(t-1)}$ , (3) keys held by the group controller,  $K_{\mathcal{GC}}^{(t-1)}$ , and (4) keys held by group members,  $\{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}} = \{K_{U_i}^{(t-1)} : U_i \in \Delta^{(t-1)}\}$ .

It outputs updated system state information, including: (1) identities of new group members,  $\Delta^{(t)} \leftarrow \Delta^{(t-1)} \cup \Delta'$ , (2) new keys for  $\mathcal{GC}$  itself,  $K_{\mathcal{GC}}^{(t)}$ , (3) new keys for new group members,  $\{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}$ , which are *somehow* sent to the legitimate users through the authenticated channels (depending on concrete schemes), (4) new group key  $k^{(t)} \in K_{\mathcal{GC}}^{(t)} \cap K_{U_1}^{(t)} \cap \dots \cap K_{U_{|\Delta^{(t)}|}}^{(t)}$ .

Formally, denote it by  $(\Delta^{(t)}, K_{\mathcal{GC}}^{(t)}, \{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}) \leftarrow \text{Join}(\Delta^{(t-1)}, \Delta', K_{\mathcal{GC}}^{(t-1)}, \{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}})$ .

<sup>2</sup>It is also known as a session key in the group communication literature.

**Leave:** This algorithm is executed by the group controller  $\mathcal{GC}$  at time, say,  $t$  due to leave or revocation operation(s). It takes as input: (1) identities of previous group members,  $\Delta^{(t-1)}$ , (2) identities of leaving group members,  $\Delta' \subseteq \Delta^{(t-1)}$ , (3) keys held by the controller,  $K_{\mathcal{GC}}^{(t-1)}$ , and (4) keys held by group members,  $\{K_{U_i}\}_{U_i \in \Delta^{(t-1)}}$ . It outputs updated system state information, including: (1) identities of new group members,  $\Delta^{(t)} \leftarrow \Delta^{(t-1)} \setminus \Delta'$ , (2) new keys for  $\mathcal{GC}$ ,  $K_{\mathcal{GC}}^{(t)}$ , (3) new keys for new group members,  $\{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}$ , which are *somehow* sent to the legitimate users through the authenticated channels (depending on concrete schemes), (4) new group key  $k^{(t)} \in K_{\mathcal{GC}}^{(t)} \cap K_{U_1}^{(t)} \cap \dots \cap K_{U_{|\Delta^{(t)}|}}^{(t)}$ .

Formally, denote it by  $(\Delta^{(t)}, K_{\mathcal{GC}}^{(t)}, \{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}) \leftarrow \text{Leave}(\Delta^{(t-1)}, \Delta', K_{\mathcal{GC}}^{(t-1)}, \{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}})$ .

**Rekey:** This algorithm is executed by the legitimate group members at some time  $t$ , namely all  $U_i \in \Delta^{(t)}$  where  $\Delta^{(t)}$  is derived from a **Join** or **Leave** event. In other words,  $U_i \in \Delta^{(t)}$  runs this algorithm upon receiving the message from  $\mathcal{GC}$  over the authenticated channel. The algorithm takes as input the received message and  $U_i$ 's secrets, and is supposed to output the updated keys for the group member. If the execution of the algorithm is successful,  $U_i$  sets: (1)  $\text{acc}_{U_i}^{(t)} \leftarrow \text{TRUE}$ , (2)  $K_{U_i}^{(t)}$ , where  $k_{U_i}^{(t)} \in K_{U_i}^{(t)}$  is supposed to be the new group key.

If the rekeying event is incurred by a **Join** event, every  $U_i \in \Delta^{(t)}$  erases  $K_{U_i}^{(t-1)}$  and any temporary storage after obtaining  $K_{U_i}^{(t)}$ . If the rekeying event is incurred by a **Leave** event, every  $U_i \in \Delta^{(t)}$  erases  $K_{U_i}^{(t-1)}$  and any temporary storage after obtaining  $K_{U_i}^{(t)}$ , and every *honest* leaving group member  $U_j \in \Delta'$  erases  $K_{U_j}^{(t-1)}$  (although a *corrupt* one does not have to follow this protocol).

We require that any group communication scheme satisfy the following **correctness**: for any  $t = 1, 2, \dots$  and  $\forall U \in \Delta^{(t)}$ , if  $\text{acc}_U^{(t)} = \text{TRUE}$ , then  $k_U^{(t)} = k^{(t)}$  and  $K_U^{(t)} \subset K_{\mathcal{GC}}^{(t)}$ .

### 3.2 Security Definitions

We consider an adversary that has complete control over all the communications in the network. To simplify the definition, we assume that the group controller is never compromised; this is not necessarily a restriction because the adversary could have compromised all the group members (and thus have obtained the secrets the group controller holds).

