1998 International Computer Symposium
Workshop on Cryptology and Information Security
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

# A SCALABLE HIERARCHICAL CRYPTOGRAPHIC PROTOCOL FOR SECURE MULTICAST *

Longsong Lin† Lih-Chyau Wuu, Bei-Li Wang
Department of Electronic Engineering
National Yunlin University of Science and Technology
Touliu, TAIWAN, 640 R.O.C.

## Abstract

*There has been a growing trend on supporting security service over Internet. Providing security-enhanced multicast service for private conference could be even problematic due to that the network is full of non-trusted nodes, it is hard to achieve a safe registration procedure without incurring the implosion of confirmation messages, it is easy to trap into a wait deadlock when members performing peer validation, and the key distribution procedure is not scalable when distributing a single key to multiple members. In this paper, we identify these problems in each steps associated with providing the security-enhanced multicast service for private conferencing. We present a cryptographic protocol that employs core-based and source-based tree algorithms to construct a hierarchical tree and several security mechanisms to provide authentication, integrity, and confidentiality services. The protocol uses a pre-authorization scheme to construct trusted registrars in the multicast network to avoid incurring confirmation implosion, a decentralized validation procedure to prevent deadlock in peer-validation, cryptographic algorithms to keep message exchanges intact and confidential, and a hierarchically procedure to make key distribution scalable.*

## 1 INTRODUCTION

It has been a growing trend to provide security services for Internet for group-based applications to protect, safeguard, and validate the information between communicating parties. Such group-based network applications often takes advantage of the multicast service of the Internet, as the availability of the MBONE [1] architecture widely spread in Internet. A typical example is a multiparty private conferencing application that requires security functions in multicast services [2] to protect the integrity of traffic from modifications, guard for confidentiality of communication from electronic eavesdrop, and validate participant's authenticity.

A private multiparty conferencing system is a multicast application that provides participation-restricted, security-enhanced functions for registration and authentication of participant, and integrity and confidentiality for the data communication. The practice of such a private multiparty conferencing involves five basic phases: initialization, registration, validation, communication, and key distribution. A multiparty conferencing system allows several conference sessions to take place simultaneously, one for each party. A session holder initiates a session and holds a long-term secret key for that session, called *session key*. It is the session holder who decides how a registration procedure will proceed later. The session holder determines himself along or delegates some agents to assist him to register new joining users. If successfully completing the registration, the user is granted the session key and a certificate signed by the session holder, which becomes a participant of the conference. The session key will be used to encrypt messages to be sent to other participants, whilst the certificate will be used to authenticate the participant's status. During the session proceeds, each participant periodically sends a validation request message containing this certificate to the session for other participants to authenticate him. As time goes by, the session key might be susceptible to malicious intruders; hence, it is often practical to periodically change and distribute the session key.

In view of the above procedures, conducting a private multicast conference requires the provision of functions such as authentication of joining users during registration phase, authentication of participants during validation phase, traffic confidentiality in communication phase, and secure session key delivery in key distribution phase. To support these security functions, various cryptography mechanisms are utilized to ensure participant authentication, traffic integrity and confidentiality. Specifically, digital signature schemes are required for authorizing certificates [3] in the registration phase, authentication procedures [4] are required to validate participants in the validation phase, message digesting and cryptographic algorithms [5] are required to guarantee traffic integrity and confidentiality in the communication phase, and finally confiden-

tial key distribution methods [6] are required in the key distribution phase.

The essential concerns for embedding security functions in multicast are the scalability and validation delay problems. Multicast schemes can be generally classified into two categories: the source-based tree [7] schemes that construct one tree for each source and the shared tree schemes [8] that build a single tree to share with all multicasting sources. As an example for the first category, Deering's work extended IP service model to accommodate multicast capability and allow hosts to arbitrarily join and leave a multicast group. Lately works by Deering and Cheriton [9] and Macedonia and Brutzman's [10] also fall into this category. Since the source based tree protocols, for examples MOSFP [11] and DVMRP [12], suffer from the problem such as the scalability, more bandwidth requirement, and soft-state maintenance effort, CBT [13] and PIM [14] protocols are innovated for scalable, simple-to-implement multicast routing protocols. The source-based tree is able to optimize the delay via shortest path tree algorithm; the validation delay between participant could be minimized. On the other hand, the shared-based tree can offers smaller total tree cost and better scalability; it is suitable for constructing a scalable, security-eccentric tree that minimizes the total threat of the tree.

