Journal of Computing and Information Technology - CIT 17, 2009, 2, 203–214
doi:10.2498/cit.1001215

Analysis of Key Management Schemes for Secure Group Communication and Their Classification

R. Aparna[1] and B. B. Amberker[2]

[1] Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumkur, Karnataka, India
[2] Department of Computer Science and Engineering, National Institute of Technology, Warangal, Andhra Pradesh, India

Secure Group Communication is very critical for applications like board-meetings, group discussions and teleconferencing. Managing a set of secure group keys and group dynamics are the fundamental building blocks for secure group communication systems. Several group key management techniques have been proposed so far by many researchers. Some schemes are information theoretic and some are complexity theoretic in nature. Users in the secure group may negotiate with each other to derive a common group key or may compute the group key on their own. Some schemes involve a trusted Key Distribution Center (KDC), which generates and distributes initial pieces of information, whereas in other schemes users themselves select their private information. Storage at each user and communication cost among members of the group vary from scheme to scheme. Here, in this paper, we discuss some of the key management schemes proposed earlier, based on the considerations mentioned above. We also analyze the schemes with respect to storage, communication and computation costs.

Keywords: secure group communication, secure group key, key distribution scheme, contributory key agreement, privileged user set

1. Introduction

The advancement of digital technologies and widening Internet bandwidth have increased the demand for new multimedia services. Some of the service types include video-on-demand, group discussions, board meetings, etc. There are many users who participate in such services. The communication among the users pertaining to any such service must be carried out confidentially.

Other scenarios may be out of the n users in a network, some $t(t \ll n)$ of them would like to discuss on a common concern. These t parties termed *privileged users* must communicate themselves over a public channel in confidence and others must not be able to listen to the conversation amongst these t parties. Hence, there is a need to find new technology for such confidential communication termed as *Secure Group Communication* or *Secure Conferencing*.

A Naive solution is to have a shared key between every pair of users, which leads to storing $(n-1)$ keys with each user. To send a message, sender must encrypt the message to each user in the group separately. This increases the amount of storage at each user-end; computation and communication costs are also increased. Another possible solution would be to have a publicly available directory, called Public Key Infrastructure (PKI). Here, user will have a *(private, public)* key pair, private key is known only to that user, whereas public keys of all n users are stored in the publicly available directory. To send a message, sender uses public keys of all other group members to encrypt the message, so that other users can decrypt using their own private keys. This leads to $(n-1)$ encryptions to send a single message and any non-group member can try to act as privileged user and send the message after encrypting it with public keys of privileged users. Since these default solutions require huge amount of storage, communication and computation, they do not fit for group communication scenario. Hence, the general goal of Secure Group Communication is to establish a common *secret key*, called *Secure group key* or *Secure conference key* among privileged users for confidential communication.

Many group key management techniques have been proposed so far. Some schemes are information theoretic in nature and some are complexity theoretic. There may be a trusted *Key Distribution Center* (KDC) involved in the system which generates and distributes initial private pieces of information to users in the group. Group key may be generated and distributed by the KDC or group members may compute the group key, either interactively or non-interactively. Based on the involvement of KDC, key management techniques can be classified into two categories: (i) *Key Distribution Scheme* and (ii) *Contributory Key Agreement Scheme*. Key distribution scheme [4, 7, 20, 19, 16, 13, 18] involves a single entity termed as trusted KDC. There exists a pair-wise secure channel with the KDC and every group member. KDC is responsible for generating and distributing keys to all the members in the group via secure channel. It is also responsible for generating the group key and distributing it to all the users in the group.

In key distribution scheme, after receiving the initial information from KDC, secure group key can be derived by the members in the *privileged set*, either *asynchronously* or *interactively*. In asynchronous mode, each member in the privileged group can derive a common group key on his own for secure conference with the help of information obtained by KDC. In interactive mode, users in the privileged group can communicate with each other to set up secure conference key. However, a KDC incurs management overhead and is a single point of failure.

In contrast to key distribution scheme, contributory key agreement schemes [6, 5, 10, 1, 2, 14, 11] require every group member to contribute an equal share to the common secret key. Secret key for secure group communication is computed as a function of all members contribution.

Once a conference (group key) is set up, users in the group can communicate with each other in secured manner. Since the group is dynamic, membership in the group may change over time, i.e., new members may join the group and existing members may leave the group. Group membership can change because a single member may join/leave the group or a set of members may join/leave the group simultaneously. Set of members join/leave scenario may arise because of the changes in *network connectivity*. These changes in network connectivity may be due to network partitions (which divides the group) and network re-heals (which combines several groups together). A new member can always join the group voluntarily, but the member in the group may leave the group either voluntarily or involuntarily (being forced by other members). Involuntary exit may be due to processor crash or network disconnect. Whenever there is a membership change, group key must be changed to prevent a new user from reading past communications, called *backward access control* and a departed user from reading future communications, called *forward access control*.

In key management schemes, an *adversary* may try to eavesdrop on the conversation of the privileged user set. An adversary may be an insider (member of among n parties, but not a member of privileged set) or outsider (member of other than n parties). A group of users, termed *malicious parties* may collude with each other and try to derive the common group key. The security of the group key management scheme is based on number of colluding parties. Scheme is termed as k-secure if it is not possible to derive the common key even after k non-group (non-privileged) members collude with each other, where k is termed as *threshold*.

The cost of the group key management scheme is determined by three factors: *communication, storage* and *computation*, amongst which, communication and computation costs are very important. Usually, efficiency in one comes at the expense of the other. Schemes that require minimum computation usually require more communication among group members, while schemes minimizing communication require more computations.