An adversary's interaction with principals in the network is modeled by allowing it to have access to (some of) the following oracles:

- $\mathcal{O}_{\text{Send}}(U, t, M, \text{action})$ : Send a message  $M$  to  $U' \in \{\mathcal{GC}\} \cup \mathbb{ID}$  at time  $t \geq 0$ , and output its reply, where  $\text{action} \in \{\text{Setup}, \text{Join}, \text{Leave}, \text{Rekey}\}$  meaning that  $U'$  will execute according to the corresponding protocol, and  $M$  specifies the needed information for executing the protocol. Of course, the query of type **Setup** is only made at time  $t = 0$ .

- $\mathcal{O}_{\text{Reveal}}(U, t)$ : Output group key held by  $U \in \Delta^{(t)}$  at time  $t$ , namely  $k_U^{(t)}$ .
- $\mathcal{O}_{\text{Corrupt}}(U, t)$ : Output the keys held by  $U \in \Delta^{(t)}$  at time  $t$ , namely  $K_U^{(t)}$ .
- $\mathcal{O}_{\text{Test}}(U, t)$ : This oracle may be queried only once, at any time during the adversary's execution. A random bit  $b$  is generated: if  $b = 1$  the adversary is given  $k_U^{(t)}$  where  $U \in \Delta^{(t)}$ , and if  $b = 0$  the adversary is given a random key of length  $|k_U^{(t)}|$ .

**DEFINITION 3.1.** (active outsider attack model) *In this model, the adversary  $\mathcal{A}$  may have access to all the oracles specified above. In particular, an "outsider"  $\mathcal{A} \notin \Delta^{(t)}$  is allowed to issue an  $\mathcal{O}_{\text{Reveal}}(U, t)$  or  $\mathcal{O}_{\text{Corrupt}}(U, t)$  query for some  $U \in \Delta^{(t)}$ .*

**DEFINITION 3.2.** (passive attack model) *In this model, the adversary is only allowed to make  $\mathcal{O}_{\text{Send}}(\cdot, \cdot, \cdot, \cdot)$  and  $\mathcal{O}_{\text{Test}}(\cdot, \cdot)$  queries. In other words, the adversary is only allowed to join and leave the group (in an arbitrary fashion though).*

In each of the two models, we define two security notions: backward-security and forward-security.

**DEFINITION 3.3.** (**security**) *Intuitively, it means that  $\mathcal{A}$  learns no information about a group key if (1) with respect to the corresponding rekeying event there is no corrupt legitimate member (this implicitly implies that all the members that were corrupted by  $\mathcal{A}$  must have been revoked), and (2) no member is corrupted by  $\mathcal{A}$  after the rekeying event. Formally, consider the following event **Succ**:*

- (1) *The adversary queries the  $\mathcal{O}_{\text{Test}}(U, t)$  oracle with  $\text{acc}_U^{(t)} = \text{TRUE}$ , and correctly guesses the bit  $b$  used by the  $\mathcal{O}_{\text{Test}}(U, t)$  oracle in answering this query.*
- (2) *There is no  $\mathcal{O}_{\text{Reveal}}(V, t)$  query for any  $V \in \Delta^{(t)}$ . (Otherwise, the group key is trivially compromised.)*
- (3) *If there was any  $\mathcal{O}_{\text{Corrupt}}(V, t_1)$  query where  $t_1 < t$ , then there must have been an  $\mathcal{O}_{\text{Send}}(\mathcal{GC}, t_2, V, \text{Leave})$  query where  $t_1 < t_2 \leq t$ . This captures that the corrupt members must have been revoked before the rekeying message at time  $t$ .*
- (4) *There is no  $\mathcal{O}_{\text{Corrupt}}(V, t_3)$  query for any  $t_3 \geq t$  and  $V \in \Delta^{(t_3)}$ .*

The advantage of the adversary  $\mathcal{A}$  in attacking the group communication scheme is defined as  $\text{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\text{Succ}] - 1|$ , where  $\Pr[\text{Succ}]$  is the probability that the event **Succ** occurs, and the probability is taken over the coins used by  $\mathcal{GC}$  and by  $\mathcal{A}$ . We say a scheme is a **secure** if for all probabilistic polynomial-time adversary  $\mathcal{A}$  it holds that  $\text{Adv}_{\mathcal{A}}(\kappa)$  is negligible in  $\kappa$ .

**DEFINITION 3.4.** (**strong-security**) *Intuitively, it means that an adversary learns no information about a group key if, with respect to the rekeying event of interest there is no corrupt legitimate member (this implicitly implies that all the previously corrupt members have been revoked). Formally, consider the following event **Succ**:*

(1)-(3) The same as in the definition of security.

(4) There is no  $\mathcal{O}_{\text{Corrupt}}(V, t_3)$  query for  $t_3 = t$  and  $V \in \Delta^{(t_3)}$ . (This does not rule out that there could be some  $\mathcal{O}_{\text{Corrupt}}(V, t_3)$  query for  $t_3 > t$ .)