Although multicast routing being a very active research area, the enforcement of security services within the routing framework is just about to burgeon. Some works have been done that successfully merged the techniques from both respects into frameworks. A discussion about the security threats and countermeasures associated with multicast is first found in Ballardie and Crowcroft's work [15]. The first work addressing the use of distributed registration and key distribution (DiRK) in the context of multimedia over Internet Mbone is found in Oppliger's work [6]. DiRK has founded the basic framework for registration and key distribution corresponding to different levels of security requirements; still, some critical problems and issues that deserve further investigation are perceived. First, during participant registration phase, it may be required to provide more trusted registrars if more restrict privacy is required. Secondly, a joining user may repeatedly encounter a rapid burst of confirmation messages with respect to his registering request; resulting in a performance deterioration. Thirdly, the registration validation phase would likely to produce deadlock during the recursive waiting for validating a series of participants' certificates. Also, in the key distribution phase, using present session key to encrypt a new key could be fragile; as long as the initial session key is deciphered, the subsequent new keys will be obvious. Finally, the key distribution scheme should be scalable as the number of participant increases.

Another important issue related to security services for multicast is associated with the key distribution. Group key management problem has been addressed in [16], in which a group key management protocol (GKMP) is proposed to distribute the burden of key distribution on a group of members. The scalability issue is addressed extensively in Ballardie's work, RFC 1949 [17]. He proposed a scalable multicast key distribution (SMKD) scheme that uses CBT model to provide secure registration of a CBT group tree and as a result solves the scalability problem in multicast key distribution. SMKD assumes that the routers on the delivery tree are trusted; however, the authors advised that a core router should authenticate joining requests. This suggests that further work should be done on what authentication scheme between them would be and how to select the core routers. Also, although the author's idea of restricting key distribution capability to "trusted" routers implies that the level of trust should be considered in the multicast, there still needs to take a step further to explicitly design a "trusted" network by taking advantage of the hierarchical architecture of Internet.

In the paper we explicitly identify some critical problems encountered in providing security services for private multicast conferencing. These problems are stated as follows:

1. *Trusted registrar*: Who will be the trusted registrar responsible for authorizing certificates and distributing session keys? Delegating agent is a feasible solution; Nevertheless, improper selection of agents will weaken the security of the system. In SMKD, it is assumed that all routers on a delivery tree are trusted and will not misbehave, so any router can join to be a core router delegating the session holder. In DiRK, an user can register as an active participant; consequently, he is eligible to register new users and provide for the session key on session holder's behalf. So, both methods allow arbitrary node to join as a delegating agent for session holder. Obviously, the number of agents could increase out of the session holder's control as active requests grow, which becomes a serious problem when executing authentication, as a result of more security holes.

2. *Confirmation implosion*: During registration phase, active participants who receive the registration request from a joining user may simultaneously return registration confirmation messages to the joining user - an implosion of confirmation messages will occur at each joining origin.

3. *Recursive certificate authentication*: During the validation phase, an active participant who wants to verify any participant's certificate would need to wait for the validation messages from all the participant's ancestors until he receives the session holder's certificate. If there are several validation requests issued simultaneously, the recursive waiting in active participants will likely result in a wait

deadlock.

4. *Confidential key distribution*: An encryption scheme is called computational secure if the time required to break the cipher exceeds the lifetime of the information [4]. For example, using the massively parallel computing power it is easy to break the 56-bit DES key in hours. If the initial session key $K_S$ is broken within the lifetime of the initial session key, the private conferencing it could be not computational secure because the subsequent new keys can be decrypted easily.

5. *Scalable key distribution*: The complexity of session key distribution is subject to the growth of network size. Using single key distribution center for distributing keys does not offer a scalable solution. This is even complicated in sender-specific multicast, in which multiple session keys with one for each sender being distributed to its participants.

The paper proposes a hierarchical cryptographic multicast protocol that can be used to handle participants' registration, validation, communication, and key distribution in a private conferencing system. The design goal of the protocol is to achieve a secure, trusted registration, to avoid the confirmation implosion and recursive authentication problems, and to provide confidential, scalable key distribution scheme.

The paper is organized as follows. The next section will give a general description of relevant cryptography techniques and the procedure for generating the hierarchical tree. and Session 3 will be dedicated to the discussion of the multicast cryptographic protocol. Then, it follows the state diagram of the protocol and a demonstrative example. Finally a conclusion is drawn with a probe to future directions.

## 2 SECURITY-ENHANCED HIERARCHICAL MULTICAST TREE

The foundation of a secure protocol is based on cryptography; messages are encrypted and decrypted between peer communicating entities. Such protocols are also named as cryptographic protocol [18]. The cryptography notation given in Table 1 will be used throughout our discussion.