Many researchers have made surveys on group key management schemes [2, 15, 9]. In [2], performance evaluation has been made on five distributed key management techniques. Rafaeli and Hutchison classify the schemes in [15] into centralized, decentralized and distributed key management protocols. In [9], Judge and Ammar concentrate on providing security for multicast content distribution in group key management.

In this paper, we focus on the performance analysis of different key management protocols proposed by Blundo et al. [4], Wong et al. [20], Diffie-Hellman [6, 17], Fiat-Naor [7] and Burmester-Desmedt [5], with respect to storage, communication and computation. We classify the protocols based on (i) method used to establish the group key i.e., whether they are based on key distribution scheme or contributory key

agreement scheme, (ii) role of KDC (active only during initial stage or active throughout) (iii) security (unconditional or conditional). Section 2 describes the protocols that come under key distribution scheme and Section 3 discusses key agreement protocols. We show the classifications in Section 4. In Section 5, we analyze the protocols with respect to storage, communication cost and computation cost. We conclude the paper in Section 6.

2. Key Distribution Schemes

As mentioned earlier, key distribution schemes are classified into asynchronous or non-interactive and interactive modes. In asynchronous mode, each member in the privileged group can derive a common group key on his own, for secure conference, with the help of information obtained by KDC. In interactive mode, users in the privileged group can communicate with each other to compute secure conference key as described below.

2.1. Non-interactive Key Distribution Schemes

2.1.1. Blundo et al. Conference Keying Protocol

Blundo et al. [4] have proposed a protocol to derive a common conference key. In this approach, a trusted off line server, which is active only at initial stage, distributes some information among a set of n users, ($User_1, User_2, \ldots, User_n$) so that any t of them can join and generate a secure group key asynchronously. Figure 1 depicts the Blundo et al. non-interactive k-secure t-conference protocol. It makes use of a symmetric polynomial in t variables (number of users in the group) of degree k (threshold) with coefficients over GF(q), $q > n$. In this scheme, t is fixed beforehand. Each join/leave restarts the protocol from initial stage.

Maximum number of coefficients in the polynomial are $\begin{pmatrix} k+t-1 \\ t-1 \end{pmatrix}$ out of which each user is required to store $\begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix}$ values from GF(q) and is required to perform at the most $t * \begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix}$ multiplications, $\begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix}$ additions and $k * t$ exponentiation operations.

2.1.2. Broadcast Encryption Zero Level Scheme

Fiat and Naor proposed a basic scheme [7] which allows users to determine a common key for every subset, which is resilient to any set S of size $\leq k$. In this scheme, for a given k each user is preassigned with a set of keys such that, once the users are told which of them are in privilege set and which are not, each user in the privilege set can construct the common key on his own. This scheme considers a set U which comprises all users i.e., $|U| = n$. For every set $B \subset U, 0 \leq |B| \leq k$, a key K_B is defined and K_B is given to every user $x \in U - B$. Let T denote the privileged user set. The common secret key to the privileged set T is obtained by performing exclusive-or of all keys K_B, $B \in U - T$.

- KDC picks at random a symmetric polynomial $P(x_1, \ldots, x_t)$ of degree k with t variables with coefficients over $GF(q)$, $q > n$.
- To each user $User_i$, $i = 1, \ldots, n$, in the system, KDC distributes the polynomial $f_i(x_2, \ldots, x_t) = P(i, x_2, \ldots, x_t)$, that is the polynomial obtained by evaluating $P(x_1, \ldots, x_t)$ at $x_1 = i$.
- If the users $User_{j_1}, \ldots, User_{j_t}$ want to set up a conference key, then each user $User_{j_i}$ evaluates $f_{j_i}(x_2, \ldots, x_t)$ at $(x_2, \ldots, x_t) = (j_1, \ldots, j_{i-1}, j_{i+1}, \ldots, j_t)$.
- The conference key for users $User_{j_1}, \ldots, User_{j_t}$ is equal to $s_{j_1,\ldots,j_t} = P(j_1, \ldots, j_t)$.

Figure 1. Blundo et al. non-interactive k-secure t-conference protocol.

The scheme is resilient to every coalition of up to k users, since, even if up to k users collude, they will all be missing key K_S. Hence they are unable to compute the common secret key. In this scheme, each user in the system is assigned with $\sum_{i=0}^{k} \binom{n-1}{i}$ keys and each user is required to perform $\sum_{i=0}^{k} \binom{n-t}{i}$ exclusive-or operations to obtain the common secret key.

2.1.3. Broadcast Encryption One Level Scheme

This scheme developed by Fiat and Naor [7], considers a family of functions $f_1, f_2, \ldots, f_l, f_i : U \longmapsto 1, \ldots, m$ with the property that for every subset $S \subset U$ of size k, there exists some $1 \leq i \leq l$ such that for all $u, v \in S : f_i(u) \neq f_i(v)$, that means the family of functions f_i contains a perfect hash function (defined in [8], [12]) for all size k subsets of U when mapped to the range $\{1, 2, \ldots, m\}$. This family can be used to obtain a k-resilient scheme from 1-resilient scheme.

For every $1 \leq i \leq l$ and $1 \leq j \leq m$, an independent 1-resilient scheme $R(i,j)$ is used. Every user $u \in U$ receives the keys associated with schemes $R(i, f_i(x)) \; \forall \; 1 \leq i \leq l$. Suppose the secret massage is M. To send this message to a privileged set $T \subset U$, KDC generates random strings $M^1, \ldots, M^l$ such that $\bigoplus_{i=1}^{l} M^i = M$. KDC broadcasts for all $1 \leq i \leq l$ and $1 \leq j \leq m$ the message M^i to the privileged subset $\{u \in T | f_i(u) = j\}$ using scheme $R(i,j)$. Every user $u \in T$ can obtain all the messages $M^1, \ldots, M^l$ and by performing exclusive-or of them can get the message M.

