

# Scalable Secure Bidirectional Group Communication

Yitao Duan and John Canny  
Computer Science Division  
University of California, Berkeley  
Berkeley, CA 94720, USA  
Email: {duan, jfc}@cs.berkeley.edu

**Abstract**—Many network applications are based on a group communications model where one party sends messages to a large number of authorized recipients and/or receives messages from multiple senders. In this paper we present a secure group communication scheme based on a new cryptosystem that admits a rigorous proof of security against adaptive chosen ciphertext attack (IND-CCA2). Our scheme is bi-directional, supporting both one-to-many and many-to-one communications. Compared with existing solutions, our scheme achieves the following improvements: (1) It guarantees data confidentiality and authenticity in *both* directions; (2) It is the most scalable solution so far among all existing schemes achieving (1). The group member storage overhead is constant while both the center storage and rekeying communication complexity are independent of group size. (3) It can be made to achieve higher level of security and hide even the information about the group dynamics. We show that this protection is more effective and more efficient than existing solutions.

## I. INTRODUCTION

A large number of existing and emerging network applications are based on *group communications* where one party, denoted the *center*, sends messages to a large number of authorized recipients and/or receives data from multiple sources. The one-to-many communication model is typically denoted *multicast* and we call the many-to-one pattern *aggregation*. Examples of group-oriented commercial applications include pay-per-view distribution of digital media, multi-player games, teleconferencing etc. Group communications can also be seen in many noncommercial and non-IP-based applications such as military communication, sensor network, etc.

Multicast offers a scalable solution to delivering the same messages to a group of receivers. On the Internet, multicast can be performed at both IP and application levels. For instance, at IP level, a multicast group is identified by a Class D IP address and any receivers can join or leave a multicast group by sending IGMP (Internet Group Management Protocol) [1] messages to their local router. Any sender can send message to a multicast group by addressing the message to the group address. The current IP Multicast is an “open” service in that it does not restrict delivery of data to a specified set of receivers. Access control in multicast must be achieved via other means. A typical solution is to encrypt data using symmetric-key encryption with a global traffic encryption key (TEK) that is known only to the multicast group members. The difficulty here is key management: the TEK may have to be changed

whenever members join or leave the group. This is known as rekeying.

Compared to multicast, aggregation as a communication paradigm received less attention, especially in the context of IP networks, and was largely studied in the settings of other network communication such as sensor networks. However, we observe that, in a realistic application scenario, there are compelling needs for aggregation, even in “multicast” applications. For instance, in a pay-per-view broadcast system, a user needs to transmit messages to the center to request the service or confirm her agreement to pay for it.

Another area where many-to-one communication is becoming more and more important is network Intrusion Detection Systems (IDS). Most IDSes share a similar underlying structure: agents (or sensors, probes) reporting detections to a management system. This already falls into the aggregation communication paradigm. Furthermore, many present intrusion detection systems are known to have problems such as alert flooding, scalability, etc., and data aggregation is proposed to address these issues [2]. Data aggregation enables the system to group alerts together, analyze data in a broader context and distribute the load of handling alerts. A basic prerequisite for data aggregation is the support for many-to-one communication.

In this paper we consider the security in *bidirectional* group communication (i.e., both multicast and aggregation) and present a scalable solution. Similar to unicast communication, there are two fundamental security properties in group communication: data *confidentiality* and *authenticity*. Confidentiality ensures that no parties other than the intended recipients should be able to access the message. Authenticity, on the other hand, is concerned with protecting the message against tamper and guaranteeing that it is indeed originated from the alleged sender. With existing technology, these two properties pose different challenges in the two communication modes we are considering. In multicast, authenticity is easy to achieve since there is only one sender and it is not much of a burden for each receiver to store the center’s public key that can be used to verify its signature. In aggregation, on the other hand, it is confidentiality that can be obtained trivially, by using the center’s public key to establish secure channels. However, authenticity is crucial in aggregation-based applications. For example, in an IDS, the system must ensure the authenticity

of the data sent by each probe otherwise an attacker can inject false monitoring data and thwart the intrusion detection effort. Thus in this paper we focus on achieving confidentiality in multicast and authenticity in aggregation.

Authenticating aggregation messages poses unique challenges and it is not easy to simply “patch” existing multicast schemes with aggregation security while still maintaining the same scalability. Let us denote the information that a user uses to authenticate her aggregation messages the user’s *authentication key*. For the authentication to work, the authentication keys must not be global or public. The “obvious” solution of running a PRF on a user’s ID and using the resulting random bits as authentication key simply does not work: since the IDs are public, anyone can produce the authentication key of any other users. Using the global multicast secret key has the same problem. Indeed, it appears that, without resorting to expensive digital signature, the center must share some pairwise secret with each members of the group for this purpose.

Efficiency wise, since in aggregation, the center is handling many data sources, the authentication method must be efficient otherwise the center will become a bottleneck. In addition, one must consider possible “hidden cost” when evaluating the feasibility of a scheme. Some schemes may feature smaller keys but they require the center to store a list of all active member IDs. The effective storage overhead is still linear in the size of the group. In a large scale system storing and accessing such a list can be too expensive.

In this paper we present a scheme for bidirectional group communication that provides both confidentiality and authenticity in *both* directions. Our scheme is based on a new multicast framework recently proposed by Duan and Canny [3] that can be used to construct multicast cryptosystems provably secure against adaptive chosen ciphertext attacks (IND-CCA2) which is a very strong notion of security. We show that the key structure of the Duan-Canny (DC) multicast system provides a natural and efficient framework for sharing a pair-wise secret between the center and each member (to be elaborated in section IV). And this feature can be exploited to construct efficient group communication schemes that are secure in both directions. The contributions of our work include

- We define the communication paradigm of aggregation and formalize its security properties that are appropriate for most group communication applications.
- We introduce Alternating Bit DC (ABDC) to overcome the limitation of DC cryptosystem and allow the revocation of an arbitrary number of members.
- We present “in-place” update that provides backward confidentiality which is lacking in the original DC system.
- We extend DC construction’s key arrangement to provide an efficient and scalable solution for authenticating aggregation messages. This is achieved while maintaining DC system’s scalability. Our scheme uses symmetric key crypto for authenticating aggregation messages thus is much more efficient than digital signature-based solutions. In our scheme, the data authentication tag also serves as a group membership authentication which al-

lows the center to verify the data authenticity and the sender’s membership *without* having to keep the list of active member IDs online.