To take advantage of Internet architecture, we construct the multicast tree into a two-level hierarchical network. In Internet terminology, the backbone routers comprise the first-level zone, called transit zone, whereas routers connecting to any of the backbone routers comprise the second level zone, called stub zone. Likewise, we categorize the nodes in the first level of the multicast tree as core nodes, or simply *cores*, and the other nodes that connect to each of the

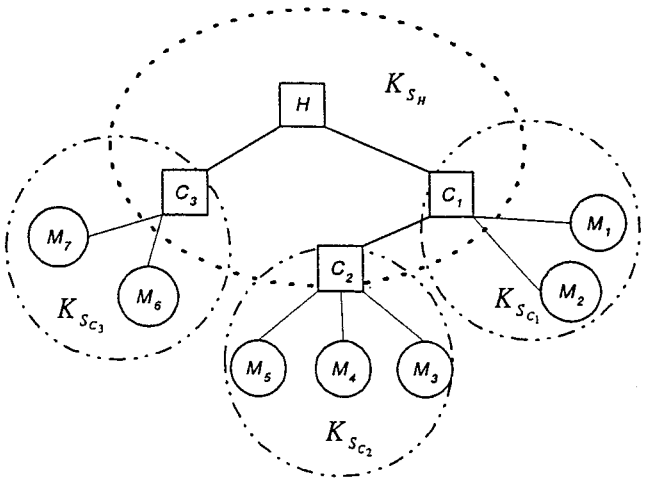


**Figure 1. A hierarchical multicast tree with trusted cores and members**

cores as member nodes, or simply *members*. A general form of such a multicast tree is depicted in Figure 1 in which intermediate non-member nodes are omitted for simplicity. A session in the conference is thus dissected into two subsessions in the hierarchy. The session held by the session holder $H$ for a group of cores is called *core subsession* $S_H$. The session holder authorizes these cores as trusted nodes for the conference and delegates each core $C_i, i = 1, ..., n$, to hold a *localsubsession* $S_{C_i}$ and process registration requests. During registration phase, $H$ distributes session key $H_{S_H}$ to the cores in the same manner as core $C_i$ distributes local session key $K_{S_{C_i}}$ to its members. Any message exchange between members in different local subsections or between members and cores will involve a series of pair-wise encryption and decryption using the session keys $K_{S_H}$ and $K_{S_{C_i}}$ to ensure traffic integrity and confidentiality.

To maintain the authenticity of the participants in the conference, we employ two types of certificates during the registration phase. In the upper level of the hierarchical tree, the session holder issues *core registration certificates* $H\langle\langle C_i, S_H\rangle\rangle = \langle H, C_i, S_H, k_{c_i}\rangle k_h^{-1}$ to a trusted core $C_i$ ; thereafter, the core $C_i$ can issue a *member registration certificate* $C_i\langle\langle M_j, S_{C_i}\rangle\rangle = \langle C_i, M_j, S_{C_i}, k_{m_j}\rangle k_{c_i}^{-1}$ to a legitimate member $M_j$. The certificates are shown in Table 2.

The procedure for constructing the multicast tree like Figure 1 is explained as follows. The multicast tree is constructed during the phase of participant registration. The tree generation procedure employed in core subsession is different from that used in local subsessions. A subtree for the core subsession is constructed by using a CBT-like scheme [19] in which a group of cores are connected to form a trusted core tree in order to minimize the cost of the tree. In contrast, the subtree for local subsession is formed by using a

DVMRP-like scheme [20] which minimizes the path delay from the member to the corresponding core. As shown in Figure 1 the $H - C_1 - C_2 - C_3$ a minimum cost tree whereas each local subtree is a shortest path tree (SPT). The primary concern of using this mixture approach is justified in our previous work [21] that the core-base tree approach can provide better scalability and is simple to implement whilst the source-base tree can achieve shortest delay form each member to a source.

For a core to register with session holder, it sends a registration request that will set up transient joining state in each router it traverses. This transient state eventually times out unless it is confirmed with a join acknowledgement from upstream routers. The acknowledgement message traverses the reverse path of the corresponding join request message. Once it reaches the router that originated the join request message, the registration complete and the new participant can receive traffic sent to the session.

For a member to register with a core, it sends a registration request message that will record the addresses of upstream routers to the source so that in the reverse-path forwarding [22] phase the previous hop is always known to the node who receives the message in the reverse path. On routing the request message, the intermediate nodes will simply perform routing using conventional routing protocol, e.g., DVMRP: they will not be allowed to join the conference and hence will not need to perform authentication for the request message. During the request confirmation phase, the reverse-path forwarding technique is employed to construct the multicast tree, taking the advantage of recorded routing information.

# 3 THE CRYPTOGRAPHIC MULTICAST PROTOCOL

The cryptographic multicast protocol consists of the following phases: session initialization, registration, communication, validation, and rekeying.