The advantage of the adversary  $\mathcal{A}$  in attacking the group communication scheme is defined as  $\text{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\text{Succ}] - 1|$ , where  $\Pr[\text{Succ}]$  is the probability that the event **Succ** occurs, and the probability is taken over the coins used by  $\mathcal{GC}$  and by  $\mathcal{A}$ . We say a scheme is **strongly-secure** if for all probabilistic polynomial-time adversary  $\mathcal{A}$  it holds that  $\text{Adv}_{\mathcal{A}}(\kappa)$  is negligible in  $\kappa$ .

**DEFINITION 3.5.** (forward-security in the passive attack model) This definition is the same as the **security** definition except that  $\mathcal{A}$  is always not allowed to make any  $\mathcal{O}_{\text{Reveal}}$  or  $\mathcal{O}_{\text{Corrupt}}$  query.

**DEFINITION 3.6.** (backward-security in the passive attack model) This definition is the same as the **strong-security** definition except that  $\mathcal{A}$  is always not allowed to make any  $\mathcal{O}_{\text{Reveal}}$  or  $\mathcal{O}_{\text{Corrupt}}$  query (i.e., it can only take advantage of the joining/leaving operations).

### 3.3 On the Relationships between the Security Notions

We summarize the relationships between the security notions of stateful group communication schemes in Fig. 2, where  $X \rightarrow Y$  means  $X$  is stronger than  $Y$ ,  $X \leftrightarrow Y$  means  $X$  is equivalent to  $Y$ ,  $X \not\rightarrow Y$  means  $X$  does not imply  $Y$ , and  $X \stackrel{?}{\not\rightarrow} Y$  means it is unclear where  $X$  does not imply  $Y$  — conjectured though. The exploration of the non-trivial relationships showed in Fig. 2 is deferred to [27].

## 4. A COMPILER

Suppose  $\{f_k\}$  is a secure pseudorandom function family. Now we present a compiler that transforms a **secure** group communication scheme  $\text{SGC} = (\text{Setup}, \text{Join}, \text{Leave}, \text{Rekey})$  into a **strongly-secure** group communication scheme  $\text{SSGC} = (\text{Setup}^*, \text{Join}^*, \text{Leave}^*, \text{Rekey}^*)$ . The compiler applies to the subclass of stateful group communication schemes where the different keys belong to  $K_{\mathcal{GC}}^{(t)} \cup K_{U_1}^{(t)} \cup \dots \cup K_{U_{|\Delta^{(t)}|}}^{(t)}$  are computationally independent of each other, where  $t = 0, 1, 2, \dots$ . In what follows “a key  $k$  needs to be changed” means that it should be substituted with a random key that is information-theoretically independent of  $k$ .

**Setup\***: This is the same as  $\text{SGC.Setup}$ .

**Join\***: This algorithm is executed by  $\mathcal{GC}$  at time, say,  $t$ . Let  $K$  be the set of keys that need to be changed (including the group key  $k^{(t-1)}$ ),  $K^*$  be the set of common key(s) shared between the  $\mathcal{GC}$  and the joining user(s), and  $K^{**}$  be the new keys (including the new group key  $k^{(t)}$ ) that are used to replace the keys in  $K$ .

1. Execute  $\text{SGC.Join}$  except for the following: (1) let  $f_{k_i}(0)$  play the role of the corresponding key  $k_i$ , where  $k_i \in (K_{\mathcal{GC}}^{(t-1)} \setminus \{k^{(t-1)}\}) \cup K^*$ ; (2) only if  $k_i \in K^{**} \setminus \{k^{(t)}\}$  is used as an encryption key in  $\text{SGC.Join}$ , let  $f_{k_i}(0)$  play the role of  $k_i$ .

2. Every individual key  $k_i \in (K_{\mathcal{GC}}^{(t-1)} \setminus K) \cup K^* \cup (K^{**} \setminus \{k^{(t)}\})$  is replaced by  $f_{k_i}(1)$ .

**Leave\***: This algorithm is executed by  $\mathcal{GC}$  at time, say,  $t$ . Let  $K$  be the set of keys that need to be changed (including the group key  $k^{(t-1)}$ ) or eliminated, and  $K^{**}$  be the new keys (including the new group key  $k^{(t)}$ ) that are used to replace (possibly a subset of) the keys in  $K$ .

1. Execute  $\text{SGC.Leave}$  except for the following: (1) let  $f_{k_i}(0)$  play the role of the corresponding key  $k_i$ , where  $k_i \in K_{\mathcal{GC}}^{(t-1)} \setminus K$ ; (2) only if  $k_i \in K^{**} \setminus \{k^{(t)}\}$  is used as an encryption key in  $\text{SGC.Leave}$ , let  $f_{k_i}(0)$  play the role of  $k_i$ .
2. Every individual key  $k_i \in (K_{\mathcal{GC}}^{(t-1)} \setminus K) \cup (K^{**} \setminus \{k^{(t)}\})$  is replaced by  $f_{k_i}(1)$ .