- We show that a scheme based on DC construction is affable to protecting group dynamics information (GDI) due to its property that the size of the ciphertext is independent of the group size. We present techniques that protect such information even against members of the group. To the best of our knowledge, this is the only effective GDI protection technique (to be elaborated in Section VI).

## II. PRELIMINARIES

We consider the situation where a single party, called *the center*, communicates, over insecure channels, with a group of  $n$  parties who are called *the members* of the group. The communication is assumed to be two way (i.e. both multicast and aggregation) and we require that security properties defined in Section II-A be guaranteed.

We assume the center is also trusted with managing the group membership. This is realistic in many applications (since oftentimes the center is the data distributor) and is in line with almost all existing multicast schemes such as [4], [5], [6], [7], [8], [9], [10]. We assume a computationally bounded adversary who is allowed to attack the system from both outside and *inside* the group. The insider’s attack is modelled by allowing the adversary to corrupt and gain total control of up to  $t$  group members where  $t$  is a predefined threshold. The system deals with corrupted members by evicting them from the group.

We adopt the following notation in subsequent discussion. For both symmetric and asymmetric cryptosystems, let  $E_K(X)$  denote the encryption of  $X$  with key  $K$  and  $D_K(Y)$  be the decryption of  $Y$  with key  $K$ . We use  $x \leftarrow y$  to denote that  $x$  is assigned the value of  $y$ . We write  $x \leftarrow_R D$  to denote the process of selecting uniformly randomly from domain  $D$  an element and assigning it to  $x$ . We assume the process can be completed in constant time. We use  $||$  to denote concatenation. For a bit  $b$ , we write  $\bar{b}$  to denote its complement.

### A. Security in Bidirectional Group Communication

The security notions in group communication are complicated by the fact that the group can be dynamic with members joining and leaving. Research in secure multicast has identified a number of security properties in dynamic group communication:

- M.1 Non-group Confidentiality: users that were never part of the group should not have access to any multicast data sent to the group;
- M.2 Forward Confidentiality: users deleted from the group at some time  $\tau$  do not have access to any data after  $\tau$ , unless they are authorized to join the group again;
- M.3 Collusion Freedom: no subset of deleted users should be able to decrypt future group communication, even by sharing the keys they had before deletion;

M.4 Backward Confidentiality: a user added at time  $\tau$  should not have access to any data before  $\tau$  while the user was not part of the group.

Aggregation security can be defined as as the dual of multicast security. We introduce the following:

- A.1 Non-group Authenticity: users that were never part of the group should not have the ability to forge messages that the center will accept as coming from one of its members;
- A.2 Forward Authenticity: users deleted from the group at some time  $\tau$  cannot create messages that the center accepts as originated from one of its members after  $\tau$ , unless they are authorized to join the group again;
- A.3 Collusion Freedom: no subset of  $t$  or less *active* members, nor any subset of deleted members, should be able to forge messages that the center accepts as originated from another member not in the colluding subset, even by sharing the their keys;
- A.4 Backward Authenticity: a user added at time  $\tau$  should not have the ability to forge messages that the center accepts as coming from a member who was in the group before  $\tau$ .

Note that in A.3 we allow the collusion of (up to  $t$ ) *active* members, not only deleted ones. This is different from the collusion freedom for multicast (M.3). In multicast, the primary concern is data confidentiality. Any active member by definition should have access to the multicast message so it does not make sense to protect against their collusion. M.3 is therefore actually expressed as a stronger version of forward confidentiality (M.2). We keep the definitions separate because they emphasize different aspects of the system’s security (temporal and collusion-resilience, respectively) for which most previous multicast security works kept separate definitions.

The primary security goal for aggregation, on the other hand, is data authenticity which should be resilient against forgery and tamper even by active members. A.3 defines such attack by, in addition to deleted members, allowing a subset of  $t$  or less active members to collude. Note that A.4 is actually implied by A.3. We keep a separate definition to stress the temporal aspect of the security goal.

Also note that the definitions state that a scheme should be resilient against collusion of *any number* of deleted members. The threshold  $t$  only restricts the number of colliding *active* members. These properties provide basic security goals for many bidirectional group communication applications and will be used as guidelines for designing and evaluating our scheme.

In addition, some applications may demand even higher level of security and require that the information about the group dynamics such as group size and user join/departure rate be kept secret from an adversary. We discuss this issue in Section VI and show that our scheme can be made to effectively protect such information.

### III. OVERVIEW OF DC MULTICAST CRYPTOSYSTEM CONSTRUCTION

In this section we give an overview of Duan-Canny multicast cryptosystem construction. We focus on how DC multicast cryptosystem works and omit some non-relevant elements from the notation. For detailed definition and proof of security please see [3].

Fundamental to [3]’s constructions is a public-key cryptosystem with a sharable decryption. Let  $\mathcal{E} = (\text{KeyGen}, \text{E}, \text{D})$  be such a cryptosystem. It consists of the following algorithms:

- 1) Key Generation KeyGen: a probabilistic polynomial-time algorithm that, on a given security parameter  $1^k$  (expressed in unary), generates a private-public key pair  $(x, y)$ .
- 2) Encryption E: a probabilistic polynomial-time algorithm that, on inputs  $y$ , the encryption key, and a string  $m \in \{0, 1\}^k$ , produces as output  $\psi \in \{0, 1\}^*$  called the ciphertext.
- 3) Decryption D: a deterministic polynomial-time algorithm such that  $\forall m \in \{0, 1\}^k, D_x(\text{E}_y(m)) = m$ . On all other inputs it outputs a special symbol  $\perp$ .

There also exists a share generation algorithm that, given  $(x, y)$ , a threshold  $t$  and an integer  $n > t$ , generates  $x_1, \dots, x_n$  (in the same space as  $x$ ), called *shares* of  $x$  such that any subset of at least  $t + 1$  shares uniquely determines the secret key  $x$  while any subset with less than  $t + 1$  shares has no information about  $x$ . In addition, D should be “sharable”: given a ciphertext  $\psi$  and a share  $x_i$  of the secret key, D produces a share of the decryption  $\psi_i = (i, D_{x_i}(\psi))$  such that given a set  $\Lambda$  of the decryption shares with  $|\Lambda| > t$ , a “share combination” algorithm can produce the correct decryption  $\eta(\Lambda, y) = D_x(\psi)$ .

