

2123dT
2003
p121

**GROUP KEY MANAGEMENT UTILIZING
HUFFMAN AND PETRICK BASED
APPROACH**

By

SENTHAMIL ILANGO

Bachelor of Engineering

University Of Madras

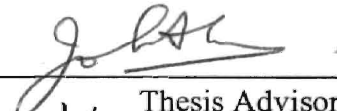
Madras, India

1999

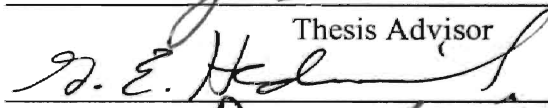
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in the partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
August, 2003

**GROUP KEY MANAGEMENT UTILIZING
HUFFMAN AND PETRICK BASED
APPROACH**

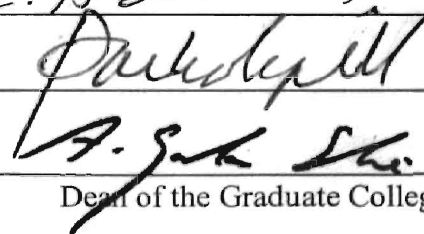
Thesis Approved:



Thesis Advisor



G. E. Hedman



Dean of the Graduate College

PREFACE

Fields of application of Networks, these days, demand high deal of security. From E-Banking to highly classified data being on the Network, any unauthorized data will be a huge threat. Even though these Networks needs to be highly secured the applications does not like compromising the speed for that. These Networks uses a popular scheme of encryption for data security. The scope of this thesis is to minimize the data traffic in the Network because of these encryption key distribution and Re-distribution. Minimizing these traffic increases the better utilization of Network bandwidth. This minimization is done without compromising the security.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Dr. Johnson Thomas for his assistance and able guidance through out my thesis in Oklahoma State University. I would also like to thank my committee members, Dr. G.E. Hedrick and Dr. Nophil Park, for their timely advice and help.

I would like to thank my parents for their ever-lasting encouragement and emotional support throughout the years.

Finally I would like to thank all my friends for their great support throughout my life.

TABLE OF CONTENTS

	Page
Chapter 1 Introduction	
1.1 Network.....	1
1.2 Network Security.....	2
1.3 Security in Multicast Networks.....	3
Chapter 2 Key Management: Multicast Networks	
2.1 Overview of Multicast Networks.....	4
2.2 Interloping and Prevention in Multicast Networks.....	6
2.3 Key factors in Key Management.....	8
Chapter 3 Operating mechanisms of various Key Management	
3.1 Key Graphs for Key Management.....	9
3.2 Definitions in Key graphs.....	11
3.3 Boolean logic Minimization tech.....	15
3.3.1 Key Management after the user leaves the group.....	16
3.3.2 Special Procedure for Multiple leaves.....	17
Chapter 4 Problems With Multicast Network Security	
4.1 User Leaving the group.....	20
4.2 User Entering the group.....	21
Chapter 5 Proposed Solution	
5.1 Basic Technique.....	22
5.2 Modified Huffman Scheme.....	22
5.2.1 Steps involved in generation of UID.....	24
5.3 Generation of UID.....	25
5.4 User leaving the Network.....	25
5.4.1 Changing the group key.....	27

5.4.2	Removal of group due to expiry of subscription.....	28
5.4.3	Algorithm.....	28
5.5	Petrick's method of Boolean logic minimization.....	28
5.5.1	Introduction.....	29
5.5.2	Steps involved in minimization.....	29
5.6	Reordering of keys.....	34

Chapter 6 Simulation and Results

6.1	Highly stable Network (Predictable user leave).....	36
6.2	User leave predictable with good probability.....	36
6.3	Highly unpredictable user leaves.....	37
6.4	Charts.....	48
6.5	Results.....	40

Chapter 7 Conclusions

7.1	Conclusions.....	41
7.2	Future Work.....	41
	References.....	42
	Appendix.....	45
	1. source code.....	45

List of Tables

Table		Page
3.1	Truth table for Boolean logic Minimization.....	13
5.1	UserID vs. probability table.....	23

List of Figures

Figure		Page
3.1	Key Graphs.....	13
5.1	Huffman Code Construction.....	23
5.2	Modified Huffman codes for Key Management	26
6.1	Performance comparison for a highly stable Network.....	38
6.2	Performance comparison for a predictable Network.....	39
6.3	Performance comparison for a non-predictable Network.	40

List Of Symbols

CN	-	Computer Networks
MHT	-	Modified Huffman Technique
UID	-	UserIDs.
GSA	-	Group Security Agents
GSI	-	Group Security Intermediaries
SOP	-	Sum Of Products
POS	-	Product of Sum
CAA	-	Centralized Authentication Agent

Chapter 1

Introduction

1.1 Networking

The computer industry had a mammoth growth over the past two decades. The wide usage of computers in various fields and the increasing user friendliness led to its growth. Along with computers, another field, which had a notable growth, is Communication. The high speed Internets and wireless communication has become an integral part of our daily life.

Data sharing is the key idea, which is driving this industry's growth. By keeping the widely used data in a common place, so that any number of users can access it, the utilization of the resources is increased. This key idea of sharing has an obvious threat of unwanted persons accessing the information. This basic problem is the reason for the increasing need for Security in Networks. One of the major security challenges for Networks is authorizing the users who are accessing the Network and preventing unauthorized users from "hacking" the Network.

1.2 Network Security

The areas in which Networks have a wide range of applications require a great deal of security. For example, one of the major applications of Networks is in Internet

Banking. Authorization of users and preventing a third party from having illegal access are a huge challenge in this field. This is also true of secret databases like military and Government information on the Internet. Some one able to break this security could be a potential threat.

In addition to preventing unauthorized persons from accessing the Network there is also a need to provide different access levels for different users. One example for this kind of scenario is the Cable Television Network where viewers access to different channels are determined by the subscription fee paid by them. Even though all are in the same Network, different users have different privilege levels, which is a challenge to the centralized controller. Also the user's authorization expires after certain period of time. This is an ideal example for a secure multicast Networks, where the users and their access rights are valid for a certain period of time and their validity expires after some duration, which is determined by a centralized unbiased agent.

The way in which data is being transmitted in these Secure Networks are through various encryption schemes. The data is not transmitted as such. The transmitter and the receiver in the network share some common key (or compliments). The transmitter computes the encrypting function, Key (transmitted data) before transmission, where the receiver just has to De-key (transmitted data). This allows the intended receiver to receive this data and also prevents the unwanted users from accessing this data because they will not have the De Key to read these data

Security in Multicast Networks

In Multi-cast Networks the Users are organized as a group. In order to send the messages to the group, the user encrypts the message with a Group Key, the Key shared by all the users in the group. All the members of the group can now decrypt this message to its original form using the group key. This encryption makes sure that all the users in the group can access this message. No one else outside the group can have access because they do not have the group key. The users entry/exit are major challenges in these Networks.

In this thesis we are designing a scheme that can be used of managing the key distribution in the Multi-cast Network. We are proposing an algorithm which can be used for constructing the userIDs which in turn manages the keys possessed by each users. We are also devising a technique for key management after the removal of users from the Network.