**Rekey\***: There are two cases.

- The rekeying event is incurred by a **Leave** event at time  $t$ . In this case, every honest leaving user should erase all the secrets as in  $\text{SGC.Rekey}$ , and every remaining user,  $V \in \Delta^{(t)}$ , executes the following. Denote by  $K'_V \subseteq K_V^{(t-1)}$  the subset of keys that need to be changed to a set of new keys  $K''_V$ . (We notice that both  $K'_V$  and  $K''_V$  can be derived by  $V$  after receiving the rekeying message and that  $k^{(t)} \in K''_V$ .) First,  $V$  executes  $\text{SGC.Rekey}$  except for letting  $f_{k_i}(0)$  play the role of  $k_i$  under the circumstance that  $k_i \in K_V^{(t-1)} \setminus \{k^{(t-1)}\}$  or  $k_i \in K''_V \setminus \{k^{(t)}\}$  is used as an encryption key, and updates every  $k_i \in (K_V^{(t-1)} \setminus K'_V) \cup (K''_V \setminus \{k^{(t)}\})$  as  $f_{k_i}(1)$ . Second,  $V$  erases the outdated keys (other than  $K_V^{(t)}$ ) as in  $\text{SGC.Rekey}$ .
- The rekeying event is incurred by a **Join** event at time  $t$ . We notice that user  $V \in \Delta^{(t)}$  holds a set of keys  $K_V^{(t-1)}$  (in the case of  $V$  being a joining user,  $K_V^{(t-1)}$  consists of the only common key between  $\mathcal{GC}$  and  $V$ ), of which a subset  $K'_V$  of keys (which may be empty) are to be changed to a set of new keys  $K''_V$ . (We notice that both  $K'_V$  and  $K''_V$  can be derived by  $V$  after receiving the rekeying message and that  $k^{(t)} \in K''_V$ .) First,  $V$  executes  $\text{SGC.Rekey}$  except for letting  $f_{k_i}(0)$  play the role of  $k_i$  under the circumstance that  $k_i \in K_V^{(t-1)} \setminus \{k^{(t-1)}\}$  or  $k_i \in K''_V \setminus \{k^{(t)}\}$  is used as an encryption key, and updates every  $k_i \in (K_V^{(t-1)} \setminus K'_V) \cup (K''_V \setminus \{k^{(t)}\})$  as  $f_{k_i}(1)$ . Second,  $V \in \Delta^{(t)}$  erases the outdated keys (other than  $K_V^{(t)}$ ) as in  $\text{SGC.Rekey}$ .

### 4.1 Analysis

First we analyze the complexity of  $\text{SSGC}$ .

- It does not introduce any extra communication complexity over  $\text{SGC}$ ; this is important in many applications such as MANETs and sensor networks.
- It does not introduce any extra storage complexity over  $\text{SGC}$ , provided that the temporary storage for the keys such as  $f_{k_i}(0)$  is insignificant.

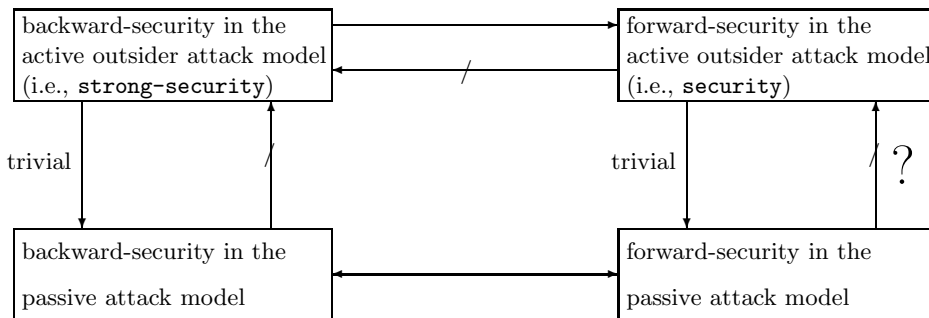


Figure 2: The relationships between the security notions in stateful group communication schemes

- The only extra complexity of SSGC over SGC is the evaluation of the pseudorandom functions. Specifically, the server needs to conduct  $O(\max\{|K_{GC}^{(t-1)}|, |K_{GC}^{(t)}|\})$  pseudorandom function evaluation operations; a user  $V$  needs to conduct  $O(\max\{|K_V^{(t-1)}|, |K_V^{(t)}|\})$  pseudorandom function operations. We notice that typically typically  $|K_V^{(t)}| = O(\log(|K_{GC}^{(t)}|))$  (e.g., [26]).