DC encryption works as follows. The center generates a public/private key pair (e.g., ElGamal key pair), publishes the public key  $y$  and secret-shares the private key  $x$  using  $(t + 1, n + t)$ -threshold scheme among all group members and the center. Let  $x_1, x_2, \dots, x_{n+t}$  be the shares of  $x$ . Each member  $i$  gets  $x_i$  and the center is given  $\{(j, x_j) | j \in T\}$  where  $T = \{n + 1, n + 2, \dots, n + t\}$ . To multicast the message  $m \in \mathcal{M}$ , the center encrypts it using  $y$ . Let  $c = \text{E}_y(m)$ . It then decrypts  $c$   $t$  times using its shares of  $x$  and produces  $\bar{m} = \{(j, m_j = D_{x_j}(c)) | j \in T\}$ . The ciphertext is  $\psi = (c, \bar{m}, T)$ . The center multicasts  $\psi$  to the group. To decrypt the message, member  $i$  first decrypts  $c$  using  $x_i$ , her share of the private key, and then combines all the partial decryptions she has to obtain

$$m = \eta(\bar{m} \cup \{(i, D_{x_i}(c))\}, y)$$

DC encryption can revoke up to  $t$  members. Let  $R$  be the indices of the members whose capability to decrypt the message should be revoked. The center construct a “blacklist”  $\bar{T}$  as  $\bar{T} = R \cup T'$  where  $T'$  is a random subset of  $T$  with  $t - |R|$  elements. The center then use  $\bar{T}$  in place of  $T$  in the encryption. The members whose indices are in  $R$  will not be able to decrypt the message.

[3] proved that their scheme is secure against adaptive chosen ciphertext attack (IND-CCA2) when  $\mathcal{E}$  is IND-CCA2. This security is defined against up to  $t$  colluding members. In the following, to take advantage of this guaranteed high security, we will assume an IND-CCA2 DC cryptosystem.

#### IV. SCALABLE BIDIRECTIONAL GROUP COMMUNICATION

To make it concrete, we describe our group communication scheme using a DC multicast cryptosystem instantiated with Shoup and Gennaro's TDH2 threshold scheme in [11]. This instantiation was based on the decisional Diffie-Hellman (DDH) problem and proven secure against adaptive chosen ciphertext attack. For detailed proof and description of the scheme please see [11] and [3]. For the purpose of this paper, it suffices to know that the encryption is essentially ElGamal augmented with non-interactive zero-knowledge proof of discrete logarithm (to thwart chosen ciphertext attack, which we omit in this paper). We defer the description of the scheme until after we introduce some improvements over the original DC cryptosystem.

##### A. Extension for Aggregation

Although DC cryptosystem is for multicast encryption, its key structure has a very nice property that can be used to efficiently authenticate *aggregate* messages. Recall that the center is trusted with both distributing messages and managing the group. In this case the center can be given  $t+1$  shares of  $x$ . Let  $T^+ = T \cup \{n+t+1\}$  and  $\Gamma = \{(i, x_i) | i \in T^+\}$  are given to the center. Each member  $i$  attaches to an aggregation message a secure message authentication code (MAC) computed with a key derived from  $x_i$ . The center first recovers  $x_i$  using  $\Gamma$  and then tests the validity of the MAC. This uses symmetric crypto and is much more efficient than digital signature. This is essentially sharing a distinct secret key between the center and each member, *without* having to require the center to store all of them. The center stores only  $t+1$  keys instead of  $n$  and recovers the member keys as needed. Theoretically, using only these  $t+1$  keys, the center is able to authenticate an *arbitrary* number of members, limited only by the system's assumption on the proportion of members that can be simultaneously corrupted. This provides a scalable solution for large groups.

##### B. Alternating Bit DC

DC cryptosystem allows for the revocation of up to  $t$  members. This count is accumulated from the start of the system. If the total number of members that need to be revoked during the life time of this group communication should exceed  $t$ , their scheme must be "refreshed" with new public key and private key shares. This process cannot be carried out using multicast and must resort to the inefficient pairwise unicast communication. Yet the need to evict more than  $t$  members is quite compelling in many applications and this limitation may force the communication to pause for the refreshing.

To overcome this limitation, we devised an improved scheme, denoted Alternating Bit DC (ABDC), that moves

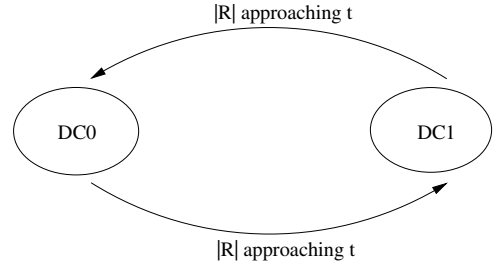


Fig. 1. Alternating Bit DC

the refreshing process off-line and allows for uninterrupted communication. ABDC enables the revocation of up to  $t$  members for each rekeying operation and allows arbitrary number of members, limited by the size of the group, to be revoked during the lifetime of the communication.

ABDC alternates between two Duan-Canny schemes, denoted  $DC_0$  and  $DC_1$ . The system maintains a bit  $b$  that specifies which is currently being used and toggles the bit when one scheme is approaching its limit of  $t$  revocations. This information is included in the rekeying message and members will respond by switching to the other DC scheme. The previous DC scheme is then refreshed off-line. The transition is illustrated in Figure 1 and the process and messages will be clear in Section IV-F.

##### C. Initialization

This stage is carried out when the communication group is first being formed. Let  $n$  be the current size of the group. Let  $p$  and  $q$  be two large primes selected by the key generation algorithm as defined by the DC cryptosystem on a security parameter  $1^k$  such that  $q|p-1$ . We use  $G_q$  to denote the unique subgroup of  $\mathbb{Z}_p^*$  of order  $q$  and let  $g, \bar{g}$  be a generators of  $G_q$ . DDH is assumed to hold in  $G_q$ .