The organization of this thesis is as follows. In Chapter 2, background information on Multicast Network Security is provided. Chapter 3 describes the operating mechanisms of various Security techniques in these kinds of Networks. The typical problems in Network Security are discussed in Chapter 4. In Chapter 5, the description of the proposed approaches is given. Chapter 6 evaluates the performance of the proposed approaches, and in the last Chapter the thesis is concluded with pointers for future research.

Chapter 2

Key Management: Multicast Networks

2.1 Overview of Multicast Networks

Multicast is the term used to describe communication where a piece of information is sent from one or more points to a set of other points. In this case there may be one or more senders, and the information is distributed to a set of receivers (there may be no receivers or any other number of receivers). One example of an application, which may use multicast, is a video server sending out networked TV channels. Simultaneous delivery of high quality video to each of a large number of delivery platforms will exhaust the capability of even a high bandwidth network with a powerful video clip server. This poses a major scalability issue for applications, which required sustained high bandwidth. One way to significantly ease scaling to larger groups of clients is to employ multicast networking.

Unlike broadcast transmission (which is used on some local area networks), multicast clients receive a stream of packets only if they have previously elect to do so (by joining the specific multicast group address). Membership of a group is dynamic and controlled by the receivers (in turn informed by the local client applications). The routers in a multicast network learn which sub-networks have active clients for each multicast group and attempt to minimize the transmission of packets across parts of the network for

which there are no active clients. The multicast mode is useful if a group of clients require a common set of data at the same time, or when the clients are able to receive and store (cache) common data until needed. Where there is a common need for the same data required by a group of clients, multicast transmission may provide significant bandwidth savings (up to $1/N$ of the bandwidth compared to N separate unicast clients).

The Majority of installed LANs are able to support the multicast transmission mode. Shared LANs (using hubs/repeaters) inherently support multicast, since all packets reach all connected to the LAN. The earliest LAN network interface cards had no specific support for multicast and introduced a big performance penalty by forcing the adaptor to receive all packets (promiscuous mode) and perform software filtering to remove all unwanted packets. Most modern network interface cards implement a set of multicast filters, relieving the host of the burden of performing excessive software filtering.

In Multi-cast Networks [3] the Users are organized as a group. Each user holds a group key and an individual key. An un-biased centralized authentication agent manages all these users. The individual key is for the communication between any user and the centralized authentication agent and the group key is for the communication between the members in the group. The centralized agent manages the key possessed by each user. Suppose if station 'A' in the network wants to transmit data to the users in the Network then ,

Station A \rightarrow Encrypts Message with Group Key \rightarrow Transmit Encrypted data

Station $x \rightarrow$ Sees the data in Network \rightarrow Picks the data \rightarrow Decrypts the data with Group key

Where x belongs to the Multicast Network.

Any other station outside the Network will not be able to get this data because they do not have the key to decrypt it.

2.2 Interloping and Prevention in Secured Multicast Networks

Any station outside the Network trying to access the data in the Network is called as

Interloping. This is a potential threat for the multicast Networks. There are lot of hacking mechanisms these interlopers use to access the data in a secure Network. This is beyond the scope of this document.

In most of the cases the Interlopers will not be an outsider. The station, which was a part of the Network in the past and had left, could be a potential interloper. Consider a station which was the part of a Network leaves because of the expiry of subscription. If the centralized authentication agent did not change the group key then the user, who left the Network can interlope the data. Consider the following scenario,

1. User X enters the Network.
2. Authorized by the centralized authentication agent.
3. Gets the group key and the individual key.
4. The validity period of the user X expires.

5. The user X leaves the Network.

Now neither the remaining users nor the Centralized Authentication Agents want the User X to access the data. But if the stations in the network keep sending data encrypted by the group key then User 'X' will have access to the data. Because of this reason, the group key is always changed whenever a user leaves the Network. We will revisit the same scenario again.

1. User X leaves the Network.
2. The Centralized authentication agent creates a new group key.
3. The Centralized authentication agent transmits the new group key to each user in the Network.
4. For this transmission the group key has to be encrypted with the individual key of each of the users other than the one who left the Network, i.e. station 'X'.
5. Further transmission in these Networks will be encrypted with this new Group key, which user 'X' does not have.

By means of changing the group key as mentioned above after every user leaves, interloping by the previous members of the group can be prevented.

Also when users enter the multicast group, they need to be given access. In other words, they have to be provided with a individual key and the group key. For transmitting the individual key the Centralized agent uses a one to one encryption mechanism. After

transmitting the individual key, the Group key can now be transmitted by using the individual key. This group key has to be encrypted with the individual key to prevent interlopers from gaining access.

2.3 Key factors in the Key Management:

In a huge network or multicast group with a large number of users, this key management will become cumbersome. The key management scheme, which the centralized agent will use, should be efficient otherwise it will degrade the performance of the Network.

The Key Management scheme designed should meet the following basic criteria.

1. At any instant, the keys possessed by any of the users in the multicast group should be identified quickly.
2. At any given moment the centralized authentication agent should be able find out all the users possessing a particular individual key.
3. if a user leaves the network, there should be a better scheme than sending n-1 unicast message for the re-distribution of keys.
4. The time taken for the encryption of the group keys for the re-distribution should be minimum but without compromising the security.

Also even if there is an interloping the loss or the risk of that should be kept to as low as possible.

Chapter 3

Operating mechanisms of various Key Management

There are two major ways of categorizing Key management techniques. One is using graphs and the other is using the Boolean logic scheme. These security mechanisms are now described.

3.1 Key Graphs for Group communication

One of the ways of managing keys in Group communication is through Key Graphs [3]. In this approach they have dealt with the problem of the number of encryptions required for re keying after the removal of user(s) from the group.

if all the users are categorized under one large group, the problem of dealing with user(s) leaving the group becomes difficult. After a user leaves the group, the group key has to be changed. The Centralized agent after generating the new group key has to send $n-1$ different messages, encrypted with individual key of every user other than the one who is leaving the group. As the number of users in the group increases, this problem becomes even worse. The solution to this problem is to divide the group in to subgroups.

In this scheme the group of users is divided into sub-groups. Each user now has a Group Key, Individual Key and Subgroup Key. The group key is common to all the users

in the group and used for encrypting and decrypting the group communications. The individual key is to enable the Secret One to One communication between the Centralized agent and the User. The Subgroup key[3] is common to all the members in subgroup, which is used for reducing the overhead of re keying after the removal of a user from the group.

Let there be a trusted centralized authentication agent responsible for group access control and key Management. In particular, the agent securely distributes keys to group members and maintains the user-key relation[3]. This approach can be explained with a simple example.

Assume a secure group with nine members. These nine members are organized as three subgroups. Each member is given three keys: its individual key, a key for the entire group and a key for its subgroup. If a user leaves the group, the remaining eight members form a new secure group and require a new group key. Since they form a new subgroup they require a new subgroup key. The new subgroup has three, three and two members respectively.

For sending the new subgroup key to each user, except the one that leaves, the agent encrypts the new subgroup key with the individual key of each user.