## 3.1 Session Initialization

To initialize a session S, a session holder $H$ first authorizes the session key $k_H$ to a set of nodes C's as candidates, which can register with the $H$ as legitimate cores later. The $H$ then randomly generates public key pair $(k_h, k_h^{-1})$ and announces $[\mathrm{S}, k_h]k_H^{-1}$ to the channel S.

$$H \rightarrow \mathrm{S} : INIT(\mathrm{C}, k_H)$$

$$H \Rightarrow \mathrm{S} : INIT([\mathrm{S}, k_h]k_H^{-1})$$

Note that because the session identifier S and the short-term private key $k_h$ are encrypted with the $H$'s long-term public key $k_H$, only those who have been pre-authorized with $k_H$ can decrypt this announcement. Also, it should be emphasized that $[\mathrm{S}, k_h]k_H^{-1}$, rather than $\langle \mathrm{S}, k_h \rangle k_H^{-1}$, is used in this announcement. Since this announcement is for nodes to join as a core, we enforce the degree of trust by restricting the eligibility of extracting S and $k_h$ to only those nodes that have been authorized the $k_H$ by $H$. Note that the S and $k_h$ are sent in a encrypted form of $[\mathrm{S}, k_h]k_H^{-1}$, rather than in clear text $\langle \mathrm{S}, k_h \rangle k_H^{-1} = \mathrm{S}, k_h, \{\mathrm{S}, k_h\}k_H^{-1}$, in order to enforce the confidentiality in key distribution. Also note that the symbol $\Rightarrow$ in the above equation represents broadcasting. For more definitions on message transmissions, reader may refer to Table 3.

**Table 1. Communication Primitives**

| $\rightarrow$ : Unicast | $\mapsto$ : Multicast | $\Rightarrow$ : Broadcast |
|---|---|---|

## 3.2 Participant Registration

The registration phase takes place both in between cores as well as in between the members and cores. In the former case, a node $C_i$ who had $H$'s session key sends a registration request REG_REQ to $H$ by unicast. The REG_REQ includes $\langle k_{c_i} \rangle k_{C_i}^{-1}$ so that $H$ can authenticate $C_i$'s message and extract $C_i$'s short-term public key $k_{c_i}$. After receiving these registration requests, $H$ will select some from the nodes that has been pre-authorized as the legitimate trusted cores and then broadcast a registration confirmation message REG_CONF to S. The REG_CONF message includes the identity $C_i$, the destination address of $C_i$, its certificate $H\langle\langle C_i, S_H \rangle\rangle$, and a secret key $K_{S_H}$ that is encrypted by $C_i$'s short-term public key $k_{c_i}$.

$$C_i \rightarrow H : REG\_REQ(\langle k_{c_i} \rangle k_{C_i}^{-1})$$

$$H \Rightarrow \mathrm{S} : REG\_CONF(C_i, H\langle\langle C_i, S_H \rangle\rangle, \{K_{S_H}\}k_{c_i})$$

In the latter case, the registration is issued from joining users to the legitimate trusted cores. User $M_i$ sends the core $C_j$ a registration request REG_REQ that contains a $k_{m_i}$ encrypted by $M_i$'s long-term private key $k_{M_i}^{-1}$. After receiving the request, $C_j$ broadcasts a registration confirmation to S. The REG_CONF message contains $M_i$, destination address $M_i$, its certificate $C_j\langle\langle M_i, S_{C_j} \rangle\rangle$ and a secret key $K_{S_{c_j}}$ encrypted by $M_i$'s short-term public key $k_m$.

$$M_i \rightarrow C_j : REG\_REQ(\langle k_{m_i} \rangle k_{M_i}^{-1})$$

$$C_j \Rightarrow \mathrm{S} : REG\_CONF(M_i, C_j\langle\langle M_i, S_{C_j} \rangle\rangle, \{K_{S_{C_j}}\}k_{m_i})$$

Some critical problems related to the participant registration phase are identified here: the trusted registrar, the confirmation implosion, and the hop-by-hop authentication problems. First, allowing arbitrary user to register as the session holder's delegate seems a natural approach; however, it is not secure enough if the private conference requires restricted security guarantee, and it would be hard for the holder to secure his

session as the number of delegates increases. The problem is even more severe in the core-based tree scheme if the core routers responsible for key distribution are not trusted nodes. Then, who should be the trusted nodes responsible for authenticating and registering new participants? We resolve this problem by using a pre-authorization scheme to ensure that the session holder can control and secure the number of trusted registrars. A session holder will first authorize his session key $k_H$ to a group of nodes, which then become the eligible candidates for being cores; then it selects a number of nodes from these authorized nodes as the cores to delegate it for registering new participants.