For  $b = 0, 1$ , the center performs the following:

- 1) It generates  $t+1$  uniformly random numbers  $f_{b0}, f_{b1}, \dots, f_{bt} \in \mathbb{Z}_q$  and constructs a degree  $t$  polynomial  $f_b(x) = \sum_{j=0}^t f_{bj}x^j$ .
- 2) It computes  $x_{bi} = f_b(i) \bmod q$  and  $h_{bi} = g^{x_{bi}} \bmod p$  for  $i = 1, 2, \dots, n+t+1$ . Let  $x_b = f_{b0} = f_b(0)$  and  $h_b = h_{b0} = g^{x_b} \bmod p$ .
- 3) It sends  $x_{bi}$  to member  $i$  via secure channel and stores  $\Gamma_b = \{(j, x_{bj}) | j \in T^+\}$ .  $h_b$  is made public.
- 4) It sets  $R \leftarrow \emptyset$  where  $R$  is the set of indexes of the members whose memberships have been revoked.

The public key  $y_b$  for  $DC_b$  consists of a description of  $G$  and  $(g, \bar{g}, h_b)$ . The private key for each member  $i$  is  $(y_b, i, x_{bi})$ . The private key for the center is  $(y_b, \Gamma_b)$ <sup>1</sup>.

When the center is sending the keys to the members, it includes  $TEK_0$  and  $b$  as well, also via the secure unicast channels.  $TEK_0$  is the current traffic encryption key (TEK) and  $b \in \{0, 1\}$  specifies the DC scheme currently being used.

<sup>1</sup>We omit the details about the public verification key which consists of the tuple  $(y, h_1, h_2, \dots, h_n)$  and is for the purpose of verifying the validity of partial decryptions. They are not necessary here since the partial decryptions are produced by the center who is the data source.

---

**Procedure** Member  $i$  Aggregation**Input:** Aggregation message  $m$ , member key  $x_{bi}$ .

1. Generate  $IV \leftarrow_R \mathbb{Z}_q$  and  $ak_i = H(x_{bi}||IV)$ .
2. Send to the center  $M_i = (i||m||IV||\text{MAC}_{ak_i}(i||m||IV))$ .

**Procedure** Center Verify**Input:** Revoked set  $R$ , center key  $\Gamma$ , message  $M_i$ 

1. Parse  $M_i$  into  $M_i = (i||m||IV||tag)$ .
2. If  $i \in R$ , reject the message.
3. Recover  $x_{bi}$  via Lagrange interpolation:

$$x_{bi} = \sum_{j \in T^+} L_j(i)x_{bj} \text{ where } L_j(x) = \frac{\prod_{k \in T^+, k \neq j} (x-k)}{\prod_{k \in T^+, k \neq j} (j-k)}.$$

4. Compute the authentication key by  $ak_i = H(x_{bi}||IV)$ .
5. If  $tag = \text{MAC}_{ak_i}(i||m||IV)$ , accept the message. Otherwise reject it.

Fig. 2. Procedures for (a) member aggregation, and (2) center verification.

---

### D. Multicast

The center encrypts data using  $TEK_0$ , the current traffic encryption key. To guarantee data authenticity, it attaches its signature on the data. Each member stores the center’s public key that can be used to verify the signature.  $TEK_0$  and public key signature are enough to guarantee confidentiality and authenticity of the multicast data. The only overhead on a member is the the center’s public key <sup>2</sup>.

### E. Aggregation

As mentioned in Section IV-A, we can use existing key structure in the DC cryptosystem to authenticate aggregate messages. The authentication is via a Message Authentication Code (MAC) with a key derived from her share of the secret key  $x_{bi}$ . The derivation can be done via a cryptographically secure pseudorandom number generator (PRNG) with  $x_{bi}$  (and maybe some other randomness) as seed. In practice, it can be as simple as using a cryptographically secure hash function. Let  $H$  be such a hash function. Member  $i$  first generates a random initialization variable  $IV$  of sufficient length (e.g.  $IV \leftarrow_R \mathbb{Z}_q$ ) and the authentication key  $ak_i$  is computed as  $ak_i = H(x_{bi}||IV)$ . The randomness-like property of the hash makes the output look independent of the input, avoiding the possible vulnerability of using the same key in multiple cryptographic operations. This is a common practice found in many key manage schemes such as [18], [19], [16] etc., and the SSL protocol. The center, who is in possess of the polynomial that generates  $x_{bi}$ , can recover  $x_{bi}$  and generate  $ak_i$  to verify the MAC. The detailed protocol is described in figure 2.

### F. Rekeying

The rekeying protocol is summarized in figure 3. Essentially it uses DC system to multicast a new TEK such that the

<sup>2</sup>There are schemes for authenticating multicast data that require minimal buffering on the sender side and resist packet loss (e.g., [12], [13], [14], [15]). Some achieved great efficiency by making additional assumptions such as weak time synchronization between sender and receivers [12], [16]. However, as [17] showed, the general multicast message authentication problem is equivalent to digital signatures. We take the general approach here and leave the rest as options for applications.

---

**Procedure** Center Rekey**Input:** Revoked set  $R$ , center key  $\Gamma$ .

1. Select randomly  $T' \subseteq T$  such that  $|T'| = t - |R|$  and set  $\bar{T} \leftarrow T' \cup R$ .
2. Generate a new TEK, denoted  $TEK_1$ , and compute  $c = E_{y_b}(TEK_1||\delta||s)$  where  $\delta \leftarrow_R \mathbb{Z}_q$  and  $s = 1$  if  $|R|$  is close to  $t$  and 0 otherwise.
3. Multicast the following rekey message:  

$$M = (c, \{(j, D_{x_j}(c)) | j \in \bar{T}\}, E_{TEK_0}(\bar{T}))$$
4. If  $s = 1$ , set  $b \leftarrow \bar{b}$  and carry out Refresh as a separate procedure asynchronously. Also reset  $R$  to  $\emptyset$ .
5. Update the public keys by  $h_b \leftarrow h_b g^\delta \text{ mod } p$ , and it shares of the private key by  $x_{bj} \leftarrow x_{bj} + \delta \text{ mod } q, \forall j \in R \cup T^+$ .