Initially there were three users in each subgroup. Now one of the users has left, so number of encrypted messages is 2. This encryption is only for the group that lost a member.

For sending the new group key, the message has to be encrypted with just the subgroup keys (not individual key of each members). Since there are 3 subgroups there will be 3 encrypted messages.

Totally $2+3=5$ messages.

If there were no subgroups we need 8 encrypted messages (one encrypted message for each of the user in the group other than the one who left). So this scheme reduces the number of encrypted messages after the user removal.

3.2 Definition of Key graphs

A key graph is a directed acyclic graph. This graph has two kinds of nodes.

- Nodes representing users (u)
- Nodes representing keys (k)

Each 'u' node has one or more outgoing edges but no incoming edge. Each 'k' node has one or more incoming edge(s) and may or may not have out going edges(s). The node with no outgoing edge but one or more incoming edges(s) is the root node.

The Key graph [3] is a graphical representation of a secure group. The secure group is denoted by a triple (U, K, R) , where U represents the set of users, K represents the set of Keys and R represents a binary relation between U and K , $R \subset U \times K$ called the user key relation. The user u has a key k only if (u, k) is in R .

A Key Graph G represents the secure group (U, K, R) as follows.

- For every user u in U of secure group there is a node u_i , in the key graph
- For every key k in K of secure group there is a node k_i in the key graph
- If (u, k) is in R then the graph G has a directed path from node u_i to node k_i .

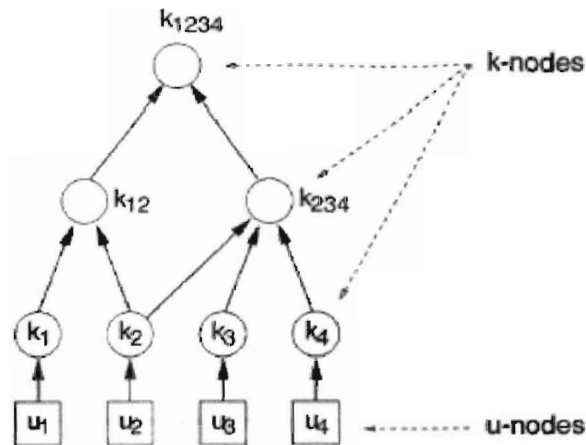
Consider the following secure group.

$$U = \{u_1, u_2, u_3, u_4\}$$

$$K = \{k_1, k_2, k_3, k_4, k_{12}, k_{234}, k_{1234}\}$$

$$R = \{ (u_1, k_1), (u_1, k_{12}), (u_1, k_{1234}), \\ (u_2, k_2), (u_2, k_{12}), (u_2, k_{234}), (u_2, k_{1234}), \\ (u_3, k_3), (u_3, k_{234}), (u_3, k_{1234}), \\ (u_4, k_4), (u_4, k_{234}), (u_4, k_{1234}) \}.$$

This can be represented as



There are two functions associated with the secure group. They are key set and the user set.

Key set (u) represents the set of keys held by the user u .

User set (k) represents the set of users that has the key k .

For example

Key set (u_1) \rightarrow $\{ k_1, k_{12}, k_{1234} \}$

User set (k_{12}) \rightarrow $\{ u_1, u_2 \}$

When a user leaves the group, the keys that the leaving user shares with other members in the group has to be changed. If 'k' is one of those keys then for replacing the key k , the centralized agent generates the key k_{new} and it transmits to every

user in User set (k) except the user who is leaving. To transmit securely the server needs to find the minimum key set k' such that $\text{user set}(k') = \text{user set}(k) - \{u\}$ and these k' keys can be used for encryption.

The calculation of this minimum set of keys k' is called the key-covering problem. In order to solve the issue, this approach suggests special kind of graphs such as star and key graphs.

Star: This is the special class of a secure group where each user has only two keys: its individual key and a *group key* that is shared by every user in-group.

Tree: Each of the non-leaf nodes represents a key. The root represent the *group key* and the every descendent other than the immediate predecessor of the leaf node represents a subgroup key.

This approach has an advantage of decreasing the number of encryptions and also decreasing the number of messages. The draw back of this approach seems to be overhead of key set and user set manipulations.

3.3 Boolean function minimization technique

One of the efficient strategies of this kind of Key-Management is through Boolean function minimization technique. In this approach each user is given a set of keys called auxiliary keys in addition to the Session's key they possess. These keys are similar to the idea of subdividing the users into sub-groups. So the overhead increases to the central authentication agent because it has to manage all these Auxiliary keys in addition to the Sessions key. What is an auxiliary key? Session key is like a group key and auxiliary key is like a sub-group key?? The way in which the Centralized agent does this, is by means of giving each of the users in the group a unique ID called UserID. The UserID is a binary string of length n . The value of this 'n' is dependent on the number of users in the group. If we have n users in the group then,

The Length of the UserID string n is $\lceil \log_2 N \rceil$

That is, if we have 1000 users in the group then the number of bits in UserID of all the users in the group is 10.

The UserID represents the set of auxiliary keys possessed by the user that is, If the UserID is say 100 then the user in addition to the Sessions Key, has auxiliary Keys k_2 , k_1 and k_0 . The Centralized Agent creates a pair of keys for each bit of the UserID. Suppose if the length of the UserID string is 10 then the Centralized agent generates 10 pair of keys ,

that is $k_0, k_0', k_1, k_1' \dots k_9, k_9'$. Then with respect to the UID string these keys are distributed among the users. Suppose if one of the users has all the bits 0s in the UID then the keys distributed to that user are $k_0', k_1' \dots k_9'$. These k_i and k_i' are just a pair of keys and not the actual compliment of one another.

Key Management after the user leaves the group

Suppose if one of the users leaves the group either by itself or by being removed by the Centralized Authentication Agent due to the expiry of the Subscription period. Therefore the session key has to be changed. The reason for changing this session's key is to make sure that the user who leaves the group now, cannot have access to the group communication anymore. In addition to the change of the session's key the Auxiliary keys possessed by the user also has to be changed. If we don't change the auxiliary keys possessed by the user who is leaving now, then the user by listening to messages encrypted with the keys that he is possessing, can Interlope into the group.

When a user leaves, in order to change the session's key, the Centralized Authentication Agent first generates a new session key. This key now has to be accessed by every one in the group except the user who is leaving now. So in order to satisfy these criteria the new session's key is encrypted with the keys that are compliments of the one

possessed by the user who is leaving now. That is if the user with UID all 10 bits 0 leaves, the new session's key is encrypted with key k_0, k_1, \dots, k_9 . This approach makes sure that every one else other than the one who is leaving can decrypt this key because every one in the group differs by every other user at least by one bit in UserIDs. So all the users in the group except the one who is leaving has at least one of bit as high otherwise the UID will not be unique. So every other user is having at least one key k_i instead of k_i' (depends of which bit in UID are high) . The user just needs one of these 10 keys for decrypting this message.

In order to change the auxiliary keys possessed by the user, the Centralized agent generates new set of keys for all the ones possessed by the leaving user. Then this key is encrypted with new session's key and the old auxiliary keys. Only the valid users in the group who has these old auxiliary keys can access these keys. The user who has left even though has the old auxiliary key cannot decrypt this because he don't have the new session's key any more.