Every user receives $(n * l)$ number of keys from KDC where l denotes number of functions, hence the storage required at every user is $(n * l)$ and 3 messages are broadcast by KDC.

2.1.4. Keystone Architecture

This scheme assumes a trusted key server which is responsible for distributing the keys; and, the key server uses key tree [20, 19, 3] for group key management. A key tree is a directed acyclic graph with two types of nodes, U nodes representing members and k nodes representing keys. The nodes of the tree hold the keys at that particular level. The leaves of the tree correspond to group members and each leaf contains individual key of that member. The key present at the root of the tree is the group key. Key server is at the root of the tree. Keys at the intermediate levels of the tree are termed *auxiliary keys* and will be used to convey new group key to the members in an efficient manner. A member U is given key k if and only if there is a directed path from U-node u to k-node k in the key path.

If the tree is a balanced binary tree and if there are t users in the group, then the height h of the tree is $log_2 t$ and each member stores at most $(h+1)$ keys. In general, if d is the degree of the tree and if there are t users, then height of the tree is $log_d t$. Figure 2 depicts a binary tree structure for $t = 8$ users. K-nodes represent keys and U-nodes represent users. User U_1, for example, knows the keys K_1, K_{12}, K_{14}, and K where K is the group key, K_1 is U_1's individual key, K_{12} and K_{14} are *auxiliary keys* along the path from leaf to root.

For each join/leave operation, server changes keys along the path from leaf to root from the position where join/leave operation has occurred. For example, in Figure 2, if user U_7 leaves the group, then the keys K_{78}, K_{58} and K must be changed. Changed group key and auxiliary keys must be conveyed to remaining members in the group. New group key K' must be conveyed to users U_1 to U_6 and U_8, K'_{58} must be sent to users U_5, U_6, U_8 and K'_{78} to U_8. The following messages are constructed to convey changed keys:

(Message $\{K'\}_K$ indicates new key K' encrypted with key K)

For users U_1 to U_4 : $\{K'\}_{k_{14}}$

For Users U_5, U_6 : $\{K'_{58}\}_{K_{56}}$, $\{K'\}_{K'_{58}}$

For User U_8 : $\{K'_{78}\}_{K_8}$, $\{K'_{58}\}_{K'_{78}}$, $\{K'\}_{K'_{58}}$

Similarly, when it joins the group, keys at the same position are to be changed. The following messages are constructed:

For users U_1 to U_4 : $\{K'\}_K$

For users U_5, U_6 : $\{K'\}_K$, $\{K'_{58}\}_{K_{58}}$

For user U_8 : $\{K'\}_K$, $\{K_{58}\}'_{K_{58}}$, $\{K'_{78}\}_{K_{78}}$

For user U_7 : $\{K', K'_{58}, K'_{78}\}_{K_7}$

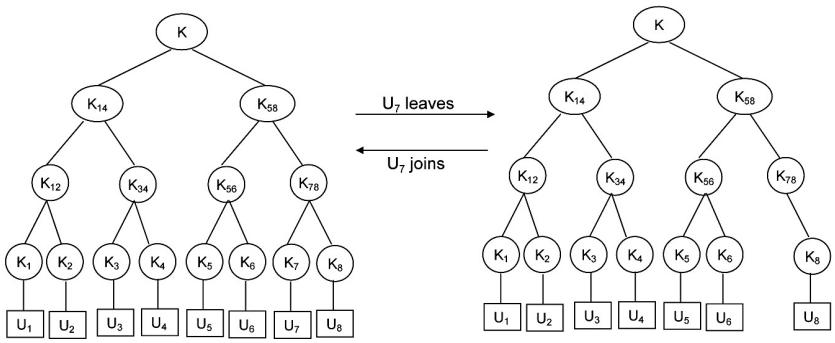

Figure 2. Tree structure for $n = 8$ users K-nodes represent keys and U-nodes represent users, K_1 to K_8 are user's private keys, K_{12}, K_{34}, K_{56}, K_{78}, K_{14}, K_{58} are auxiliary keys and K is the group key.

2.2. Interactive Key Distribution Schemes

2.2.1. Blundo et al. One-way Interactive Scheme

As in Blundo et al.[4] non-interactive scheme, here also trusted server chooses a symmetric polynomial, but in two variables of degree $(k + t - 2)$ with coefficients over $\mathrm{GF}(q), q > n$. Protocol in Figure 3 realizes an one-time interactive k-secure t-conference key distribution scheme.

The scheme is one-way interactive, since the t^{th} user $User_t$ among users $(User_1, User_2, \ldots, User_t)$ selects the common secret key and sends it securely to all the other $t-1$ users $(User_1, User_2, \ldots, User_{t-1})$. In this scheme, each user is required to store $(t + k - 1)$ elements of $\mathrm{GF}(q)$ and is required to perform at the most $(k + t - 2) * 2$ exponentiations and same number of multiplications and $(k + t - 2)$ additions and one exclusive-or operation. $User_t$ sends securely the group key to every other user in the group. Each join/leave operation re-initiates the protocol.