**Procedure** Member  $i$  Rekey**Input:** Rekey message  $M$ , member key  $x_{bi}$ .

1. Decrypt  $M$  using  $x_{bi}$  and obtain  $TEK_1$ .
2. Set  $TEK_0 \leftarrow TEK_1$  to be the current group key.
3. If  $s = 1$ , set  $b \leftarrow \bar{b}$ .
4. Increment  $x_{bi}$  by  $\delta$ , i.e., set  $x_{bi} \leftarrow x_{bi} + \delta \text{ mod } q$ , and update the public key by  $h_b \leftarrow h_b g^\delta \text{ mod } p$ .

Fig. 3. Procedures for (a) center rekeying, and (b) member rekeying.

revoked members cannot access. This is achieved via DC system’s built-in revocation capability. We added the “increment by  $\delta$ ” step which essentially achieves “in-place update” of  $DC_b$ . It shifts the DC scheme by a random (but secret) amount and a newly admitted member has no information about past DC scheme even if it has not been completely refreshed. As will be shown in Section V this feature guarantees backward security which is lacking in the original DC cryptosystem. And this is carried out via multicast and is very efficient.

### G. Refresh

This procedure is used to refresh  $DC_{\bar{b}}$  when it has approached its limit on the number of revocations. The center generates a fresh DC cryptosystem as it does in the Initialization stage and sends the new keys to each member via secure unicast channel <sup>3</sup>. This process is “off-line” compared to multicast and aggregation and can be performed anytime before it needs to run again. All operations in the group communication, including data communication and rekeying etc., can proceed uninterrupted using  $DC_b$ .

### H. User Join

When a user is allowed to join the communication, the center selects an unused index  $i$  and sends her all information corresponding to her index as specified by the key generation procedure. At this point we distinguish two types applications. The first allows the newly admitted members to access previous multicast data during the same session (i.e., the data encrypted with the  $TEK_0$ ). In this case the center simply sends  $TEK_0$  along with the key information to the new member via secure unicast channel. No rekeying message is necessary. The second type requires a strict backward security and prohibits the new member from accessing any

<sup>3</sup>These secure unicast channels can be constructed, for example, by letting the members initiate a key agreement protocol using the aggregation mode as described in Section IV-E. The center does not need to store all public keys of the members.

data prior to her admission. For this type of applications the rekey procedure is invoked before the center sends the key information to the new member. We will show later that this scheme guarantees the strict backward security.

### I. User Departure

To remove a member  $i$  from the group, the center recovers her share of the private key  $x_{bi}$  and adds  $i$  to  $R$ . It then invoke the Rekey procedure.

## V. ANALYSIS AND EVALUATION

In this section we evaluate our protocol in terms of security and efficiency. Since our protocol incurs constant cost in terms of all resources (storage, communication and computation) for each rekey operation, the evaluation here will be mostly analytical. The actual cost of any application is dependent solely on the group dynamics.

### A. Security

The security of our protocol is summarized in the following theorem whose proof is given in Appendix .

*Theorem 1:* Assuming the security of the DC cryptosystem, the MAC algorithm and the symmetric key encryption scheme used for data encryption, the protocol described above satisfies the 8 properties specified in Section II-A.

### B. Efficiency

The biggest advantage of our scheme is the feature that each member only needs to store 1 constant-sized key which is truly independent of the group size. This can be significant for systems where the members are limited in storage. The center is required to keep a key of length  $O(t)$  to be able to authenticate all  $n$  members which is much less than the  $O(n)$  storage overhead required for maintaining data authenticity in aggregation using other schemes. Note that the aggregation MAC can also service as a *membership* authentication. This is because, due to the security of the secret sharing scheme, the MAC algorithm and the cryptographic hash, only legitimate member with  $x_{bi}$  can produce such tag. And in our scheme, verifying the MAC only requires the use of center's own key (in contrast, in a digital signature-based scheme such as identity-based signature schemes, the center must retrieve the sender's public key to authenticate the message and verifying membership requires consulting an active member list). This means that, unlike other schemes, there is no "hidden cost" in our scheme: the information to authenticate aggregation messages (and memberships) is all contained in the center's key and the center does not even have to store a list of user IDs online.

The communication complexity for each rekey message is  $O(t)$ , also independent of the the group size. This is a useful feature not only from a scalability point of view, but also for the purposes of hiding group dynamics information, as will be explained in Section VI.

Rekeying involves public key encryption/decryption which are relatively inefficient to compute compared to symmetric

key schemes. When instantiated with Shoup and Gennaro's TDH2 threshold scheme [11], both encryption and decryption involve  $O(t + 1)$  exponentiations. For encryption, since the bases  $g, \bar{g}$  are fixed (per public key), one can pre-compute a table to speedup the operations significantly. For decryption, what is really needed is the *product* of the exponentials. The cost can be reduced considerably by using techniques such as simultaneous multiple exponentiations (Chapter 14.6.1 in [20]). Moreover, the computation is highly parallelizable. Each exponentiation is independent of another and can be carried out concurrently. This is true for both the center and the members.

Another advantage of our scheme is that authenticating aggregation messages uses symmetric key crypto. The efficiency obtained is critical for large-scaled systems supporting a large number of members who generate data at a high rate (e.g. a large IDS). Using the latest Crypto++ benchmark (<http://www.eskimo.com/~weidai/benchmarks.html>), HMAC, a popular MAC algorithm with proven security [21], features a throughput of 216.674 MB/s. This is superior to any public key crypto-based schemes.

## VI. HIDING GROUP DYNAMICS INFORMATION

In an INFOCOM '04 paper [22], Sun and Liu first raised the issue of protecting group dynamics information (GDI) in multicast. They showed in [22] that in many schemes, the key management messages can disclose information about the dynamics of group membership such as group size and the user join/departure rate. This information can be used by an adversary or competitor in a way that is detrimental to the multicast group. [22] outlined two techniques an attacker can use to obtain GDI. The first technique applies to schemes where the attacker can distinguish rekey messages caused by user join from those resulted from user departure: he can conclude, with high certainty, that a user joins the group when he sees a join rekey message and a user leaves the group when a departure rekey message is observed. The second technique estimates the group size from the rekey message size when the two are related. The first attack is typically mounted by a group member who can decrypt the rekey messages and the second one can be launched by an outsider observing the traffic. [22] showed that most tree-based centralized key management schemes (e.g. [23], [24], [25], [5], [26]) are vulnerable to both attacks. Variants of these attacks are also effective on some other schemes. Please see [22] for details.