Secondly, when allowing arbitrary registrar to delegate the session holder, it is likely that a joining user will receive extra confirmation messages from the registrars that received his joining requests. This will result in a burst of traffics within a short period in the joining origin – a so-called *confirmation implosion problem*. Although a simple mechanism as random timer can be used for the confirming registrar to impose a random time before sending the confirmation messages, it still cannot avoid producing such excessive confirmation messages into the network. In our scheme, a joining node will choose the one and only one core to register among the legitimate trusted cores, so it will receives only one confirmation message from the core.

Finally, unlike SMKD that requires all the routers along the tree to join the conference and to perform corresponding authentication hop-by-hop during joining phase, our scheme will not allow the intermediate routers to join the conference for security consideration. Instead, the authentication will be performed on the end-to-end basis using specific unicast transmission via the best path to a specific target (session holder or cores). This will avoid the inefficiency by using hop-by-hop authentication repeatedly, which will induce more security holes due to the join of the intermediate routers and overhead in soft-state maintenance in these routers.

### 3.3 Message Communication

Once registered, the participant can communicate with the other members in the conference. When a core $C$ wants to multicast data to the members, it first multicasts the data to the core subsession by encrypting the data with secret key $K_{S_H}$. After receiving and decrypting the data with the secret key, each core $C_i$ encrypts the data again using the key $S_{C_i}$ of the local session and multicasts it to the local members.

$$C \mapsto S_H : COMM(\{Data\}K_{S_H})$$

$$C_i \mapsto S_{C_i} : COMM(\{Data\}K_{S_{C_i}}), \forall i$$

When a member $M$ wants to send a data, it will first encrypt it with the subsession key $S_C$ and multicast to the local subsession. After the core $C$ of the local subsession receives the data, it decrypts the data with the same secret key and multicasts the data encrypted with session key $K_{S_H}$ to the other cores, which will then deliver the data to its local member subsequently.

$$M \mapsto S_C : COMM(\{Data\}K_{S_C})$$

$$C \mapsto S_H : COMM(\{Data\}K_{S_H})$$

$$C_i \mapsto S_{C_i} : COMM(\{Data\}K_{S_{C_i}}), \forall i$$

### 3.4 Membership Validation

During the conferencing, a participant is required to send validation messages periodically to other participants for them to verify his participation. A participant sends a validation request message to the session from which he anticipate for a confirmation message. A validation message contains identify, certificate, and a timestamp $T$ – all digitally signed by the private key that corresponds to the public key included in the certificate. Validation requests are issued either from cores or from members and the validation confirmations are replied from session holder or cores respectively.

$$C_i \rightarrow H : VAL\_REQ(\{C_i, H\langle\langle C_i, S_H\rangle\rangle, \{T\}k_{c_i}^{-1}\}k_h)$$

$$H \mapsto S_H : VAL\_CONF(\{C_i, H\langle\langle C_i, S_H\rangle\rangle, \{T\}k_h^{-1}\}K_{S_H})$$

$$C_j \mapsto S_{C_j} : VAL\_CONF(\{C_i, H\langle\langle C_i, S_H\rangle\rangle, \{T\}k_{c_j}^{-1}\}K_{S_{C_j}}), \forall j.$$

When a core $C_i$ is willing to request for validation, it uses $k_h$ to encrypt $C_i$'s identity, certificate, and an encrypted timestamp $T$ into a VAL_REQ message and send it to the session holder. The session holder extracts $k_c$ from $C_i$'s the certificate, decrypts $T$ with $k_c$, and compare the result with a timestamp $T'$, which is taken from local clock. The status of the participant is valid only if the $T' - T$ is within a certain threshold value. After authenticating $C_i$'s status, the session holder multicast to all cores a VAL_CONF message encrypted with $K_{S_H}$. Remind that a legitimate core will hold a public key $k_h$ obtained in the initialization phase, and a secret key $K_{S_H}$ obtained in the registration phase. With these two keys, core $C_j$ can decrypt the message and decipher for the timestamp $T$ in the VAL_CONF. After $C_j$ authenticates the $C_i$'s status, it will forward the confirmation message to its local members.

$$M \rightarrow C_j : VAL\_REQ(\{M, C_j\langle\langle M, S_{C_j}\rangle\rangle, \{T\}k_m^{-1}\}k_{c_j})$$