3. Contributory Key Agreement

3.1. Group Diffie-Hellman Key Agreement

In this scheme, secret key is never transmitted from one user to the other over the network. Group Diffie-Hellman Protocol in [17] general-

1. KDC chooses a symmetric polynomial $P(x, y)$ of degree $(k + t - 2)$ with coefficients over $GF(q)$, $q > n$.
2. To each user $user_i$, $i = 1, \ldots, n$, in the system, KDC distributes the polynomial $f_i(y) = P(i, y)$, that is the polynomial obtained by evaluating $P(x, y)$ at $x = i$.
3. If the users $User_{j_1}, \ldots, User_{j_t}$, where $j_1 < j_2 < \ldots < j_t$, want to set up a conference key, then:
   (a) $User_{j_t}$, randomly chooses a secret key s in $GF(q)$ (the value s is the conference key).
   (b) $User_{j_t}$ evaluates the polynomial $f_{j_t}(y)$ at $y = j_l$, for $l = 1, \ldots, t-1$, and, then, he computes temporary keys $s_{j_t,j_l} = f_{j_t}(j_l)$ (which is equal to $P(j_t, j_l)$).
   (c) $User_{j_t}$ sends to $User_{j_l}$ the value $\gamma_l = s_{j_t,j_l} \bigoplus s$, for $l = 1, \ldots, t-1$, where $\bigoplus$ is the bitwise ex-or.
   (d) For $l = 1, \ldots, t-1$: $User_{j_l}$, first computes $s_{j_t,j_l} = s_{j_l,j_t} = f_{j_l}(j_t)$ (which is equal to $P(j_l, j_t) = P(j_t, j_l)$). Then, $User_{j_l}$ computes s by taking the bitwise ex-or between s_{j_t,j_l} and the value γ_l received by $User_{j_t}$.

Figure 3. Blundo et al. one-time interactive k-secure t-conference protocol.

izes the basic 2 party Diffie-Hellman key exchange discussed in [6]. In Diffie-Hellman, two parties agree on a common key. Group Diffie-Hellman protocol considers t users, instead of two users. The group agrees upon a pair of primes q and g and starts calculating intermediate values. First member calculates first value, g^{r_1} and passes it to the next member. Each subsequent member in the group receives the set of intermediate values and raises them using its own contribution generating a new set. A set generated by j^{th} member will have j intermediate values with $j-1$ exponents and a *new value* comprising all the exponents. For example, the fifth member receives the set:

$$\{g^{r_2r_3r_4}, g^{r_1r_3r_4}, g^{r_1r_2r_4}, g^{r_1r_2r_3}, g^{r_1r_2r_3r_4}\}$$

and generates the set

$$\{g^{r_2r_3r_4r_5}, g^{r_1r_3r_4r_5}, g^{r_1r_2r_4r_5}, g^{r_1r_2r_3r_5}, g^{r_1r_2r_3r_4}, g^{r_1r_2r_3r_4r_5}\}$$

The *new value* in this set is $g^{r_1r_2r_3r_4r_5}$. The last member (t) can calculate the new value $K = g^{r_1r_2\cdots r_t}$. It raises all intermediate values to its contribution and multicasts it to the set. Each member extracts its respective intermediate value and calculates group key K. During the first phase, all the users send g^{r_i} i.e., one message is sent by each user. During the second phase, all the users send $g^{r_ir_j}, j \neq i, j = 1$ to t, i.e., $(t-1)$ messages are sent by each user. During the third phase, each user will send $\binom{t-1}{2}$ messages and so on. Hence, $\binom{t-1}{i-1}$ messages are sent by each user during i^{th} phase. Total number of communications required to set up group key $= t*\{1+\binom{t-1}{1}+\binom{t-1}{2}+\ldots+\binom{t-1}{t-1}\} = t*(2^{t-1}+1)$ and each user must perform $(2^{t-1}+1)$ exponentiations.

If a set of k members want to join the group, then a list of partial keys for each member of the group is generated and sent. Upon receiving partial keys each user uses its partial key to compute the common secret key. Usually, one particular member of the group (termed as controller) is responsible for generating and distributing these partial keys. Figure 4 shows a protocol to merge k members to the group and Figure 5 shows a partition protocol.

In merge protocol the current group controller (usually t^{th} user) removes its contribution from the current group key, generates a new token and passes it to one of the new members. The new member adds its own contribution to the token received by group controller and passes the token to the next new member. Every new member (except the last one) adds its own contribution and passes it in the same manner. When the

Assume that k members are added to a group of size t.

Step 1**:** M_t generates a new exponent $r'_t \in Z_q$, computes $g^{r_1\cdots r_{t-1}r'_t} \bmod p$, and unicasts the message to M_{t+1}

Step j+1 for $j \in [1, k-1]$: New merging member M_{t+j} generates an exponent $r_{t+j} \in Z_q$, computes $g^{r_1\cdots r'_t\cdots r_{t+j}} \bmod p$ and forwards the result to M_{t+j+1}.

Step $k+1$**:** Upon receipt of the accumulated value, M_{t+k} broadcasts it to the entire group.

Step $k+2$**:** Upon receipt of the broadcast, every member $M_i, \forall i \in [1, t+k-1]$, computes $g^{r_1\cdots r'_t\cdots r_{t+k}/r_i} \bmod p$ and sends it back to M_{t+k}.

Step $k+3$**:** After collecting all the responses M_{t+k} generates a new exponent r_{t+k}, produces the set $S = g^{r_1\cdots r'_t\cdots r_{t+k}/r_i} \bmod p | \forall i \in [1, t+k-1]$ and broadcasts it to the group.

Step $k+4$**:** Upon receiving the broadcast message, every member $M_i, \forall i \in [1, t+k]$ computes the key $K = (g^{r_1\cdots r'_t\cdots r_{t+k}/r_i})^{r_i} \bmod p = g^{r_1\cdots r'_t\cdots r_{t+k}} \bmod p$.