THEOREM 4.1. Assume  $\{f_k\}$  is a secure pseudorandom function family (as specified in Section 2). If SGC is **secure**, then SSGC is **strongly-secure**.

## 4.2 Performance Optimization

In this section we show how to reduce the communication complexity in the SSGC; this might be very useful in applications such as MANETs and sensor networks. Suppose a Join event occurs at time  $t$ . The key observation includes:

1. In Join\* of SSGC we could simply let the server sends the updated keys to the joining user  $U$ . We notice that, before  $U$  receiving the rekeying message from the server,  $K_U^{(t-1)}$  consists of a single key, denoted by  $k^*$ , that is also known to the server. After sending the rekeying message, the server update  $K_{GC}^{(t-1)} = \{k_i\}$  to  $K_{GC}^{(t)} = \{f_{k_i}(1)\} \cup \{f_{k^*}(1)\}$ .
2. When the joining user  $U$  executes Rekey\* corresponding to the Join\* (i.e., after receiving the rekeying message), it lets  $f_{k^*}(0)$  plays the role of  $k^*$ . Then,  $U$  updates  $k^*$  to  $f_{k^*}(1)$  while keeping intact the other keys received from the server.
3. When an existing user  $V \in \Delta^{(t-1)}$  executes Rekey\* corresponding to the Join\*, it simply updates every  $k_i \in K_V^{(t-1)}$  (including the group key) to  $f_{k_i}(1)$ .
4. The encryption of group communications is based on new group key  $k^{(t)} = f_{k^{(t-1)}}(1)$ .

We notice that the idea of substituting  $k_i$  via a certain function was pointed out in [24, 7] with respect to the specific scheme of [26]. Here we show that it can actually be extended to accommodate the class of group communication schemes discussed in this paper. This justifies why we treat it as a possible feature of the compiler, which we call *optimized compiler*.

THEOREM 4.2. Assume  $\{f_k\}$  is a secure pseudorandom function family, and SGC is **secure**. If SGC does not adopt the afore-discussed performance optimization, then the scheme output by the optimized compiler is also **strongly-secure**.

## 5. A CONCRETE STRONGLY-SECURE GROUP COMMUNICATION SCHEME

### 5.1 The WGL Model

The WGL model is best known as a *key tree*, which outperforms the others (e.g., star key graph, or general key graph which actually leads to a certain NP-hard problem as we always need to minimize the communication complexity). A key tree  $T$  can be seen as a special class of directed acyclic graph with two types of nodes: *u-nodes* representing users and *k-nodes* representing keys. Each *u-node* is a leaf that has one outgoing edge but no incoming edge, and each *k-node* is an inner node that has one or more incoming edges. Moreover, there is a *k-node* (i.e., the root) that has incoming edges but no outgoing edge.

Let  $U$  be a finite and nonempty set of users and  $K$  be a finite and nonempty set of keys. We are interested in a relation,  $R \subseteq U \times K$ , that can be specified by a key tree  $T$  as follows:

- There is a one-to-one correspondence between  $U$  and the set of *u-nodes* in  $T$ .
- There is a one-to-one correspondence between  $K$  and the set of *k-nodes* in  $T$ .
- $(u, k) \in R$  if and only if  $T$  has a directed path from the *u-node* that corresponds to a user  $u \in U$  to the *k-node* that corresponds to a key  $k \in K$ .

This means that the group key is at the root of the tree, which is shared by all the users in  $U$ . Since a key tree can be specified by two parameters – the height  $h$  of the tree is the length (in number of edges) of the longest directed path in the tree, and the degree  $d$  of the tree is the maximum number of incoming edges of a node in the tree – each user in  $U$  has at most  $h$  keys.

In order to clarify the presentation, we define two functions,  $\text{KEYSET} : U \rightarrow K$  and  $\text{USERSET} : K \rightarrow U$ , as follows:

$$\begin{aligned} \text{KEYSET}(u) &= \{k \mid (u, k) \in R\}, \\ \text{USERSET}(k) &= \{u \mid (u, k) \in R\}. \end{aligned}$$

Intuitively,  $\text{KEYSET}(u)$  is the set of keys held by user  $u \in U$ , and  $\text{USERSET}(k)$  is the set of users that hold key  $k \in K$ . Moreover, it is natural to generalize the definition of  $\text{KEYSET}(u \in U)$  to  $\text{KEYSET}(U' \subseteq U) = \bigcup_{u \in U'} \text{KEYSET}(u)$ , and the definition of  $\text{USERSET}(k \in K)$  to  $\text{USERSET}(K' \subseteq K) = \bigcup_{k \in K'} \text{USERSET}(k)$ .

## 5.2 The Scheme

The new scheme is obtained by applying the compiler described in Section 4 to the WGL scheme based on the so-called *group-oriented* rekeying strategy, which is reviewed in Appendix A for completeness. (The WGL scheme can be based on the less efficient *key-oriented* and *user-oriented* strategies [26]. Nevertheless, it should be straightforward to adapt our scheme to these rekeying strategies.) The resulting scheme consists of four protocols, namely SSGC = (Setup\*, Join\*, Leave\*, Rekey\*).