Special Procedure for Multiple Leaves

Most techniques deal with the scenario of more than one user leaving at the same time by considering them as separate consecutive leaves. That is if 5 users leave at the same time then all these users are dealt as if they leave one after another. But

this Boolean function minimization technique deals this problem in a special way so that the number of encryption is reduced by a larger amount compared to other schemes.

In the case of Boolean function scheme this multiple leaves simplifies to a problem that all the users who are all leaving now should not access the new session's key but all other users (those not leaving) should be able to access the new session key. This problem is similar to a Boolean logic truth table where for each set of values the output is either a 1 and for the other set it is 0. Since the UID is binary the same idea of truth table can be applied of this multiple leaves.

if there are eight users in the group, then the UID is going to be composed of three bits. (000,001,...,111). Suppose if the user with UID 000 and 100 leaves at the same time this problem can be stated using a truth table where for all the UID values the output is a '1', except for the UID values of those leaving, in which case the output value is a '0'.

Truth Table:

000	0
001	1
010	1
011	1
100	0
101	1
110	1
111	1

The problem of dealing with multiple leaves is minimized to solving the truth table. There are several ways in which this table can be minimized. The schemes suggested by the Boolean function minimization approach are K-maps and QuineMcLuskey method.

Chapter 4

Problems With Multicast Network Security

4.1 User leaving the Group

In Multi-cast groups the users are valid for a certain period of time and after that their validity expires. A central authentication agent verifies the validity of the users. One of the ways of determining the validity is with respect to the subscription fee paid by the users to group. The user who pays more subscription fee is valid for longer period than one who pays lesser subscription fee. It is up to Central authentication agent to make sure that the user for whom the validity is expired cannot decrypt the messages of the group. This can be achieved by means of changing the group key after the removal of a user.

Changing the group key is not as easy as it is said. It requires lot of re keying. if a user leaves the group, then to change the group key, the Central authentication agents have to generate the new Group key first. This new group key has to be distributed to all the group members other than one who has left now. For distributing this new group key the central authentication agent has to send n separate encrypted messages, that is, the group key encrypted with each user's individual key (n is the number of users in the group after the leaves). By encrypting the new group key with each user's individual key, other than the one who left the group, the Central authentication agent can make sure that the user who left the group cannot get the new group key and hence he cannot have access to the group communication anymore. But ' n ' separate encryptions and

transmissions is expensive in high security networks where the number of bits in encryption is large.

This issue can be solved by the concept of Sub grouping described in the previous chapter. In this, the group as a whole is divided in to subgroups. If the user leaves now he only leaves the sub-group, which in turn reduces the number of re-keying when the user leaves the network.

Example:

If a user leaves the group,

For the subgroups, other than the subgroup of the leaving user,

We need Encryption and re transmission but the over head is less because the New Subgroup key can be encrypted with Old Subgroup keys.

For the subgroup of the leaving user,

n-1 unicast messages encrypted with individual keys of the user other than the leaving user.(n, number of users in subgroup)

4.2 User Entering the Group

When user(s) enter the Network, the centralized authentication agent has to first generate the individual key for that user. After distributing this individual key with any of the encrypting mechanism, then the agent transfers the Group key encrypted with the individual key.

Again this re-keying and redistribution is quite a cost in highly secured Networks.

Chapter 5

Proposed Solution

5.1 Basic Technique

IN this thesis we propose the technique that uses the Boolean logic minimization technique for Key Management. The advantage of using this scheme is, it reduces the key management overhead in the Multicast networks. In this scheme, the creation of UID for the group members by the controller is not done through the usual binary notation. The UID is created by the technique suggested by Huffman for communication theory. The scheme used for Boolean logic minimization is also going to be changed from QuineMcLuskey 's method to Petrick's method. The advantage of using the Petrick's method is , it has more systematic way of solving the Prime implicants and does not make more guesses like the Quine Mccluskey scheme.

5.2 Modified Huffman Technique

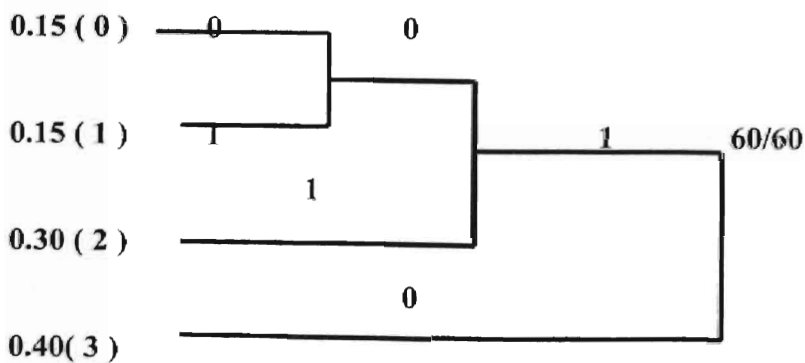
Huffman codes belong to a family of codes with a variable codeword length. That means that individual symbols, which make a message, are represented (encoded) with bit sequences that have distinct length. This characteristic of the code words makes data compression possible, that is, the data that is transmitted more frequently are represented by a lesser length code word than the data transmitted with a less frequency.

The Huffman technique requires probability of transmission for each user as input and o/p will be the variable length code in binary for each of the users in such a way that the user having the higher probability of transmission will have lesser length codes. The following example illustrates the way in which the Huffman code is generated from the probability of transmission. Suppose the probability of transmission of the following data are as given below

0	0.15
1	0.15
2	0.30
3	0.40

0 is transmitted with probability 0.15 and henceforth.

Then



Now the Representation of these data will be,

0	001
1	101

This method of UID generation makes sure that the user with expiry of subscription in near future has the least number of bits in its ID. So the number of keys that has to be changed when that user leaves the group is going to be less. However because of variable length coding there are going to be users whose UIDs have more bits than what they would have had if their UIDs were represented by normal binary notation. By reconstructing these UIDs again by this approach after some specific time period this can be resolved. This is because these users subscription expiration is now closer to the expiration deadline then previously.

5.3 Generation of UID

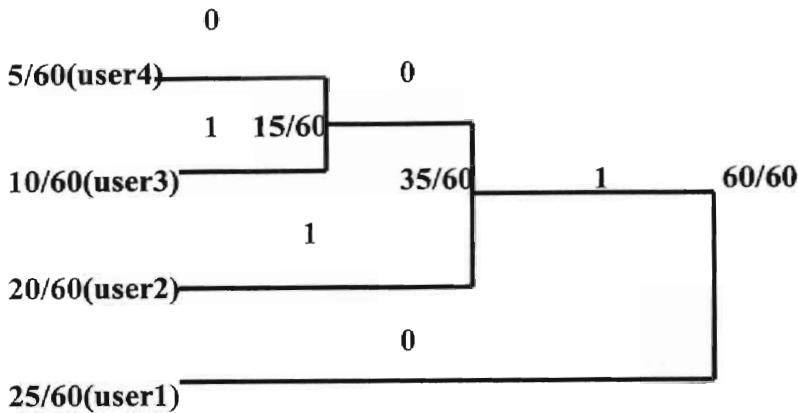
The generation of UID can be illustrated with an example. If the duration of stay of the users is calculated as 5,10,20,25 time units then the UserIDs can be generated as follows.

