EFFICIENT MULTICAST KEY DISTRIBUTION

By

ANJU.P.JOSE

Reg. No: 0820108001

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna university, Coimbatore)

COIMBATORE – 641 006

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements
for the award of the degree
of

MASTER OF ENGINEERING
IN

COMPUTER SCIENCE AND ENGINEERING

MAY, 2010

BONAFIDE CERTIFICATE

Certified that this project report titled "**Efficient Multicast Key Distribution**" is the bonafide work of **ANJU.P.JOSE(0820108001)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report of dissertation on the basis of which a degree or ward was conferred on an earlier occasion on this or any other candidate.

GUIDE

(Mrs. N.SUGANTHI)

HEAD OF THE DEPARTMENT

(Dr.S.THANGASAMY)

The candidate with **University Register No. 0820108001** was examined by us in Project Viva-Voce examination held on 17.05.2010

17/5/10

Internal Examiner

External Examiner

VELALAR COLLEGE OF ENGINEERING AND TECHNOLOGY

THINDAL, ERODE - 12.

NATIONAL CONFERENCE ON RECENT TRENDS IN INNOVATIVE TECHNOLOGIES

NCRTIT '10

Organized by ***Department of Computer Science and Engineering***

CERTIFICATE

This is to certify that ANJU.P.JOSE, PG STUDENT **of** KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE **has presented a paper titled** EFFICIENT MULTICAST KEY DISTRIBUTION USING REEDSOLOMON CODE **in the NATIONAL CONFERENCE ON RECENT TRENDS IN INNOVATIVE TECHNOLOGIES (NCRTIT '10) held on March 27, 2010.**

Convener

Principal i/c

Administrative

ABSTRACT

Efficient key distribution is an important problem for secure group communications. The communication and storage complexity of multicast key distribution problem has been studied extensively. In the proposed system instead of using conventional encryption algorithms, reed Solomon code is used to distribute multicast key dynamically. This scheme drastically reduces computation load of each group member compared to existing schemes employing traditional encryption algorithms. There is a Group Controller to distribute the session keys among the group members. The group members change dynamically that is, some new members will join to the group while some old members will be excluded. So session keys are changed dynamically in order to ensure both forward secrecy and backward secrecy. Forward secrecy is maintained when an old member has been excluded from the current session and not allowed to access the current session. Also backward secrecy is maintained when a new member joins to the current session and not allowed to recover the communication of past session. This scheme is combined with a key-tree, this scheme provides much lower computation complexity, for secure dynamic multicast key distribution.

ஆய்வுச்சுருக்கம்

இன்றைய காலக்கட்டதில் ஒரு பிணையத்தில் தகவல் பரிமாறுதல் மிக கவனமாகவும் ரகசியமாகவும் இருக்க வேண்டும் இல்லையென்றால் அந்த தகவல்கள் மற்றவர்களுக்கு வெளிப்பட்டு தவரான மாறுதல்கள் எற்பட வாய்ப்புண்டு.

அந்த நிகழ்வினால் பிணையதில் பெரும் மாறுதல்கள் ஏற்படும் மற்றும் பிணைய ஆதரங்களுக்கும் பாதுகாப்பு இல்லாமல் போய்விட வாய்ப்புண்டு.

இந்த ஆய்வில் தகவல் பரிமாற்றத்தை பாதுகாக்க குழு தணிக்கையாளர் ஒருவர் பிணையில் இருக்கும் எல்லா உருப்பினர்களுக்கும் பொது விசை வினியோகம் செய்வார். ஒரு புது உருப்பினர் பிணையில் சேரும் போது எல்லா உருப்பினர்களுக்கும் வேறு விசை வினியோகம் செய்வார், அப்படி வினியோகம் செய்வதற்க்கு குறியேற்றம் செய்ய ரீட்- சோலொமொன் என்ற முறையை பின்பற்றினால் நாம் எதிர்ப்பார்க்கும் பாதுகாப்பான தகவல் பரிமாற்றத்தை பெற மிக உதவியாக இருக்கும்.