Figure 4. Group Diffie-Hellman merge protocol.

Assume that a set W of members is leaving a group of size t.

Step 1: The group controller M_d generates a new exponent $r'_d \in Z_q$, produces the set $S = g^{r_1 \ldots r'_d} \bmod p | M_i \notin W$, and broadcasts it to the remaining members in the group.

Step 2: Upon receiving S, every remaining member M_i, $\forall i \notin W$ computes the key $K = (g^{r_1 \ldots r'_d / r_i})^{r_i} \bmod p = g^{r_1 \ldots r'_d} \bmod p$.

Figure 5. Group Diffie-Hellman partition protocol.

token reaches the last new member, it broadcasts the token to the group without adding its contribution. From now on, this becomes the new group controller. Upon receiving the token, every group member (including new members), removes (factors out) its contribution and sends it to the new group controller (last new member). After receiving this information from all the members, new controller adds its own contribution to each and this represents a partial key. After computing all the partial keys, it is broadcast to the group. Every member gets the new group key by adding its contribution to the corresponding partial key that was broadcast.

In partition protocol, when a set of members leave the group, the most recent remaining group member is termed as the group controller. Group controller removes the corresponding partial keys of the leaving users from the list of partial keys, adds its new contribution after removing the previous contribution and broadcasts it to the group. Each remaining member in the group can compute the secret group key by adding its own contribution to the corresponding partial key in the list. For every join operation, $(t+3)$ messages are sent and each user must perform $(t+3)$ exponentiation operations. If any user leaves the group, one message is broadcast and each user performs one exponentiation operation, hence totally $(t-1)$ exponentiation operations are performed.

3.2. Burmester-Desmedt Scheme

Burmester and Desmedt proposed a protocol [5] which is very efficient and executes in only three rounds. Figure 6 shows the Burmester-Desmedt protocol. The same protocol is executed irrespectively of the type of membership change (i.e, whether it is join or leave). This scheme does not involve any group controller, computation is distributed among members of the group. Users U_1 through U_t with individual Diffie-Hellman exponentials $z_i = g^{r_i}$ will form a conference key $K = g^{r_1 r_2 + r_2 r_3 + \ldots + r_{t-1} r_t}$. Every user is required to perform $(t+1)$ exponentiations and $(t-1)$ multiplications and must broadcast $(2*t)$ messages.

If any user joins the group, $(2*t+2)$ messages are broadcast and 3 exponentiation operations are performed. For a leave operation, $(2*t-2)$ messages are broadcast and 3 exponentiation operations are performed.

Any group of $t \leq n$ users $(U_1, \ldots, U_t)$ $(t \ll n)$, derive a common group key K as follows:
(Every user U_i is aware of other $t-1$ members in the group)

1. Each U_i selects a random integer r_i, $1 \leq r_i \leq p-2$, computes $z_i = g^{r_i} \bmod p$, and sends z_i to each of the other $t-1$ group members.
2. Each U_i after receiving z_{i-1} and z_{i+1}, computes $X_i = (z_{i+1}/z_{i-1})^{r_i} \bmod p$ (i.e., $X_i = g^{r_{i+1} r_i - r_i r_{i-1}}$), and sends X_i to each of the other $t-1$ group members.
3. Upon receiving X_j, $1 \leq j \leq t$ excluding $j = i$, U_i computes $K = K_i$ as $K_i = (z_{i-1})^{t r_i} . X_i^{t-1} . X_{i+1}^{t-2} \ldots X_{i+(t+3)}^{2} . X_{i+(t-2)}^{1} \bmod p = g^{r_1 r_2 + r_2 r_3 + \ldots + r_{t-1} r_t} \bmod p$.

Figure 6. Burmester-Desmedt conference keying Protocol.

4. Classification

The schemes discussed above are deliberately classified based on the factors like (i) method used to establish the group key (ii) role of Key Distribution Center and (iii) security.

4.1. Based on Group Key Establishment

Based on the way group key is derived, the schemes can be classified into (a) Key Distribution scheme (b) Contributory Key Agreement scheme. In key distribution scheme, as we saw earlier, there is a key distribution center which is responsible for generating and distributing keys to all the group members via secure channel. KDC also generates and distributes initial group key to the group members when the group is set up and also new group key whenever there is a membership change (join/leave). Again the schemes which come under Key Distribution can be further classified into Asynchronous or Non-Interactive and Interactive Key Distribution schemes. In Non-Interactive, members in the privileged group can derive a common group key on their own with the help of the information obtained by KDC. Of the schemes discussed so far, Blundo et al. conference keying protocol [4], Broadcast encryption zero level and one level schemes [7], and Keystone architecture scheme [21] are Non-interactive Key Distribution schemes.

In Interactive, users in the privileged group can communicate with each other to set up the group key. Blundo et al. one-way interactive scheme [4] comes under this classification, as t^{th} user is choosing the group key and distributing it securely to the rest of the $(t-1)$ members in the group.

In Contributory Key Agreement schemes, there is no KDC involved. Instead, each group member picks its own contribution (maybe some random number) to group key and interacts with other members to compute the group key. Schemes proposed by Diffie-Hellman [6, 17] and Burmester-Desmedt [5] come under this scheme.