As observed in [22], the fundamental cause of their vulnerability is that most key management schemes allow an inside attacker to separate the rekey messages for user join and those for user departure and/or produce rekey message whose size is dependent on the group size. [22] proposed using batch rekeying and phantom users, who join and leave the group artificially, to augment existing schemes to conceal GDI. However, batch rekeying is only possible when it is acceptable to delay updating the session key in face of group membership change. Phantom users causes much overhead in terms of communication and computation. Neither technique is

free and completely satisfactory. In particular, the effectiveness of phantom users is totally dependent on the artificial group size (which is the total number of real and phantom users), denoted  $N_0$  as in [22]. The approach proposed in [22] is “all-or-nothing” in that if the actual group size,  $N$ , never exceeds  $N_0$ , it is perfectly hidden. Should  $N$  becomes greater than  $N_0$  for some time, the adversary can obtain accurate estimate on  $N$  for the entire duration when  $N > N_0$ . To actually conceal group size, a large number of phantom users must be used to ensure  $N_0$  is greater than the maximum number of users the group can ever accommodate. This is a serious problem for highly dynamic groups and the approach does not provide a really scalable solution. Yet there seems no better way out for schemes where the rekey message size is dependent on group size. This is one of their inherent limitations.

If we only consider the leakage by multicast and key management messages, as is done in [22]<sup>4</sup>, we can show that our scheme has some nice properties that make it more affable to hiding GDI. First of all, it is completely immune to attacks on GDI mounted by *outsiders*. This is due to the following two features: (1) the only information in the rekey message that reveals GDI is the blacklist  $\bar{T}$  and it is encrypted with current session key; and (2) the size of rekeying messages is independent of group size. Even if an outsider can observe the rekey messages, he won't be able to differentiate user join events from user departure, nor can he obtain any information about the size of the group.

As for the insider's attack, we first observe that, while revealing GDI to *outsider* is hardly ever acceptable, in some applications, it is actually beneficial to disclose group dynamics information to legitimate nodes of the system. For example, in some network, knowing the compromised nodes allows the good ones to avoid routing messages towards them. In this case, our scheme does not need to change at all. The blacklist  $\bar{T}$  that is part of the rekey message provides a natural vessel for conveying the information about who have been evicted to legitimate members.

If GDI has to be hidden even from legitimate members, we introduce the following modifications. Note that our scheme as described so far already guarantees that the message size is independent of the group size even to an insider. All we need to fix is to “mix” the events of user join with departure.

- 1) We use batch rekeying and invoke the Rekey procedure only when the system has accumulated some number of join/departure users.<sup>5</sup>
- 2) In initialization, the center selects a large number  $N_0$  greater than the upper bound of the group size.  $\mathbb{Z}_{N_0}$  will

<sup>4</sup>Other leakages are possible, e.g. via the initialization. They are common to all schemes and fixing them is orthogonal to securing the multicast messages. Besides monitoring all other traffic is substantially harder than listening to the multicast channel, which can easily be done by joining the group. And even if such monitoring is possible, it is often possible to “bury” group dynamics-related non-multicast messages in other traffic. Therefore we take a similar approach as [22] and focus on multicast management messages.

<sup>5</sup>The consequence of this modification is that the forward and backward security properties are only guaranteed at a coarser granularity, determined by the intervals between the Rekey operations.

be the space where user indices are drawn. For  $s = 0, 1$ , each of the initial set of users will be assigned a unique but random index from  $\mathbb{Z}_{N_0}$ . Note that each member gets independent indices for  $DC_0$  and  $DC_1$ .

- 3)  $\mathbb{Z}_{N_0}$  for each DC scheme is divided into 3 disjoint sets. For  $DC_s$ , where  $s = 0, 1$ ,  $A_s$  is the set of indices currently assigned to active members.  $R_s$  is the set of indices belonging to the users whose membership have been revoked using the current DC scheme.  $MP_s$  is the “mixing pool” whose purpose will be clear later. During Initialization  $A_0$  and  $A_1$  are initialized according to the index generation process,  $R_0$  and  $R_1$  are all set to  $\emptyset$ .
- 4) The blacklist  $\bar{T}$  for each rekey message is constructed as usual by including  $R_b$  and subset of  $MP_b$ <sup>6</sup>. Each rekey message will set  $s = 1$  which causes the system to switch to the other DC scheme.
- 5) When switching to a new DC scheme, the center sets  $b \leftarrow \bar{b}$  and  $R_b \leftarrow \emptyset$ .
- 6) When a user joins the group, for  $s = 0, 1$ , the center randomly selects a number  $i_s$  from  $MP_s$  as her index for  $DC_s$  and generates the key information accordingly. It then sets  $MP_s \leftarrow MP_s \setminus \{i_s\}$ ,  $A_s \leftarrow A_s \cup \{i_s\}$ .
- 7) When a member, with indices  $i_b$  and  $i_{\bar{b}}$  for  $DC_b$  and  $DC_{\bar{b}}$  respectively, leaves the group, the center sets  $A_b = A_b \setminus \{i_b\}$ ,  $A_{\bar{b}} = A_{\bar{b}} \setminus \{i_{\bar{b}}\}$ ,  $R_b = R_b \cup \{i_b\}$ ,  $MP_{\bar{b}} = MP_{\bar{b}} \cup \{i_{\bar{b}}\}$ .

*Theorem 2:* The modified protocol hides GDI from an insider.

*Proof:* (Sketch) First because of the batch rekeying, user join and departure are not distinguishable from rekey messages. Second, the system switches to a fresh DC scheme after each rekey message and the set  $R$ , whose elements will appear in  $\bar{T}$ , will be completely independent for the next rekey operation. An adversary cannot build her knowledge of revoked members, or the new members by observing multiple rekey messages. ■

With these modifications, the costs for member storage, member computation, rekey communication all remain the same. The only added overhead is for the center to maintain these sets. We can use the following to mitigate the cost.

First, if the center has enough storage capacity, these sets can be maintained as linked lists to minimize computation cost. Although the storage is now  $O(n)$ , the computation remains  $O(t)$ .