ACKNOWLEDGEMENT

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam B.sc, F.I.E** and Correspondent **Shri.Balasubramanian M.Com, M.B.A (USA)** and joint Correspondent **Dr.A.Selvakumar Ph.D.,** for giving this great opportunity to pursue this course.

I thank **Dr.S.Rama Chandran,** Principal, Kumaraguru College of Technology, Coimbatore, for providing me with the necessary facilities and Infrastructure to work on this project.

I express my sincere thanks to **Dr.S.Thangasamy,** Professor and Dean, Department of Computer Science and Engineering, for being my greatest of inspiration and for embedding the quest for innovative ideas.

I record my thanks to the My Guide **Mrs.N.Suganthi, M.E.,** Assistant professor, Department of Information Technology, who has extraordinarily helpful in preparing this project work successfully.

I express my deep sense of gratitude and gratefulness to Project Coordinator **Mrs.V.Vanitha, M.E, (Ph.D),** Asst. Professor, Department of Computer Science and Engineering Information, for her supervision, tremendous patience, active involvement and guidance.

I would like to convey my honest thanks to all Teaching staff members and Non Teaching staff members of the department for their support. I would like to thank all my classmates who gave me proper light moments and study breaks apart from extending some technical support whenever I needed them most.

I dedicate this Project work to my **Parents** for no reasons but feeling from bottom of my heart, without their love this work wouldn't possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
MANET	Mobile Ad hoc NETwork
WLANS	Wireless Local Area Networks
DES	Data Encryption Standard
GC	Group Controller
AES	Advanced encryption Standard
GF	Galois Field
RS	Reed Solomon
MDS	Maximum Distance Separable
GUI	Graphic User Interface
AWT	Abstract Window Tool
JFC	Java Foundation Classes
GKMP	Group Key Management Protocol

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF NETWORK SECURITY

In the field of networking, the specialist area of network security consists of the provisions made in an underlying computer network infrastructure, policies adopted by the network administrator to protect the network and the network-accessible resources from unauthorized access, and consistent and continuous monitoring and measurement of its effectiveness (or lack) combined together.

Network security involves all activities that organizations, enterprises, and institutions undertake to protect the value and ongoing usability of assets and the integrity and continuity of operations [11].

1.1.1 Network Security Tools

Antivirus software packages: These packages counter most virus threats if regularly updated and correctly maintained.

Secure network infrastructure: Switches and routers have hardware and software features that support secure connectivity, perimeter security, intrusion protection, identity services, and security management. Dedicated network security hardware and software-Tools such as firewalls and intrusion detection systems provide protection for all areas of the network and enable secure connections.

Virtual private networks: These networks provide access control and data encryption between two different computers on a network. This allows remote workers to connect to the network without the risk of a hacker or thief intercepting data.

Identity services: These services help to identify users and control their activities and transactions on the network. Services include passwords, digital certificates, and digital authentication keys.

Encryption: Encryption ensures that messages cannot be intercepted or read by anyone other than the authorized recipient [11].

1.2 NETWORKS

There are wired and wireless networks. In the recent years wireless networks have witnessed a tremendous increase of popularity in both research and industry. There are currently two variations of mobile networks. The first is widely known as infrastructure networks since the gateways that connect them to other networks (like the internet) are fixed and wired. The bridges in these networks are also known as base stations. In an environment like this, a node is able to roam freely and establish a connection link with the nearest base station that is within its communication range. As the mobile node moves out of the base station that it was connected with, it falls into the range of another and hand off occurs between the old base station and the current one, enabling the mobile unit to continue communication seamlessly through the network. These types of networks are most widely applied in office areas and include the wireless local area networks (WLANS).

The second type of wireless networks is the infrastructure less mobile network that is also known as an ad hoc network. Infrastructure less mobile network has no fixed routers and base stations and the participating nodes are capable of movement. Due to the limited transmission range, multiple hops may be required for nodes to communicate across the ad hoc network. Routing functionality is incorporated into each host, thus ad hoc networks can be characterized as having dynamic, multi-hop and constantly changing topologies.