The time units are mapped to an input, equivalent to the probability of transmission. Each users validity period is divided by the sum of duration of stay of all the users. This gives the values $5/60, 10/60, 20/60, 25/60$. The inputs for generation of UID for each of the users are assigned in such a way that the user with least time units of validity has the highest value and henceforth.

User1	25/60
-------	-------

User2	20/60
User3	10/60
User4	5/60

Constructing Huffman codes,



This can be used converted to Huffman's codes,

User1	0
User2	11
User3	101
User4	100

So by having less number of bits for the user leaving most recently we can reduce the number of key to be changed after the leave. But the Keys with the new Session's Key to be encrypted is no longer with the complements of bits as suggested by the Boolean logic minimization technique.

5.4 User leaving the group

If the user leaves the group the following are required for preserving group security.

- Group key is changed
- The auxiliary keys possessed by the leaving user has to be changed
 - (Auxiliary keys are the additional keys that reduce the key redistribution overhead.

Changing group key

In the Boolean logic minimization technique changing the group key is easy because the centralized agent has to encrypt with the compliments of the UID of the leaving user. But in the modified Huffman approach its quite different.

Removal from the group due to expiry of subscription:

If the expected user leaves the group, that is the user leaves according to the expiry of the subscription, then the centralized agent has to encrypt the new Sessions key with k_{i+i} , k_{i+1} first, where 'i' is the most significant bit number of the user. In other words, if the leaving user has 3 bits UID then first step of encryption is with k_3 and k_3 . This encryption makes sure that every user with one or more bits more than the leaving user will get the new group key. Then for the users with same number of bits, the compliments of each of the bits of the leaving user starting from

the least significant bit is encrypted and it is repeated with next higher order bits until all the users of the same length UID gets the Group Key.

Algorithm:

1. Encrypt the new group key with k_{i+1} , k_{i+1} ('i' length of leaving user's UID)
2. if there are some users with same length UID in the group
 - j=0;
 - do
 - encrypt with j^{th} bit's compliment of leaving user's UID
 - j=j+1;
 - while(all the users of same length UID got the group key);
 - else
 - Stop the encryption.

5.5 Petrick's method of Boolean logic minimization:

If the user leaves are unpredictable then the Modified Huffman Technique cannot be used for generation of UID. But the minimization of Boolean function can be made efficient using the Petrick's method to reduce the Prime implicant chart.

The central authentication agent first runs the Quine McCluskey algorithm to construct the Prime Implicant Chart[10]. But in Petrick's method, the chart is not simplified using the usual trial and error method. The method suggested by petrick for minimization of chart is executed in the centralized authentication agent to make the output efficient.

Introduction

Petrick's method is a technique of determining the all-minimum sum of products solutions from the Prime Implicant chart. As the number of variables in the prime implicant chart increases, the complexity in solving them increases. This is proportional to the time taken to redistribution of keys in the real time networks using these kind of mechanism. Petrick's method is a more systematic way of finding the minimum solution from the Prime Implicant chart, which is constructed via Quine McCluskey's algorithm.

Before applying Petrick's method of Boolean logic minimization to the Prime Implicant chart, the min terms they cover in the chart should be removed.

Steps involved in the Petrick's method

- Eliminating prime implicant rows and corresponding columns to reduce the prime implicant chart.
- The rows of the reduced prime implicant chart are labeled as P1, P2. etc

- Logic function P is formed which is true when all the columns in the prime implicant chart are covered.
- P is reduced to minimum SOP [10] by multiplying out and applying $X+XY=X$ Boolean logic theorem [10].
- To determine the min solution finds the terms with minimum number of variables. Each of these terms represents the solution with a minimum number of prime implicant.
- Choose the one with minimum literals.

Example:

Lets start with the Quine Mcluskey and show the derivation with respect to Petrick's method.

$$F = \sum m (0, 1, 2, 5, 6, 7)$$

Derivation of Prime implicants:

First step: Group with respect to number of 1's.

0	000	*	(Zero 1s)
<hr/>			
1	001	*	(One 1s)
2	010	*	
<hr/>			
5	101	*	(Two 1s)
6	110	*	

7 111 * (Three 1s)

Second Step: Order the element that differs in one bit. And mark the previous grouping whichever gets entered here.

0, 1 00_

0, 2 0_0

1, 5 _01

2, 6 _10

5, 7 1_1

6, 7 11_

Step 3: No more grouping possible. The adjacent group does not differ by single bit any more. This is the point we construct a Prime Implicants Chart. Take all the unmarked items and construct the table

	0	1	2	5	6	7
(0, 1) $a'b'$	*	*				
(0, 2) $a'c'$	*		*			

(1, 5) c'd	*		*			
(2, 6) cd'		*			*	
(5, 7) ac			*			*
(6, 7) ab				*		*

Step 3: Cross out columns using the following scheme.

1. IF there is a column with only one marking choose that. Cross that row and column and select that term

2. If there are no columns with only one * then **guess** is made and a column is chosen arbitrarily. These is where the Petrick's work out to be a better technique.

Since here there are no columns with one marking we are starting with 1st column.

	0	1	2	5	6	7
(0, 1) a'b'		*	*			
(0, 2) a'c'		*		*		
(1, 5) c'd				*		
(2, 6) cd'				*	*	
(5, 7) ac				*		*
(6, 7) ab					*	*

The solution turns out to be

$$F = a'b' + bc' + ac$$

If we make other guess by choosing the second column first it turns out to be the solution is $F = a'c' + b'c + abc$

With respect to key management,

If we would have chosen the second solution it's more Re-Keying.

Let see Petrick's approach,

Step 1: Label the rows of the table as P1,P2..P6.

P1 is a logical variable which will be true if this row is selected.

Construct the equation for each row,

First row (P1 + P2) (since first and second columns are checked)

Second row (P1+P3)

Third row (P2+P4)

Fourth row (P3+P5)

Fifth row (P4+p6)

Sixth row (P5+P6)

All of these should contribute to the solution which implies

$$(P1+P2) (P1+P3) (P2+P4) (P3+P5) (P4+P6) (P5+P6) = 1$$

Reduce this equation applying the law $(X+Y) (X+Z) = X + YZ$

$$\text{And } (X+XY) = X + YZ$$

$$P = P1P4P5 + P1P2P5P6 + P2P3P4P5 + P1P3P4P6 + P2P3P6$$

This equation implies there are five different results. It turns out that minimum is the P1P4P5 which is **(Minimum implies less keys for us)**

$$a'b'+bc'+ac$$

The solution we see in this equation will come with different guesses from the Quine Mcluskey. Since we can analyze all the results systematically here, we can find out the best solution rather than going by guess.