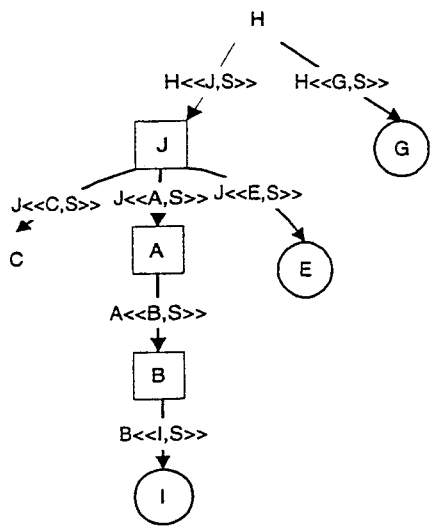


**Figure 2. CAT tree formed by using arbitrary registrar. The square marks a core while the circle marks a member.**

$$C_j \mapsto S_H : VAL\_CONF(\{M_i, C_j\langle\langle M_i, S_{C_j}\rangle\rangle, \{T\}k_{c_j}^{-1}\}K_{S_H}).$$

In contrast, when a member $M$ wants to send a validation request, instead of sending to the session holder as in the previous case, it first sends the validation message to the core $C_j$ from which the member has obtained the certificate $C_j\langle\langle M, S_{C_j}\rangle\rangle$. After the core validates $M$'s status, it multicasts a VAL_CONF message to the core subsession. Every core in the subsession then forwards the confirmation message to its local members.

### 3.4.1 Certificate Authorization Tree

Consider the following case when a joining user $I$ is registered by an arbitrary user $B$, who has been registered by $A$. The registration relation can be recorded by a tree data structure called *certificate authorization tree* (CAT). Figure 2 shows the result of the registration that allows arbitrary registrar during registration phase where $B\langle\langle I, \mathrm{S}\rangle\rangle$ represents the certificate issued to $I$ by $B$ for session S and $A\langle\langle B, \mathrm{S}\rangle\rangle$ to $B$ by $A$, and so on.

Such scheme could easily trap into a situation called *recursive authentication*, which is explained as follows. Assuming that $B$ issues a validation request message VAL_REQ, which is captured by an arbitrary member $I$. If $I$ wanted to validate $B$, he would have to extract the $k_b$ from $B$'s certificates $A\langle\langle B, \mathrm{S}\rangle\rangle = \langle A, B, \mathrm{S}, k_b\rangle k_a^{-1}$. To authenticate $k_b$, $I$ must to verify the digital signature that is associated with $A\langle\langle B, \mathrm{S}\rangle\rangle$. But in order to verify this signature, $I$ must have $A$'s public key $k_a$. Hence, $I$ waits for $A$'s VAL_REQ message from which he can extract $k_a$, but again, in order to make sure that $k_a$ is authentic, $I$ must verify the digital signature that is associated with this VAL_REG message. Node $I$ is thus pending until it gets the public key $k_j$ that is certified with $H$'s private key $k_h^{-1}$. Now, since $k_h$ is publicly available, $I$ can extract $k_j$ from $H\langle\langle J, \mathrm{S}\rangle\rangle = \langle H, J, \mathrm{S}, k_j\rangle k_h^{-1}$, then extract $k_a$ from $J\langle\langle A, \mathrm{S}\rangle\rangle = \langle J, A, \mathrm{S}, k_a\rangle k_j^{-1}$ contained in $A$'s VAL_REQ, and finally use $k_a$ to validate the $A\langle\langle B, \mathrm{S}\rangle\rangle = \langle A, B, \mathrm{S}, k_b\rangle k_a^{-1}$, which completes the $B$'s validation request. In fact, any one who wants to validate $B$ will have to wait for the validation requests of all participants in the path from $B$ to $H$. This recursive validation could result in *wait deadlock* [24] when simultaneous requests are waiting for each other. Furthermore, a bottleneck exists in the network when conference participants distributes densely and join to a specific node, node $J$ in the figure for example; the concentration on validation traffic will prevent the conference from being scalable.