If, on the other hand, the center is limited on storage, these sets can be maintained as follows. For each DC scheme, the center maintains 2  $N_0$ -bit numbers,  $N_A$  and  $N_R$ . The membership of an index  $i$  with respect to the 3 sets is encoded by the two bits  $N_A[i]$  and  $N_R[i]$  where  $X[i]$  means the  $i$ th bit of  $X$  in a binary representation. We define the following encoding:  $i \in A$  if  $N_A[i] = 1$ ,  $i \in R$  if  $N_R[i] = 1$  and  $i \in MP$  if  $N_A[i] = N_R[i] = 0$ . Note when a user leaves the group, her indices are all known and updating the sets amounts to simply flipping bits at known locations. The most

<sup>6</sup>Note that the center does not maintain set  $T$  anymore.

expensive computation is when a user joins the group and the center needs to find an unused index in  $MP$  for her. For this purpose the center generates a random number  $i$  in the range of 1 to  $N_0$ . If  $N_A[i] = N_R[i] = 0$ , then  $i$  is assigned to the user and the sets are updated accordingly. Otherwise the center tries another random number. The probability that the center cannot find an unused index in  $MP$  after  $k$  tries is  $(\frac{n}{N_0})^k$ . Assuming  $N_0$  is at least twice of  $n$ , with high probability, the center can find an unused index in  $MP$  with just a few tries. The computation can still be bounded by  $O(t)$ .

In any case the center storage complexity becomes  $O(n)$ . However, we point out that this is the same for all other secure group communication schemes if they are used for bidirectional communications even without any consideration for protecting GDI. And in our 2nd scheme, the constant behind the  $O$  is much smaller: the center in our scheme as just described only needs to store 4  $N_0$ -bit numbers while in all other schemes must maintain at least  $n$   $l$ -bit numbers where  $l$  is the length of the key used.

## VII. RELATED WORK

In early secure multicast schemes (e.g., Group Key Management Protocol (GKMP) [27]), the center shares a pairwise key with each group member and distributes group keys to group members on a one-to-one basis. For obvious reasons this cannot scale to large groups. Some work has been done to improve the scalability of such schemes. Among the efficient solutions, the Logical Key Hierarchy (LKH) (or Key Graph) was independently discovered in [24] and [5] and has been an inspiration for many subsequent works [25], [28], [29], [30], [31], [32]. In these schemes, individual and auxiliary keys are organized into a hierarchy and each group member is assigned to a leaf and holds all the keys from its leaf to the root. The root key is shared by all group members and used as the TEK. New TEK is distributed by encrypting it with keys that deleted members do not have. So far  $O(\log n)$  seems to be the best storage (for both center and members) and communication complexity the LKH-based schemes achieved.

Asymmetric key cryptosystems are also used for multicast key management. This includes the work in cryptography such as *traitor tracing*, a concept introduced by Chor, Fiat and Naor [33], and *broadcast encryption*, initiated by Fiat and Naor [34], and a recent multicast encryption framework formalized by Duan and Canny [3]. These schemes can be used to distribute the new TEK in the rekeying operation.

All these schemes focus on achieving private communication in the multicast direction. Little attention was paid to aggregation. As we mentioned before, the global TEK is not enough for data authentication in aggregation and these schemes do not have other provisions for such purpose.

There are some techniques that can be used for authenticating aggregation messages and they appear to enjoy reasonable scalability. For example, identity-based signature (IBS) [35], [36] allows one to verify another party's signature using the signer's identity information (e.g. email address) as the public key. It appears that, when used to for authenticating

aggregation message, IBS allows the center to store 0 key since the verification keys are members' IDs. However, in a realistic group communication application, the center not only needs to verify that a message comes from an alleged sender, it also needs to ensure that the alleged sender is indeed a valid member of the group. In order to do so, the center needs to keep a list of valid member IDs online and compares each signature against it. This makes the effective center storage  $O(n)$ . In contrast, our scheme eliminates such overhead: group membership is self-authenticating through the aggregation data authentication tag. Besides IBS systems rely on a trusted key generation center (KGC) for extracting each user's private key, adding another layer of complexity.

## REFERENCES

- [1] W. Fenner, "Internet group management protocol, version 2," RFC-2236, November 1997.
- [2] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *RAID 2001*. Springer-Verlag, 2001, pp. 85–103.
- [3] Y. Duan and J. Canny, "How to construct multicast cryptosystems provably secure against adaptive chosen ciphertext attack," in *RSA Conference 2006, Cryptographers' Track*, ser. Lecture Notes in Computer Science, vol. 3860. Springer-Verlag, 2006, pp. 244–261.
- [4] D. Liu, P. Ning, and K. Sun, "Efficient self-healing group key distribution with revocation capability," in *CCS 2003*.
- [5] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. on Networking*, vol. 8, no. 1, pp. 16–30, 2000.
- [6] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean, "Self-healing key distribution with revocation," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2002, p. 241.
- [7] H. Wang, "Resilient lkh: Secure multicast key distribution schemes," in *Proceedings of the 2003 International Workshop on Advanced Developments in Software and Systems Security (WADIS)*, 2003.
- [8] W.-G. Tzeng and Z.-J. Tzeng, "A public-key traitor tracing scheme with revocation using dynamic shares," in *PKC '01*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2001, pp. 207–224.
- [9] Y. Dodis and N. Fazio, "Public key trace and revoke scheme secure against adaptive chosen ciphertext attack," in *PKC '03*, ser. Lecture Notes in Computer Science, vol. 2567, 2003, pp. 100–115.
- [10] C. H. Kim, Y. H. Hwang, and P. J. Lee, "An efficient public key trace and revoke scheme secure against adaptive chosen ciphertext attack," in *ASIACRYPT 2003*, ser. Lecture Notes in Computer Science, vol. 2894. Springer-Verlag, 2003, pp. 359–373.
- [11] V. Shoup and R. Gennaro, "Securing threshold cryptosystems against chosen ciphertext attack," *J. Cryptology*, vol. 15, no. 2, pp. 75–96, 2002.
- [12] A. Perrig, R. Canetti, D. Tygar, and D. Song, "Efficient authentication and signature of multicast streams over lossy channels," in *IEEE Symposium on Security and Privacy*, May 2000, pp. 56–73.
- [13] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," in *CCS '99*. ACM Press, 1999, pp. 93–100.
- [14] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts," Tech. Rep., 1998.
- [15] R. Gennaro and P. Rohatgi, "How to sign digital streams," in *CRYPTO '97*, ser. Lecture Notes in Computer Science. Springer-Verlag, 1997.
- [16] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient and secure source authentication for multicast," in *NDSS01*, 2001.
- [17] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in *Eurocrypt '2001*, ser. Lecture Notes in Computer Science, vol. 2045. Springer-Verlag, 2001, pp. 437–452.
- [18] A. Perrig, R. Szwedczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, 2002.
- [19] M. Shehab, E. Bertino, and A. Ghafoor, "Efficient hierarchical key generation and key diffusion for distributed sensor networks," in *IEEE SECON 2005*, 2005.