5.5 Re-ordering the Keys

If the Keys are distributed using the modified technique the performance of the Network will be good until half the users leave the network. This is based on a reasonable assumption that for at least half the number of total users in the network the UserIDs generated by the Modified Huffman Technique will be smaller than the normal binary notation. If the number of users in the network goes below the half of the total since we computed the UserIDs, then it's a good period to start the Re-construction of Keys and transmit them. Even though this reconstruction and redistribution sounds like a overhead, they are not. The previous keys possessed by the users can achieve the redistribution of keys and it can happen when the Network is idle or traffic is low .The re-computation of new keys can be done by the centralized authentication agent in parallel while managing the keys for the existing

Network. The time to start the re-computation is also critical because this might impact the performance of the Network.

After re-ordering the keys, the previous keys possessed by the users in the Network can be kept in the buffer by the Centralized authentication agent in-order-to prevent the repetition which could be a potential threat for interloping.

This reordering of keys will not apply for the network which use the Petrick's method of Boolean logic minimization instead of the MHT because of highly unpredictable user leaves.

Chapter 6

Simulation and Results

To investigate the performance of the Modified Huffman technique over normal Boolean logic minimization, the number of keys until the point of restructuring of the network is plotted. The network has to be restructured when the performance of Huffman technique goes below the Boolean logic minimization technique. The way in which this is simulated is through constructing the Huffman code and binary UserIDs for all the users in the network and simulating the user leaving the network.

6.1 Highly stable Network where the user leave is predictable

The steps for simulation.

- Generate the binary notation user Ids for all the users.
- Generate the User Ids using the modified Huffman technique.
- The user leave is predictable in this Network. Always remove the predicted user from the Network.
- Compute the number of re-keying necessary for both the schemes
- The algorithm proposed in the scheme was used for computing the number of re-keying for the Modified Huffman Scheme.
- Plot a graph between these two data.

The above steps are repeated for different interleaving.

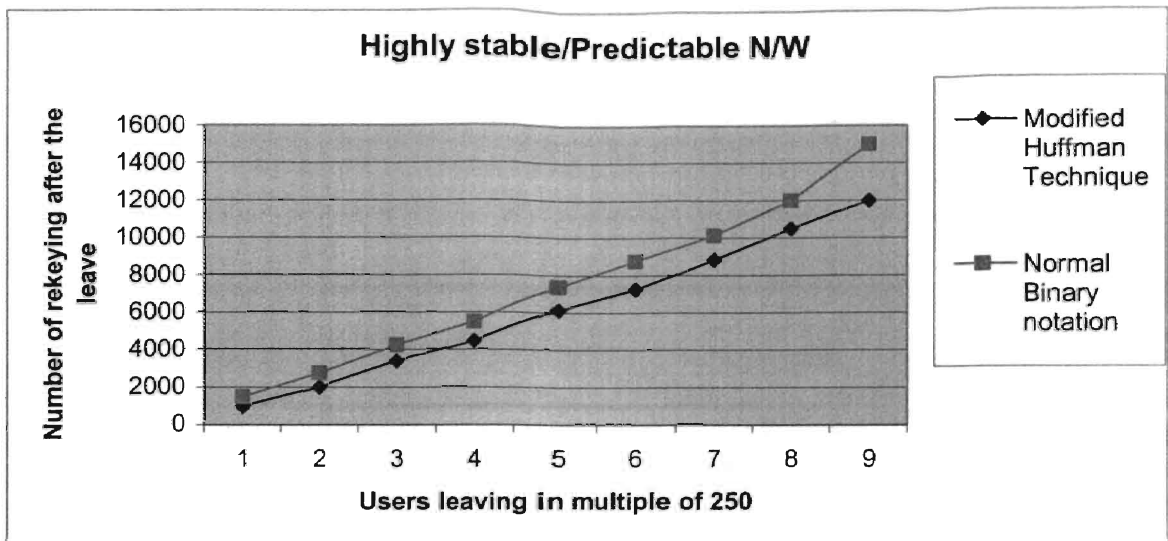
6.2 The Network where the user leave is predictable with a good probability

- Generate the binary notation user Ids for all the users.
- Generate the User Ids using the modified Huffman technique.
- Generate a random number to decide who leaves next. The high probability in the generating engine should be the predicted user.
- Compute the number of re-keying necessary for both the schemes
- The algorithm proposed in the scheme was used for computing the number of re-keying for the Modified Huffman Scheme.
- Plot a graph between these two data.

6.3 Network where the user leave is highly unpredictable

- Generate the binary notation user Ids for all the users.
- Generate the User Ids using the modified Huffman technique.
- Generate a random number to decide who leaves next. The high probability in the generating engine should be the user not predicted.
- Compute the number of re-keying necessary for both the schemes
- The algorithm proposed in the scheme was used for computing the number of re-keying for the Modified Huffman Scheme.
- Plot a graph between these two data.

6.4 Charts

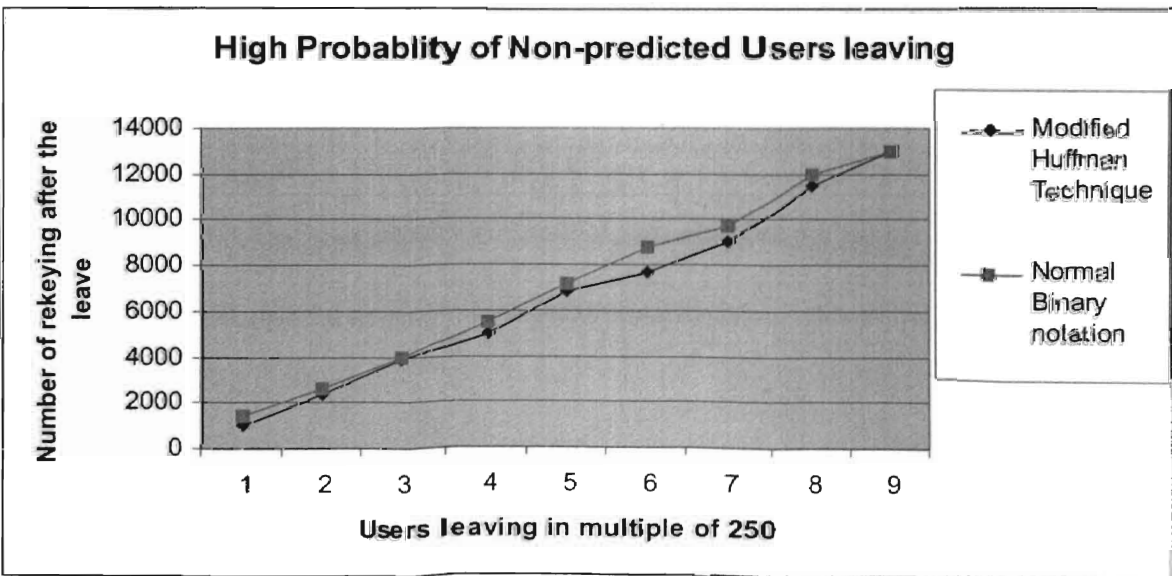
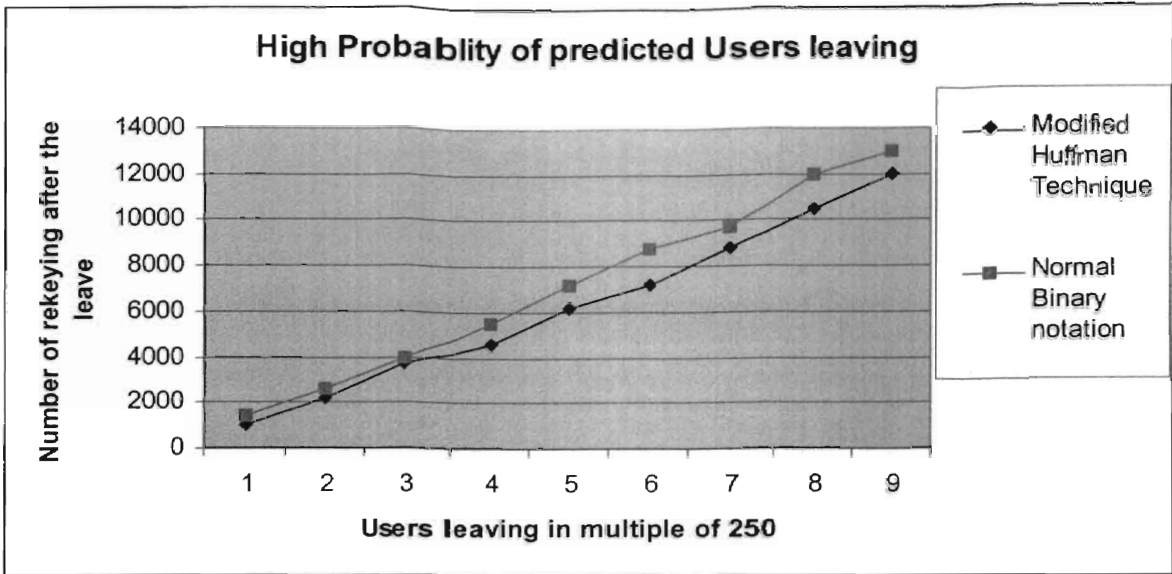