4.2. Based on Role of KDC

The schemes under Key Distribution are based on a single entity called KDC. These schemes are further classified based on the role of KDC. In some schemes, KDC is *active only during initial stage* of the protocol, i.e., KDC just generates and distributes initial private pieces of information among group members and becomes inactive. Blundo et al. Non-Interactive, Blundo et al. one-way interactive [4] and Broadcast encryption zero level schemes [7] come under this category. Where as in Broadcast encryption one level scheme [7] and Keystone architecture [21], KDC is *active throughout* the protocol. During initial stage, KDC generates and distributes securely initial private pieces of information among the privileged user set. During later stage, it becomes active during each join/leave operation. In broadcast encryption one level scheme, KDC itself generates random strings for the message and broadcasts it to the privileged set, where as in keystone architecture, whenever there is a membership change (join/leave), KDC generates new group key and auxiliary keys and sends them to the group members in an optimized way.

4.3. Based on Security

Group key management schemes provide either *unconditional security* or *conditional security*. In unconditional security, even if an adversary has unlimited computational resources, he cannot defeat the system. In conditional security, depending on the amount of computational effort required, an adversary can defeat the system and the methods depend on the hardness of the problem. The following schemes provide unconditional security: Blundo et al. non-interactive and one way interactive schemes [4], Broadcast encryption zero level and one level schemes [7], keystone architecture[21]. Where as the schemes proposed by Diffie-Hellman[6] and Burmester-Desmedt[5] provide conditional security.

5. Storage, Computation and Communication costs

Table 1 depicts storage, communication and computation costs at each user during key set up activity for the above discussed Key management schemes. This is considered after initial information is distributed to all the n users and after knowing the privileged user set, i.e.,

participating t members. In the table, n denotes total number of users in the system, t denotes size of the privileged set, k denotes the threshold and L total number of functions.

Let the number of users in the system, n be 30, size of the privileged set t be 10 and threshold value k be 3. For Blundo et al. non-interactive, one-way interactive, Broadcast encryption zero level and one level schemes, we have considered the size of the element from $GF(q)$ as 64-bits and for Group Diffie-Hellman, Burmester-Desmedt and Keystone architecture schemes, the size of the element is considered as 1024-bits. Table 2 shows the storage and communication costs in bits and total computation cost in bit-operations for different protocols after considering one join and one leave operations. If one member joins the group, size of the privileged set will be $(t+1)$ i.e., 11 and if a member leaves, it is $(t-1)$ i.e., 9. First row of each protocol indicates the storage, communication and computation costs during initial key set up activity.

Each exponentiation and multiplication operation requires $Wlog_2W$ bit-operations, where W is the size of the element. It is 64 $log_2 64 = 384$ bit-operations for the protocols for which $W =$ 64 and 1024 $log_2 1024 = 10240$ bit-operations for the protocols with $W = 1024$.

Calculations Illustration:

Consider Blundo et al. non-interactive scheme. Storage at each user is $\begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix} = \begin{pmatrix} 11 \\ 8 \end{pmatrix}$ $= 165$ co-efficients from $GF(q)$. Size of each co-efficient is considered as 64-bits, hence storage at each user $= 165 * 64 = 10560$ bits. Each term contains maximum 10 variables (t) with maximum degree 3 (k). Therefore, maximum number of exponentiations $= 10 * 3 = 30$.

$$\text{Number of Multiplications} = t * \begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix} = 10 * 165 = 1650$$

$$\text{Number of Additions} = \begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix} = 165$$

Protocol	Storage Required	Communication Cost	Computation Cost
Blundo et al. Non-interactive	$\begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix}$	No Communication	$k * t$exponentiations, $t * \begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix}$ Multiplications, $\begin{pmatrix} k+t-2 \\ t-2 \end{pmatrix}$ additions
Blundo et al. one-way Interactive	$(t+k-1)$	$(t-1)$	$2*(k+t-2)$exponentiations, and Multiplications, $(k+t-2)$additions
Broadcast Encryption zero level scheme	$\sum_{i=0}^{k} \begin{pmatrix} n-1 \\ i \end{pmatrix}$	No communication	$\sum_{j=0}^{k} \begin{pmatrix} n-t \\ j \end{pmatrix}$ Ex-or operations
Broadcast Encryption one level scheme	$t * L$	L	L Ex-Or operations
Keystone Architecture	$log_2 t$	$log_2 t$	–
Group Diffie-Hellman	random number, r_i	$2^{t-1}+1$	$t*(2^{t-1}+1)$ Exponentiations
Burmester-Desmedt	$2*t+2$	$2*t$	$(t+1)$ exponentiations $(t-1)$ multiplications

Table 1. Table depicting storage, communication and computation costs at each user during key set up activity.

Protocol		Storage Required (in bits)	Communication Cost (in bits)	Computation Cost (in bit-operations)
Blundo et al. non-interactive (64-*bits*)	$t = 10$	10,560	–	6,55,680
	Join	14,080	–	9,56,032
	Leave	7680	–	4,32,768
Blundo et al. one-way Interactive (64-*bits*)	$t = 10$	768	576	17,600
	Join	832	640	19,200
	Leave	704	512	16,000
Broadcast Encryption zero level scheme (64-*bits*)	$t = 10$	2,61,760	–	86,864
	Join	2,61,760	–	74,240
	Leave	2,61,760	–	99,968
Broadcast Encryption one level scheme (64-*bits*)$(L = 3)$	$t = 10$	1920	192	192
	Join	2112	192	192
	Leave	1768	192	192
Keystone Architecture (1024-*bits*)	$t = 10$	5120	4096	–
	Join	5120	4096	–
	Leave	5120	4096	–
Group Diffie-Hellman (1024-*bits*)	$t = 10$	1024	52,53,120	52,53,120
	Join	1024	13,312	1,33,120
	Leave	1024	1024	92,160
Burmester-Desmedt (1024-*bits*)	$t = 10$	22,528	20,480	2,04,800
	Join	24,576	22,528	30,720
	Leave	20,480	18,432	30,720

Table 2. Table depicting storage, communication and computation costs.