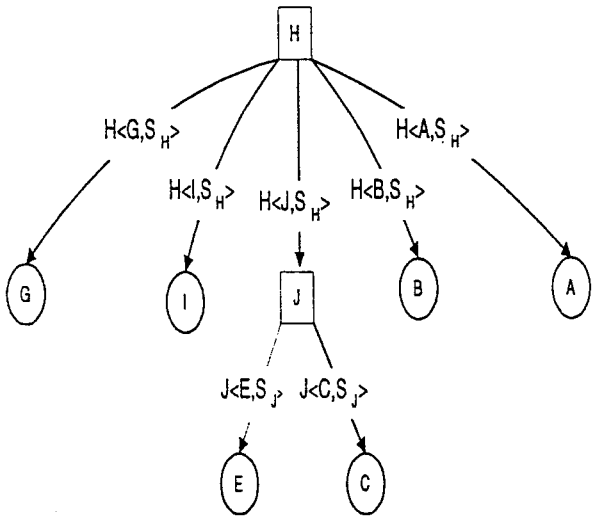


**Figure 3. Two-level hierarchical certificate authorization tree CAT.**

To cope with this problem, the validation procedure is decentralized by dint of selecting trusted core registrars and delegating them to validate its members. The result from employing our registration process is shown in Figure 3. With the CAT, the subsequent validation process can be made simple in two hierarchical stages, avoiding the recursive authentication problem. Specifically, $E$'s validation request can be verified by the core $J$ because it authorizes $E$'s certificate $J\langle\langle E, S_J\rangle\rangle = \langle J, E, S_J, k_e\rangle k_j^{-1}$ and hence surely can extract the public key $k_e$. Moreover, since each validation request message VAL_REQ is sent via a shortest path to a corresponding core in which the request is validated, only one end-to-end validation is involved, which greatly reduces the traffic incurred in the net-

work when using hop-by-hop authentication schemes.

### 3.5 Change Key and Redistribution

In private conferencing, session key used for a long time will not guarantee confidential key distribution. If the initial session key $K_S$ is used to encrypt a new session key $K'_S$, and similarly $K'_S$ to $K''_S$, and so on, a malicious attacker may take as much time as he can to decipher the initial session key. Once succeeded, he is able to obtain the subsequent new keys in short time, rendering the key distribution easily breakable. In our scheme, the session holder will encrypt the new session by using each core's public key. The REK_CORE message is multicast to the core subsession in which each core use the corresponding private key to extract the new session key between cores. For every local subsession, the core uses each member's public key to encrypt its new session key $K'_{S_C}$, which will be extracted with corresponding private key by each member.

$$H \mapsto S_H : REK\_CORE(\langle T, C_1, \{K'_{S_H}\}k_{c_1}, C_2,$$

$$\{K'_{S_H}\}k_{c_2}, .....C_m, \{K'_{S_H}\}k_{c_m}\rangle k_h^{-1})$$

$$C_i \mapsto S_{C_i} : REK\_LOCAL(\langle T, M_1, \{K'_{S_{C_1}}\}k_{m_1},$$

$$M_2, \{K'_{S_{C_1}}\}k_{m_2}, .....M_m, \{K'_{S_{C_1}}\}k_{m_m}\rangle k_{c_i}^{-1})$$

Another critical problem in key distribution is related to the scalability to the growing number of participants. An entity responsible for generating and distributing session keys for a multicast group is called key distribution center (KDC). It must be able to authenticate members and distribute a session key to them. This involves encrypting the relevant message $n$ time, once with the existing session key shared between KDC and corresponding member. The SMKD scheme is scalable for its accommodating multiple core routers as candidates for more key distribution centers. Taking a step further, we use hierarchical tree using trusted first-level cores as KDCs to take charge of authentication and key distribution for participants in the second level of the tree hierarchy. Task of key distribution is decentralized into from session holder to cores and from cores to members.

## 4 CONCLUSION REMARKS

In this paper, we have identified several essential issues related to providing security service for private multicasting conference. The proposed protocol uses pre-authorization scheme to provide trusted registrars in the network. These registrars are distributed in a hierarchical way so that the participant can locally register to corresponding cores; therefore, the confirmation implosion and recursive validation problems are avoided and session keys can be distributed to all participants in a scalable way. The hierarchical multicast tree built with both the core-based-tree approach and the source-based-tree offers such advantages as scalability, easy-to-implement, etc. Although the security services in the multicast context are self-contained in its primal form, attentions should be paid to how to connect these services to other standard security mechanisms such as directory service and certification authority.

## References

[1] M. Macedonia and D. Brutzman, "MBONE, The Multicast BackbONE," Available from http://www.cs.ucl.uk/mice/mbone_review.html.

[2] Rolf Oppliger, "Interner Security: Firewalls and Beyond," *Comm. Of ACM*, Vol. 40, No. 5, 92-102, 1997.

[3] CCITT Data Communication Networks Directory (Blue Book). Recommendation X.509, Authentication Framework.

[4] William Stallings "Network and Internetwork Security" Prentice-Hall, 1995.

[5] R. Rivest, "The MD-5 Message Digest Algorithm," *SRI Network Information Center*, 1992.

[6] Rolf Oppliger, "Distributed registration and key distribution (DiRK)," Avaliable from http://iamwww.unibe./ch/ oppliger/dirk.html.

[7] A. Ballardie, "Core Based Trees (CBT) Multicast routing Architecture," Available : Internet RFC 2201, http://ds.internic.net/rfc/rfc2201.txt.

[8] D. W. Wall, "Mechanisms for broadcast and selective broadcast," Ph.D. dissertation, Stanford University, June 1980.