- [20] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, ser. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, 1996.
- [21] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1996, pp. 1–15.
- [22] Y. Sun and K. R. Liu, "Securing dynamic membership information in multicast communications," in *Proceedings IEEE Infocom'04*.
- [23] M. J. Moyer, J. R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *IEEE Network Magazine*, vol. 13, no. 6, pp. 12–23, November/December 1999.
- [24] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," IETF Request For Comments, RFC 2627, June 1999.
- [25] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *INFOCOM'99*, 1999.
- [26] 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*, vol. 17, no. 9, pp. 1614–1631, September 1999.
- [27] H. Harney and C. Muckenhirn, "Group key management protocol (gkmp) architecture," IETF Request for Comments, RFC 2094, July 1997.
- [28] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure internet multicast using boolean function minimization techniques," in *Proceedings IEEE Infocom'99*, 1999.
- [29] C. K. Wong and S. S. Lam, "Keystone: A group key management service," in *International Conference on Telecommunications, ICT 2000*, 2000.
- [30] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch rekeying for secure group communications," in *Proceedings of the tenth international World Wide Web conference on World Wide Web*, Orlando, FL USA, 2001, pp. 525–534.
- [31] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A scalable group re-keying approach for secure multicast," in *IEEE Symposium on Security and Privacy*, 2000, pp. 215–228.
- [32] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, "Reliable group rekeying: a performance analysis," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001, pp. 27–38.
- [33] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*. Springer-Verlag, 1994, pp. 257–270.
- [34] A. Fiat and M. Naor, "Broadcast encryption," in *Proceedings of the 13th annual international cryptology conference on Advances in cryptology*. Springer-Verlag New York, Inc., 1994, pp. 480–491.
- [35] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*. Springer-Verlag New York, Inc., 1985, pp. 47–53.
- [36] J. C. Cha and J. H. Cheon, "An identity-based signature from gap diffie-hellman groups," in *PKC '03*. Springer-Verlag, 2003, pp. 18–30.
- [37] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, pp. 612–613, November 1979.

## APPENDIX

*Proof:* Assuming the security of the DC cryptosystem (which has been proven in [3]), the MAC algorithm and the symmetric key encryption used to encrypt the data, for an adversary to successfully "attack" our scheme, she must obtain enough information to recover the appropriate key(s), where "attack" means gaining access to data in multicast and forging messages in aggregation. We show that for any attempted attack on any of the properties, this is impossible. Let  $ATK_i$  denote the attack on property  $M_i$  and  $A_i$  where  $i = 1, 2, 3, 4$ .

**ATK1** We show that an outsider cannot obtain any member's key by observing network traffic. There are three types of traffic in our protocol: key distribution traffic, data traffic, and

rekey traffic. The first is via secure unicast channels and the second is encrypted with the current TEK. Both, by definition, provides confidentiality to those without proper keys. The rekey traffic is characterized by messages as specified in figure 3. This is essentially the ciphertext as defined in DC cryptosystem [3], with the blacklist encrypted with the current TEK and is shown to be secure against adaptive chosen ciphertext attacks. Therefore the attacker cannot get the keys to mount attack on  $M.1$  or  $A.1$ .

**ATK2** When a member  $i$  is removed from the group, according to our protocol, two events will happen: (1) the rekey procedure will be invoked and (2) her index  $i$  will appear, and remain, in the center's revocation list  $R$  until next time the system switches to the other DC scheme when  $R$  will be reset to  $\emptyset$ . The rekey procedure essentially utilizes the revocation capability of DC cryptosystem and guarantees that  $i$  will not be able to obtain new traffic encryption keys as long as  $i$  remains in  $R$  or a fresh DC scheme is used (to which  $i$  does not have access). Note that in our protocol, the rekey message that causes the transition to a new DC scheme will also cause all remaining members to update the DC scheme that will be used for next rekey operation (i.e., the increment of private keys by  $\delta$  in the last step on both the center and members. See Section IV-F). Since the revoked member does not have access to this rekey message, she cannot obtain  $\delta$  thus her information about the new DC scheme is obsolete and cannot enable her to decrypt future rekey messages even though  $R$  is reset to  $\emptyset$ . This guarantees forward confidentiality (M.2).

As for forward authenticity (A.2), note that when verifying the MAC, the center checks the sender's index against  $R$ . When  $i$  is in  $R$ , her MAC will always be rejected. After  $R$  is reset to  $\emptyset$ , both her private keys  $x_{0i}$  and  $x_{1i}$  are no longer valid any more because  $x_{bj}$  has been updated by  $\delta$  and  $x_{\bar{b}j}$  has been completely refreshed in an off-line process for all other legitimate members. Therefore the center will reject all her MACs keyed using her old keys. This guarantees forward authenticity (A.2).

**ATK3** Attack on collusion freedom is modelled by allowing the adversary to control up to  $t$  members. Confidentiality in multicast (i.e., M.3) comes directly from the security of DC cryptosystem [3]. Aggregation authenticity (A.3) in face of up to  $t$  colluding members is guaranteed due to similar property of the secret sharing scheme [37].

**ATK4** When a user is added to the group and assigned the index  $i$ , the center first invokes the rekey procedure. The user is then given her keys in both  $DC_0$  and  $DC_1$  and the current TEK via secure unicast channel. This allows her to access current multicast data and future rekey information. Note that the latest rekey procedure prior to her admission updates the current DC cryptosystem for existing members. This means the keys she received via the unicast channel will not enable her to decrypt previous rekey messages which is the only means to distribute TEKs. Therefore she cannot have access to data multicasted before she is admitted (M.4).

A.4 is implied by A.3. ■