**Setup\***: The key server generates a key  $k_i$  for each  $k$ -node. After the initialization, each user (corresponding to a  $u$ -node) holds the keys corresponding to the path from its parent  $k$ -node to the root.

**Join\***: After granting a join request from user  $u$ , the key server  $s$  creates a new  $u$ -node for user  $u$  and a new  $k$ -node for its individual key  $k_u$ . Then, server  $s$  finds an existing  $k$ -node (called the *joining point* for this join request) in the key tree and attaches the  $k$ -node  $k_u$  to the joining point as its child. As a consequence, the keys corresponding to the path – starting at the joining point and ending at the root – need to be changed. The algorithm is specified below:

```

Join protocol for group-oriented rekeying:
  // suppose user  $u$  joins the group
  server  $s$  generates a new key  $k_u$  for user  $u$ 
  server  $s$  finds a joining point  $x_j$ 
  server  $s$  attaches  $k_u$  to  $x_j$ 
  let  $x_0$  be the root
  suppose  $x_{i-1}$  is the parent of  $x_i$  for  $1 \leq i \leq j$ 
   $\bar{k}_{j+1} \leftarrow k_u$ 
  let  $\bar{k}_0, \bar{k}_1, \dots, \bar{k}_j$  be the current keys
    of  $x_0, \dots, x_j$ , respectively
  server  $s$  generates fresh keys  $\hat{k}_0, \hat{k}_1, \dots, \hat{k}_j$ 
    // new keys of  $x_0, \dots, x_j$ 
   $s \rightarrow \text{USERSET}(\bar{k}_0) : \{\hat{k}_0\}_{\bar{k}_0}, \{\hat{k}_1\}_{f_{\bar{k}_1}(0)}, \dots, \{\hat{k}_j\}_{f_{\bar{k}_j}(0)}$ 
   $s \rightarrow \{u\} : \{\hat{k}_0, \hat{k}_1, \dots, \hat{k}_j\}_{f_{\bar{k}_{j+1}}(0)}$ 
  FOR all  $\bar{k} \in (\text{KEYSET}(\text{USERSET}(\bar{k}_0)) \setminus \{\bar{k}_0, \bar{k}_1, \dots, \bar{k}_j\})$ 
     $\cup \{\bar{k}_{j+1}\} \cup \{\hat{k}_1, \dots, \hat{k}_j\}$ 
     $\bar{k} \leftarrow f_{\bar{k}}(1)$ 

```

**Leave\***: After granting a leave request from user  $u$ , the key server  $s$  updates the key tree by deleting the  $u$ -node for user  $u$  and the  $k$ -node for its individual key from the key tree. The parent of the  $k$ -node corresponding to the user's individual key is called the *leaving point*. As a consequence, the keys corresponding to the path – starting at the leaving point and ending at the root – need to be changed. The algorithm is specified below:

```

Leave protocol for group-oriented rekeying:
  // suppose  $u$  leaves the group
  let  $x_{j+1}$  be the deleted  $k$ -node for  $k_u$ 
   $\bar{k}_{j+1} \leftarrow k_u$ 
  server  $s$  finds the leaving point  $x_j$  (parent of  $k_u$ )
  server  $s$  removes  $\bar{k}_{j+1}$  from the key tree
  let  $x_0$  be the root
  suppose  $x_{i-1}$  is the parent of  $x_i$  where  $1 \leq i \leq j$ 
  let  $\bar{k}_0, \bar{k}_1, \dots, \bar{k}_j$  be the keys of  $x_0, x_1, \dots, x_j$ 
    // they need to be changed
  server  $s$  generates fresh keys  $\hat{k}_0, \hat{k}_1, \dots, \hat{k}_j$  as
    the new keys of  $x_0, x_1, \dots, x_j$ 