[9] Deering, S. and Cheriton, D.R. "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, 8(2), 85-110.

[10] Macedonia, M.R. and Brutzman, "Mbone Provides Audio and Video Across the Internet," *IEEE Computer*, 27(4), 30-36.

[11] J. Moy, (1994, May) "MOSPF, analysis and experience." Available: Internet RFC 1585, http://ds.internic.net/rfc/rfc1585.txt.

[12] T. Pusateri. Distance-Vector Multicast Routing protocol (DVMRP) version3. Working draft, Febuary 1996.

[13] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees: An architecture for scalable interdomain multicast routing," in *Proc. ACM SIGCOMM*, Ithaca, NY, Sept. 1993, pp. 85-95.

**Table 2. Cryptographic symbol and definition**

| | |
|---|---|
| S | The channel of the conference, including all sessions |
| $S_H$ | Session between cores held by H |
| $S_{C_i}$ | Local session held by core $C_i, i = 1, ...n$. Particularly, $S_{C_H}$ is the core session held by $H$ |
| $K$ | Secret key |
| $K_{S_H}$ | Session key for core session $S_H$ |
| $K_{S_{C_i}}$ | Session key for local session $S_{C_i}$ |
| $(k, k^{-1})$ | Public/private key pair. Capital in subscription stands for long-term keys, e.g. $k_H$, and small letters are short-term keys, e.g., $k_h$. |
| $\{m\}K$ | message $m$ is encrypted with secret key $K$ |
| $\{m\}k$ | message $m$ is encrypted with public key $k$ |
| $\{m\}k^{-1}$ | digital signature on message using privated key $k^{-1}$ |
| $\langle m\rangle k^{-1}$ | Clear-text message concatenates with digital signature on message digest $h(m)$ using one-way collision-resistant hashing function $h$ to ensure data integrity. $\langle M\rangle k^{-1} = m, \{h(m)\}k^{-1}$ |
| $[m]k^{-1}$ | Encrypted message concatenated with digital signature on message digest to ensure confidentially and integrity. $[m]k^{-1} = \{m\}k^{-1}, \{h(m)\}k^{-1}$ |

**Table 3. Registration Certificates**

| | |
|---|---|
| $H\langle C_i, S_H\rangle, i = 1, 2, ....., n$ where $H\langle C_i, S_H\rangle = \langle H, C_i, S_H, k_{c_i}\rangle k_{c_i}^{-1}$ | Core registration certificate issued by $H$ to $C_i$ for session $S_H$. |
| $C_i\langle M_j, S_{C_i}\rangle, i = 0, 1, ....., n; j = 1, 2, ....., \lvert M\rvert$ where $C_i\langle M_j, S_{C_i}\rangle = \langle C_i, M_j, S_{C_i}, k_{m_j}\rangle k_{c_i}^{-1}$ | Member registration certificate issued by $C_i$ to $M_j$ for session $S_{C_i}$. |

[14] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An architecture for wide-area multicast routing," in *Proc. ACM SIGSOMM*, London, UK, Aug. 1994, pp. 126-135.

[15] A. Ballardie and J. Crowcroft, "Multicast Specific Security Threats and Counter-Measures," in *ISOC Symposium on Network and Distributed System Security*, February 1995.

[16] H. Harney, C. Muckenhirn, (1997, July) "Group Key Management Protocol (GKMP) Architecture." Available: Internet RFC 2094, http://ds.internic.net/rfc/rfc1949.txt.

[17] A. Ballardie, (1996, May) "Scalable Multicast Key Distribution." Available: Internet RFC 1949, http://ds.internic.net/rfc/rfc1949.txt.

[18] D. Maughan, M. Schertler, M. Schneider, and J. Turner. "Internet Security Association and Key Management Protocol (ISAKMP)." Draft-ietf-ipsec-isakmp-08.txt, July 1997.(work in progress).

[19] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "Protocol independent multicast (PIM): Protocol specification," Internet Draft RFC, jan, 11, 1995.

[20] D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma, and A. Helmy. "Protocol independent multicast-dense mode (PIM-DM): Protocl specification." Work in progress. ftp://netweb.usc.edu/pim.

[21] L. Lin and B. Wang "Hybrid Tree Generation Algorithms for Hierarchical Multicast" Tech. Rep. 10(7):981-989, National Yunlin University of Science and Technology, 1997.

[22] Y.K. Dalal and R.M. Metcalfe, "Reverse path forwarding of broadcast packets," *Communications of the ACM*, 21(12): 1040-1048, 1978.

[23] Andrew S. Tanenbaum, "Distributed operating Systems." Prentice-Hall, 1995.