Total computation cost for Exponentiation, Multiplication and Addition $= (30 + 1650) * 384 + 165 * 64 = 655680$ bit-operations.

For every join/leave operation, the protocol must be restarted (i.e., KDC must pick another symmetric polynomial and distribute it) from the initial stage. For join operation, every user will have $\begin{pmatrix} 12 \\ 9 \end{pmatrix} = 220$ coefficients and is required to perform $11 * 3$ exponentiation operations, 2420 multiplications and 220 additions. Total storage is $220 * 64 = 14080$ bits and computation cost is $(33 + 2420) * 384 + 220 * 64 = 956032$ bit-operations. If a user leaves the group, storage $= 120 * 64 = 7680$ bits and computation cost $= (27 + 1080) * 384 + 120 * 64 = 432768$ bit-operations.

For Diffie-Hellman protocol, each user must store only the random number whose size we have considered as 1024-bits. During initial group key set up, totally $t * (2^{t-1} + 1)$ messages are exchanged among privileged user set. Hence, communication cost $= 10 * (2^9 + 1) = 5130$ messages * 1024 bits $= 52,53,120$ bits. Each user must perform $2^{t-1} + 1$ exponentiations, which leads to $2^9 + 1 = 513$ exponentiations with 10240 bit-operations for each exponentiation. So, total exponentiation cost $= 513 * 10240 = 52,53,120$ bit-operations.

If a user joins the group, only $(t + 3)$ messages are sent and $(t+3)$ exponentiation operations are performed. Hence, total communication cost is $13,312$ bits and computation cost is $1,33,120$ bit-operations. For a leave operation, only one message is sent (1024bits) and $(t-1)$ exponentiation operations are performed i.e., computation cost $= 9 * 10240 = 92,160$ bit-operations.

5.1. Performance Analysis

In Blundo et al. non-interactive scheme and broadcast encryption zero level scheme, amount

of storage required at each user and the computation costs are larger, but there is no communication among the group members. In Blundo et al. one-way interactive scheme, amount of storage required is very much less, compared to non-interactive scheme and it requires moderate computation cost with minimum communication cost. Broadcast encryption one level scheme requires moderate amount of storage with less communication and computation costs. In keystone architecture, no computation cost is involved since KDC itself sends the group key. In Group Diffie-Hellman, storage required is negligible, but requires more messages to be exchanged with high computation cost. Burmester-Desmedt scheme requires more storage; communication and computation costs are also high to set up the group key, but, computation cost required is less for join/leave operations as compared to Diffie-Hellman, Broadcast encryption zero level and Blundo et al. non-interactive schemes. Graphs depicting number of users versus storage, communication and computation costs are plotted. Figure 7 gives the comparison for amount of storage at each user and Figure 8 about communication and computation costs. From the graphs we can see that amount of storage required for Diffie-Hellman is negligible, but communication and computation costs are too high.

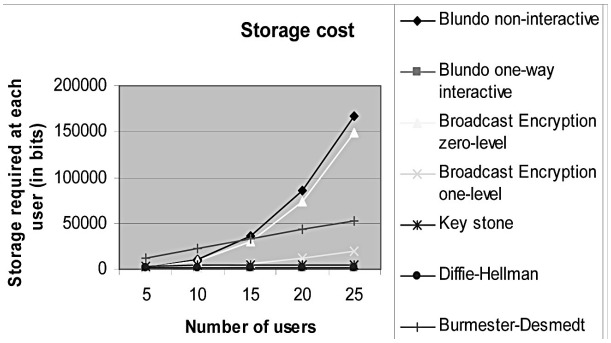

Figure 7. Storage at each user.

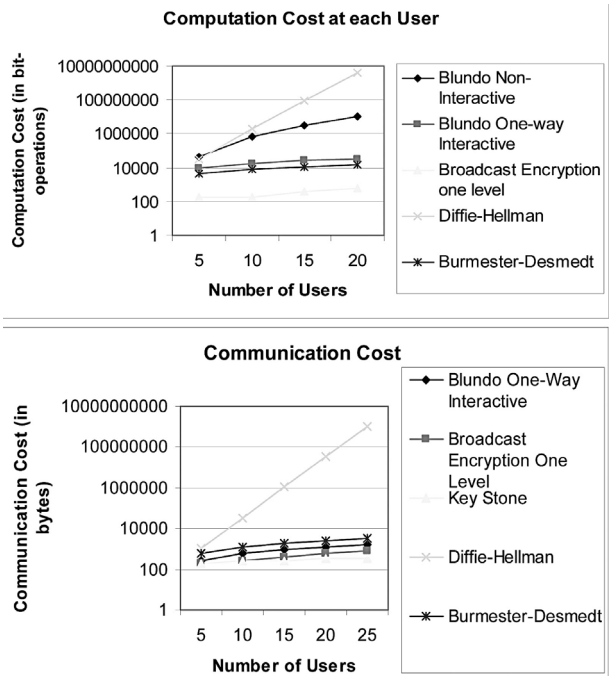

Figure 8. Communication and computation costs.

For join/leave operations, the schemes proposed by Blundo et al. and broadcast encryption zero level schemes require almost the same amount of computation as required for initial key set up, where as it is very much less in case of Diffie-Hellman and Burmester-Desmedt protocols. Among the protocols discussed in the paper, broadcast encryption one level and keystone architecture schemes are quite efficient as compared with other schemes.