```

```

FOR  $i = 0$  TO  $j - 1$ 
  let  $\bar{k}_{i_1}, \dots, \bar{k}_{i_{z_i}}$  be the keys at the children
    of  $x_i$  in the new key tree
    where  $\bar{k}_{i_a}$  is to be changed to  $\hat{k}_{i+1}$  for
      some  $a \in \{1, \dots, z_i\}$ 
   $L_i \leftarrow (\{\hat{k}_i\}_{f_{\bar{k}_{i_1}}(0)}, \dots, \{\hat{k}_i\}_{f_{\bar{k}_{i_{a-1}}}(0)},$ 
     $\{\hat{k}_i\}_{f_{\bar{k}_{i_{a+1}}}(0)}, \{\hat{k}_i\}_{f_{\bar{k}_{i_{a+1}}}(0)}, \dots, \{\hat{k}_i\}_{f_{\bar{k}_{i_{z_i}}}(0)})$ 
  let  $\bar{k}_{j_1}, \dots, \bar{k}_{j_{z_j}}$  be the keys at the children
    of  $x_j$  in the new key tree
   $L_j \leftarrow (\{\hat{k}_j\}_{f_{\bar{k}_{j_1}}(0)}, \dots, \{\hat{k}_j\}_{f_{\bar{k}_{j_{z_j}}}(0)})$ 
   $s \rightarrow \text{USERSET}(\bar{k}_0) \setminus \{u\} : (L_0, \dots, L_j)$ 
  FOR all  $\bar{k} \in (\text{KEYSET}(\text{USERSET}(\bar{k}_0)) \setminus \{\bar{k}_0, \bar{k}_1, \dots,$ 
     $\bar{k}_{j+1}\}) \cup \{\hat{k}_1, \dots, \hat{k}_j\}$ 
     $\bar{k} \leftarrow f_{\bar{k}}(1)$ 

```

**Rekey\***: If the rekeying event is incurred by a join event, a legitimate user (i.e., an existing one or a joining one) obtains a subset  $\Theta'$  of  $\Theta = \{\hat{k}_0, \hat{k}_1, \dots, \hat{k}_j\}$ , and updates each  $\hat{k} \in \Theta' \setminus \{\hat{k}_0\}$  to  $f_{\bar{k}}(1)$ . If the rekeying event is incurred by a leave event, a legitimate user (i.e., one remaining in the group) obtains a subset  $\Theta'$  of  $\Theta = \{\hat{k}_0, \hat{k}_1, \dots, \hat{k}_j\}$ , and updates each  $\hat{k} \in \Theta' \setminus \{\hat{k}_0\}$  to  $f_{\bar{k}}(1)$ . In any case, a legitimate user  $u$  updates each  $k_i \in \text{KEYSET}(u)$  to  $f_{k_i}(1)$ , as long as  $k_i$  is not changed to any key belonging to  $\Theta$ , and erases the outdated keys.

## 5.3 Analysis

**THEOREM 5.1.** *Assume that the stand-alone encryptions utilized in the WGL encryption scheme are based on a secure pseudorandom function family. Then, the WGL scheme is secure.*

As a corollary of Theorem 4.1 and Theorem 5.1 we have:

**COROLLARY 5.1.** *The scheme presented in Section 5.2 is strongly-secure.*

## 6. CONCLUSION AND OPEN PROBLEMS

We showed that a class of existing group communication schemes, both stateful and stateless alike, are vulnerable to a realistic severe attack. We presented formal models that allow us to capture the desired security properties, and explore the relationships between the security notions. We showed how some methods can make a *subclass* of existing schemes immune to the attack at a very small extra cost. An interesting open question is to make other schemes (e.g., the stateful [23, 2] and the stateless [10]) secure against the attack without imposing any significant extra complexity.

## 7. ACKNOWLEDGEMENTS

We thank Jonathan Katz for illuminating discussions that lead to the refined model in Section 3, our shepherd, Donggang Liu, for helpful feedback, and the anonymous reviewers for useful comments.

## 8. REFERENCES

- [1] R. Anderson. On the forward security of digital signatures. <http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-549.pdf>.