To be more specific,

X axis: Number of users leaving the network in the multiple of 250

Y-axis: Number of messages that has to be retransmitted.

On all these graphs



6.5 Results

Using the simulation the following observations were made.

1. The number of messages sent after the user leaves is reduced.
2. The number of keys that has to be changed is reduced.
3. The unwanted Network traffic is reduced
4. The better our prediction of when users will leave the network, the better the performance of the Network.

Chapter 7

Conclusions

7.1 Conclusions

- Number of encryptions required for re-keying after the removal of a user is decreased.
- Number of messages that has to be transmitted for the distribution of new keys might decrease.
- By using Petrick's method, the time required for finding the minimized expression will decrease.
- Since the numbers of key changes are going to be less (because of short UID) after a leave, the key management overhead is decreased.

7.2 Future Work

1. A better technique can be utilized for generating the UserIDs. Artificial Intelligence based approach can be utilized.
2. A self-learning algorithm can be devised for predicting the user leaving the Network.
3. The above approach can be implemented in Network Simulator and the results can be studied.

References

References:

- [1] Akyildiz, I.F.; McNair, J.; Martorell, L.C.; Puigjaner, R.; Yesha, Y. Medium access control protocols for multimedia traffic in wireless networks. *IEEE Network*, vol.13, (no.4), IEEE, July-Aug. 1999. p.39-47.
- [2] Avancha, S., Chakraborty, D., Gada, D., Kamdar and T., Joshi, A., “*Fast and Effective Wireless Handoff Scheme using Forwarding Pointers and Hierarchical Foreign Agents*”, University of Maryland Baltimore County, <<http://userpages.umbc.edu/~kamdar/projects/691Treport.pdf>>
- [3] C.Boyd, “A class of flexible and efficient Key Management Protocols”, *proceedings of computer security foundation workshop*, pp 2-8, 1996.
- [4] Chung Kei Wong, Mohamed Gouda and Simon S. Lam, “Secure Group Communications Using Key Graphs”, *Networking, IEEE/ACM Transactions*, vol.8, pp 16-30, 2000.
- [5] Charles H.Roth, Jr , “ Fundamentals Of Logic Design”, Fourth Edition pp 161-167.
- [6] D.Muller, G.Schafer and J.Schiller, “An Efficient Authentication Protocol For High Performance Networks”, *Globecom 1998*, pp 886-891, 1998.
- [7] Eardley, P., Mihailovic, A. and Suihko, T., “*A Framework for the Evaluation of IP Mobility Protocols*”, BT, Advanced Communications Technology Centre, King’s College, London, Nokia / VTT Information Technology, The 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Volume 1, 2000, pp.451-457.
- [8] Fan Du, Lionel M.Ni and Abdol-Hossein Esfahanian, “Towards Solving Multicast Key Management Problem”, *proceedings of eight international conference on computer communication and networks*, pp 232-236, 1999.
- [9] Fu-Kuan Tu, Chi-Sung Lai and Hsu-Hung Tung, “On Key Distribution Management For Conditional Access System On Pay-TV System”, *IEEE transaction on Consumer Electronics*, pp 151-158, 1999.
- [10] Fumiaki Sato and Shin-ya Tanaka, “A Push-Based Key Distribution And Rekeying Protocol For Secure Multicasting”, *proceedings of eight international conference on Parallel and Distributed Systems ICPADS2001* pp 214-219, 2001.

- [11] G.Chaddoud, I.Chrisment and A. Schaff, "Dynamic Group Communication Security", *proceedings of sixth IEEE Symposium* pp 49-56, 2001.
- [12] Isabella Chang, Robert Engel, Dilip Kandlur, Dimitrios Pendarakis, Debanjan Saha, "Key Management For Secure Internet Multicast Using Boolean Function Minimization Technique", *proceedings of eighteenth annual conference of IEEE computer and communication societies INFOCOMM '99*, pp 689-698, 1999.
- [13] Jing Liu and Mingtian Zhou, "Key Management And Access Control For Large Dynamic Multicast Group", *proceedings of fourth IEEE international workshop*, pp 109-120, March 2002.
- [14] John R. Michener and Tolga Acar, "Security Domains: Key Management In Large-Scale Systems", *IEEE Software* vol.17, Monterey, USA, 2000.
- [15] Michel Adballa, Yuval Shavitt and Avishai Wool , "Key Management For Restricted Multicast Using Broadcast Encryption", *IEEE-ACM Transactions on Networking*, pp 443-454, 2000.
- [16] Mingyan Li, R.Poovendran and C.Berenstein, "Design Of Secure Multicast Key Management Schemes With Communication Budget Constraint", *IEEE communication letters*, pp 108-110, March 2002.
- [17] M.J.Moyer, J.R.Rao and P.Rohatgi, "A Survey Of Security Issues In Multicast Communications", *IEEE Networks* vol.13, pp12-23, 1999.
- [18] P.S.Kruss and J.P. Macker, "Techniques And Issues In Multicast Security", *proceedings of MILCOM' 98*, pp 1028, 1998.
- [19] R.Poovendran and J.S.Baras, "An Information Theoretic Approach To Secure Multicast Key Management", *Information Theory and Networking Workshop, 1999*, pp-36, 1999.
- [20] S. Setia, S. Koussih, S.Jajodia and E.Harder, "Kronos: A Scalable Group Re-keying Approach For Secure Multicast", *proceedings of IEEE Symposium* , pp 215-228, 2000.
- [21] Shin-ya Tanaka and Fumiaki Sato, "A Key Distribution And Rekeying Framework With Totally Ordered Multicast Protocols", *proceedings of fifteenth international conference on Information Networking* pp 831-838, 2001.
- [22] Tanenbaum, S.A., "*Computer Networks* ", Third Edition, 1996.