6. Conclusion

We discussed several key management schemes and classified them based on the method used to establish the group key, involvement of Key Distribution Center in different phases of secure group communication and security factors. We analyzed the performance of the protocols based on the amount of storage required, communication and computation costs during initial key set up activity and during join/leave operations. It is evident from the tables that the schemes which require more storage, require less amount of communication. But, at the same time, computation cost required for join/leave operations is quite high. The schemes which require more communication among users, require less amount of computation for each join/leave operation.

References

[1] Y. AMIR, Y. KIM, C. NITA-ROTARU, J. SCHULTZ, J. STANTON, G. TSUDIK, Secure Group Communication using Robust Contributory Key Agreement. *IEEE Transactions on Parallel and Distributed Systems*, 15, 5, 2004, pp. 468–480.

[2] Y. AMIR, C. DANILOV, M. MISKIN-AMIR, J. SCHULTZ, J. STANTON, The Spread Toolkit: Architecture and Performance. Technical Report, CNDS-2004-1, Johns Hopkins University, 2004.

[3] D. BALENSON, D. MCGREW, A. SHERMAN, Key Management for Large Dynamic Groups: One-way Function Trees and Amortized Initialization. Internet Draft, draft-irtf-smug-groupkeymgmt-oft-00.txt, August 2000.

[4] C. BLUNDO, A. DE SANTIS, A. HERZBERG, S. KUTTEN, U. VACCARO, M. YUNG, Perfectly Secure Key Distribution for Dynamic Conferences. In *Advances in Cryptology-CRYPTO'92*, LNCS, 740 (1993), pp. 471–486.

[5] M. BURMESTER, Y. DESMEDT, A Secure and Efficient Conference Key Distribution System. *Advances in Cryptology – EUROCRYPT'94*.

[6] W. DIFFIE, M. HELLMAN, New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6): 644–654, November 1976.

[7] A. FIAT, M. NAOR, Broadcast Encryption. In D. R. Stinson, editor, *Proceedings of CRYPTO'93*, pp. 480–491.

[8] O. GOLDREICH, S. GOLDWASSER, S. MICALI, How to Construct Random Functions. *Journal of the ACM 33*, 1986, pp. 792–807.

[9] P. JUDGE, AMMAR, Security issues and solutions in multicast content distribution: A Survey. *IEEE Network*, Jan/Feb 2003.

[10] Y. KIM, A. PERRIG, G. TSUDIK, Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups. In the 7^{th} *ACM Conference on Computer and Communications Security*. ACM Press, 2000, pp. 235–244.

[11] P. P. C. LEE, J. C. S. LUI, D. K. Y. YAU, Distributed Collaborative Key Agreement Protocols for Dynamic Peer Groups. *Proc. IEEE International Conference on Network Protocols (ICNP)*, November 2002.

[12] K. MEHLHORN, *Data Structures and Algorithms: Sorting and Searching*. Springer-Verlag, Berlin Heidelberg, 1984.

[13] S. MITTRA, Iolus: A Framework for Scalable Secure Multicasting. In *Proceedings of the ACM SIGCOMM*. Vol. 27, No. 4 (NewYork, Sept.), ACM, NewYork, 1997, pp. 277–288.

[14] A. PERRIG, Efficient Collaborative Key Management Protocol for Secure Autonomous Group Communication. *Proc. of International Workshop CrypTEC*, 1999.

[15] S. RAFAELI, D. HUTCHISON, A survey of Key Management for Secure Group Communciation. *ACM Computing Surveys*, September 2003.

[16] A. T. SHERMAN, D. A. MCGREW, Key Establishment in Large Dynamic Groups Using One-way Function Trees. *IEEE Transactions on Software Engg.* 2003, pp. 444–458.

[17] M. STEINER, G. TSUDIK, M. WAIDNER, Diffie-Hellman Key Distribution Extended to Group Communication. In *SIGSAC Proceedings of the third ACM conference on Computer and Communications Security*. New Delhi, India, March 1996., ACM, New York, pp. 31–37.

[18] M. WALDVOGEL, G. CARONNI, D. SUN, N. WEILER, B. PLATTNER, The Versakey Frameworks: Versatile Group Key Management. *IEEE Journal on Selected Areas in Communication (JSAC)*, Vol. 17, No. 9, September 1999, pp. 1614–1631.

[19] D. WALLNER, E. HARDER, R. AGEE, Key Management for Multicast: Issues and Architectures. *Request For Comments (Informational) 2627, Internet Engineering Task Force*, June 1999.

[20] C. WONG, M. GOUDA, S. LAM, Secure Group Communication Using Key Graphs. In *Proceedings of the ACM SIGCOMM'98*, October 1998.

[21] C. K. WONG., SIMON S. LAM, Keystone: A Group Key Management Service. In *Proceedings of International Conference on Telecommunications*, Acapulco, Mexico, May 2000.

Received: February, 2008
Revised: November, 2008
Accepted: December, 2008

Contact addresses:
R. Aparna
Dept. of Computer Science and Engg.
Siddaganga Institute of Technology
Tumkur, Karnataka, India
e-mail: raparna@sit.ac.in

B. B. Amberker
Dept. of Computer Science and Engg.
National Institute of Technology
Warangal, Andhra Pradesh, India e-mail: bba@nit.ac.in

R. APARNA obtained her M.S. from Birla Institute of Technology, Pilani, Rajasthan, India. She is presently working as an Assistant Professor in the Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumkur, Karnataka, India and pursuing Ph.D in the area of cryptography and network security.

B. B. AMBERKER obtained his Ph.D from the Department of Computer Science and Automation, IISc., Bangalore, India. He is presently working as a Professor in the Department of Computer Science and Engineering, National Institute of Technology, Warangal, AP, India.