- [2] D. Balenson, D. McGrew, and A. Sherman. *Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization*. Internet Engineering Task Force, Feb. 1999.
- [3] M. Bellare and S. Miner. A forward-secure digital signature scheme. In *Proc. CRYPTO 99*, pp 431–448.
- [4] M. Bellare and B. Yee. Forward-security in private-key cryptography. In *Cryptographer’s Track - RSA Conference (CT-RSA)*, pp 1–18.
- [5] D. Boneh, G. Durfee, and M. Franklin. Lower bounds for multicast message authentication. In *Proc. EUROCRYPT 2002*, pp 437–452.
- [6] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *IEEE INFOCOM 1999*, pp 708–716, 1999.
- [7] R. Canetti, T. Malkin, and K. Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *Proc. EUROCRYPT 99*, pp 459–474.
- [8] A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Proc. CRYPTO 93*, pp 480–491.
- [9] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, Oct. 1986.
- [10] N. Jho, J. Hwang, J. Cheon, M. Kim, D. Lee, and E. Yoo. One-way chain based broadcast encryption schemes. In *Proc. EUROCRYPT 2005*, pp 559–574.
- [11] J. Katz and M. Yung. Complete characterization of security notions for probabilistic private-key encryption. In *Proc. STOC 2000*, pp 245–254.
- [12] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz. Secure multicast groups on ad hoc networks. In *Proc. SASN 2003*, pp 94–102.
- [13] L. Lazos and R. Poovendran. Energy-aware secure multicast communication in ad-hoc networks using geographic location information. In *Proc. IEEE ICASSP 2003*.
- [14] L. Lazos and R. Poovendran. Cross-layer design for energy-efficient secure multicast communications in ad hoc networks. In *Proc. IEEE ICC’04*.
- [15] L. Lazos and R. Poovendran. Power proximity based key management for secure multicast in ad hoc networks. *ACM Journal on Wireless Networks (WINET)*, ??(?), to appear.
- [16] L. Lazos, J. Salido, and R. Poovendran. Vp3: Using vertex path and power proximity for energy efficient key distribution. In *Proc. IEEE VTC’04 (invited paper)*.
- [17] X. Li, Y. Yang, M. Gouda, and S. Lam. Batch rekeying for secure group communications. In *Proc. WWW 2001*, pp 525–534.
- [18] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [19] D. Micciancio and S. Panjwani. Optimal communication complexity of generic multicast key distribution. In *Proc. EUROCRYPT 2004*, pp 153–170.
- [20] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proc. CRYPTO 2001*, pp 41–62.
- [21] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Survey*, 35(3):309–329, 2003.
- [22] S. Setia, S. Koussih, S. Jajordia, and E. Harder. Kronos: A scalable group re-keying approach for secure multicast. In *Proc. IEEE Symposium on Security and Privacy 2000*, pp 215–228.
- [23] A. Sherman and D. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Trans. Softw. Eng.*, 29(5):444–458, 2003.
- [24] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The VersaKey framework: Versatile group key management. *IEEE Journal on Selected Areas in Communications*, 17(9):1614–1631, Sept. 1999.
- [25] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. Internet Draft, Sept. 1998.
- [26] C. Wong, M. Gouda, and S. Lam. Secure group communication using key graphs. *IEEE/ACM Trans. on Networking* (also in SIGCOMM’98), 8, 2000.
- [27] S. Xu. On the security of group communication schemes based on symmetric key cryptosystems. Full version available from the author, 2005.
- [28] S. Zhu, S. Setia, S. Xu, and S. Jajodia. Gkmpn: An efficient group rekeying scheme for secure multicast in ad-hoc networks. In *MobiQuitous 2004*, pp 42–51.

## APPENDIX

### A. THE WGL PROTOCOLS

Join protocol for group-oriented rekeying:

```
// suppose user  $u$  joins the group
server  $s$  generates a new key  $k_u$  for user  $u$ 
server  $s$  finds a joining point  $x_j$ 
server  $s$  attaches  $k_u$  to  $x_j$ 
let  $x_0$  be the root
 $\bar{k}_{j+1} \leftarrow k_u$ 
suppose  $x_{i-1}$  is the parent of  $x_i$  for  $1 \leq i \leq j$ 
let  $\bar{k}_0, \bar{k}_1, \dots, \bar{k}_j$  be the current keys
of  $x_0, x_1, \dots, x_j$ , respectively
server  $s$  generates fresh keys  $\hat{k}_0, \hat{k}_1, \dots, \hat{k}_j$ 
// new keys of  $x_0, \dots, x_j$ 
 $s \rightarrow \text{USERSET}(\bar{k}_0) : \{\hat{k}_0\}_{\bar{k}_0}, \{\hat{k}_1\}_{\bar{k}_1}, \dots, \{\hat{k}_j\}_{\bar{k}_j}$ 
 $s \rightarrow u : \{\hat{k}_0, \hat{k}_1, \dots, \hat{k}_j\}_{k_u}$ 
```

Leave protocol for group-oriented rekeying:

```
// suppose  $u$  leaves the group
let  $x_{j+1}$  be the deleted  $k$ -node for  $k_u$ 
 $\bar{k}_{j+1} \leftarrow k_u$ 
server  $s$  finds the leaving point  $x_j$  (parent of  $k_u$ )
server  $s$  removes  $\bar{k}_{j+1}$  from the key tree
let  $x_0$  be the root
suppose  $x_{i-1}$  is the parent of  $x_i$  where  $1 \leq i \leq j$ 
let  $\bar{k}_0, \bar{k}_1, \dots, \bar{k}_j$  be the keys of  $x_0, x_1, \dots, x_j$ 
// they need to be changed
server  $s$  generates fresh keys  $\hat{k}_0, \hat{k}_1, \dots, \hat{k}_j$  as
the new keys of  $x_0, x_1, \dots, x_j$ , respectively
FOR  $i = 0$  TO  $j$ 
let  $\bar{k}_{i_1}, \dots, \bar{k}_{i_{z_i}}$  be the keys at the children
of  $x_i$  in the new key tree
 $L_i \leftarrow (\{\hat{k}_i\}_{\bar{k}_{i_1}}, \dots, \{\hat{k}_i\}_{\bar{k}_{i_{z_i}}})$ 
 $s \rightarrow \text{USERSET}(\bar{k}_0) \setminus \{u\} : (L_0, \dots, L_j)$ 
```