<http://www.iprg.nokia.com/~charliep/txt/optim/optim.txt>

- [23] T.Hardjona and B.Cain, "A Secure Group Membership Verification Protocol For IP Multicast", *proceeding of IEEE Symposium*, pp 09-15, 1999.
- [24] <www.cisco.com>

Appendix

Source code for simulation of Predictable Networks

```
# include <stdio.h>
# include <Math.h>
# include <string.h>
# include <stdlib.h>
# include <ctype.h>
# include <time.h>
# include <conio.h>

typedef struct tree
{
    int value;
    struct tree *lchild;
    struct tree *rchild;
    struct tree *parent;
}
myTree;

char codes[10000][20];

int *elements, total;

FILE *fp;

typedef struct myList
{
    myTree *node;
    struct myList *next;
}
lList;

lList *lhead=NULL;

lList *nodes=NULL;

myTree *thead=NULL;
int flag;
int inc=1;

myTree *extractmin(void);
void HuffmanCodes(void);
void Randomize(void);

myTree* extractmin(void)
{
    myTree *ret=lhead->node;
    lhead=lhead->next;
    return ret;
}
```

```

void push(myTree*);
void insert(myTree*);

void push(myTree *t)
{
    llist *temp1,*temp2;

    llist *nu;

    temp1=lhead;
    temp2=lhead;

    while((temp2!=NULL)&&(t->value>temp2->node->value))
    {
        temp1=temp2;
        temp2=temp2->next;
    }

    nu=(llist*)malloc(sizeof(llist));
    nu->node=t;

    if(lhead!=NULL)
    {
        nu->next=temp1->next;
        temp1->next=nu;
    }
    else
    {
        lhead=nu;
        flag=1;
    }
}

}

void store(int*,int);

void store(int* a,int n)
{
    int i,j,temp;
    llist *t;

    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
}

```

```

for(i=0;i<n;i++)
{
    if(lhead==NULL)
    {
        lhead=(l1ist*)malloc(sizeof(l1ist));
        lhead->node=(myTree*)malloc(sizeof(myTree));
        lhead->node->value=a[i];
        lhead->node->lchild=NULL;
        lhead->node->rchild=NULL;
        lhead->node->parent=NULL;
        lhead->next=NULL;
    }
    else
    {
        t=lhead;
        while(t->next!=NULL) t=t->next;
        t->next=(l1ist*)malloc(sizeof(l1ist));
        t->next->node=(myTree*)malloc(sizeof(myTree));
        t->next->node->value=a[i];
        t->next->node->lchild=NULL;
        t->next->node->rchild=NULL;
        t->next->node->parent=NULL;
        t->next->next=NULL;
    }
}

}

void mainProgram();

void main()
{
    int i;
    fp=fopen("output.dat","w+");
    for(i=0;i<9;i++)
    {
        mainProgram();
        inc++;
    }
    getch();
}

void mainProgram()
{
    int i;
    int binary=0;
    int n=0;
    myTree *Tree=NULL;
    int bits=0;
    lhead=NULL;
    thead=NULL;
    nodes=NULL;
    flag=0;

    /*for(i=0;;i++)
    {

```



```

        scanf("%d",&input[i]);
        if(input[i]==-1) break;
        size++;
    }*/

    Randomize();

    store(elements,total);

    while(!flag)
    {
        Tree=(myTree*) malloc(sizeof(myTree));
        Tree->parent=NULL;
        Tree->lchild=extractmin();
        Tree->lchild->parent=Tree;
        if(
            (Tree->lchild->lchild==NULL)&&
            (Tree->lchild->rchild==NULL)
        )
        {
            insert(Tree->lchild);
        }

        Tree->rchild=extractmin();
        Tree->rchild->parent=Tree;

        if(
            (Tree->rchild->lchild==NULL)&&
            (Tree->rchild->rchild==NULL)
        )
        {
            insert(Tree->rchild);
        }

        Tree->value=Tree->lchild->value+Tree->rchild->value;
        push(Tree);
    }

    for(i=0;i<total;i++)
    {
        strcpy(codes[i],"");
    }

    HuffmanCodes();

    printf("\n");

    for(i=0;i<total;i++)
    {
        if(i<=total/2) bits+=strlen(codes[i]);
    }

```

```

    fprintf(fp, "%d\t", bits);

    n=total;

    while (n>0)
    {
        n/=2;
        binary++;
    }

    fprintf(fp, "%d\n", binary*total/2);
    printf(
        "The simulation with %d users and %d of them left\n",
        total, total/2
    );

    printf("Normal Binary Notation %d\n", binary*total/2);
    printf("Modified Huffman Approach %d\n", bits);

}

void insert(myTree *p)
{
    llist *t;
    if (nodes==NULL)
    {
        nodes=(llist*)malloc(sizeof(llist));
        nodes->node=p;
        nodes->next=NULL;
    }
    else
    {
        t=(llist*)malloc(sizeof(llist));
        t->node=p;
        t->next=nodes;
        nodes=t;
    }
}

void HuffmanCodes(void)
{
    llist *temp;
    myTree *p,*q;
    int i=0;

    temp=nodes;
    while (temp!=NULL)
    {
        p=temp->node;
        q=temp->node->parent;
    }
}

```

```

        while (q!=NULL)
        {
            if (q->lchild==p)
            {
                strcat (codes [i], "0");
            }
            else
            {
                strcat (codes [i], "1");
            }
            p=q;
            q=q->parent;
        }
        strrev (codes [i]);
        i++;

        temp=temp->next;
    }
}

void Randomize (void)
{
    int i=0;
    srand( (unsigned)time( NULL ) );

    total=250*inc;

    elements=(int*)malloc (total*(sizeof(int)));

    for (i=0; i<total; i++)
    {
        elements [i]=rand()%(50*inc);
    }
}

```

VITA 2

Senthamil Ilango

Candidate for Degree of

Master of Science

Thesis: GROUP KEY MANAGEMENT UTILIZING HUFFMAN AND PETRICK
BASED APPROACH

Major Field: Computer Science

Biographical:

Personal Data: Born in Tiruvarur, Tamilnadu, India, On September 18,
1978, the son of Mr. S. Ilango and Mrs.I.Hemalatha

Education: Graduated from N.H.S.S, Nagapattinam,
Tamilnadu, India in April 1995:Received Bachelor of Engineering in
Computer Science and Engineering from the University of Madras,
Chennai, Tamilnadu, India in April 1999.
Completed the requirements for the Master of Science degree with a
major in Computer Science at Oklahoma State University, Stillwater,
Oklahoma in May 2003.

Experience: Employed as Software Engineer in Novellus Systems,
Sunnyvale CA from Aug2002 till date.