1.2.1 Mobile Ad Hoc Networks

Mobile ad hoc networks are a new paradigm of wireless communication for mobile hosts (which we call nodes). An ad hoc network is a collection of two or more devices equipped with wireless communications and networking capability. In an ad hoc network, there is no fixed infrastructure such as base stations or mobile switching centers.

A mobile ad hoc network is a self-configurable, self organizing, infrastructure less multi-hop mobile wireless network. Mobile nodes that are within each other's radio range communicate directly via wireless links, while those that are outside their radio range, use an intermediate node to relay or forward the packet from the source towards the destination. Self-configuring and self-organizing means, that a network can be formed on the fly without the need for any system administration.

Each node in a MANET is capable of moving independently, thus the network topology can change continuously and dramatically. Each node also functions as a router that discovers and maintains routes to other nodes and forwards packets for other nodes. A MANET can be deployed without any wired base stations or infrastructure support. These features make a MANET very attractive or reliable, but fast network establishment and self-reconfiguration are required [10].

1.3 MULTICAST

Multicast protocols are a growing area of interest for the internet. The largest benefit of a multicast protocol is the ability of several receivers to simultaneously get the same transmission. If the transmission is of a sensitive nature, it should be encrypted. This means that all the members of the group must share the same encryption key to take benefit of the multicast transmission

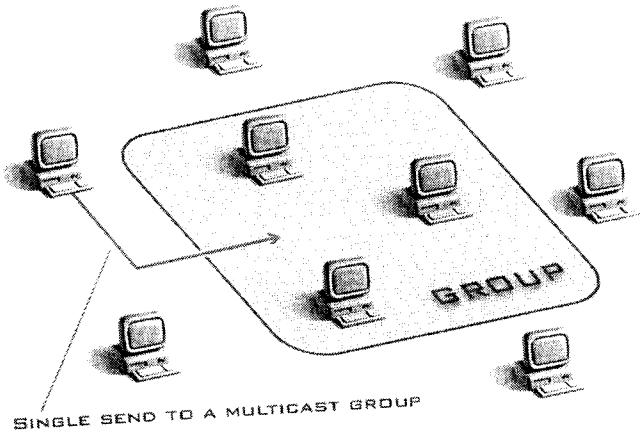


Figure 1.1 Multicast to the group members

1.4 KEY MANAGEMENT

Key management is the provisions made in a cryptography system design that are related to generation, exchange, storage, safeguarding, use, vetting, and replacement of keys. It includes cryptographic protocol design, key servers, user procedures, and other relevant protocols. Key management concerns keys at the user level, either between users or systems. This is in contrast to key scheduling; key scheduling typically refers to the internal handling of key material within the operation of a cipher.

Successful key management is critical to the security of a cryptosystem. In practice it is arguably the most difficult aspect of cryptography because it involves system policy, user training, organizational and departmental interactions, and coordination between all of these elements. These concerns are not limited to cryptographic engineering. Key management requires both technical and organizational decisions, and as a result, some aspects of key management risk being neglected by managers and engineers, out of concern that the problem is technical or managerial, respectively [2].

1.4.1 Symmetric Key Management

Symmetric key management is the key management of cryptographic symmetric encryption keys. In a symmetric key algorithm the keys involved are identical for both encrypting and decrypting a message. Such keys must be chosen carefully, and distributed and stored securely. In any system there may be multiple keys for various purposes. Accordingly, key management is central to the successful and secure use of symmetric key algorithms.

1.5 KEY DISTRIBUTION

In symmetric key cryptography, both parties must possess a secret key which they must exchange prior to using any encryption. Distribution of secret keys has been problematic until recently, because it involved face-to-face meeting, use of a trusted courier, or sending the key through an existing encryption channel. The first two are often impractical and always unsafe, while the third depends on the security of a previous key exchange.

In public key cryptography, the key distribution of public keys is done through public key servers. When a person creates a key-pair, he keeps one key private and the other, public-key, is uploaded to a server where it can be accessed by anyone to send the user a private, encrypted, message [4].

CHAPTER 2

LITERATURE SURVEY

2.1 MULTICAST KEY MANAGEMENT ARCHITECTURE

There are several electronic mechanisms for generating and distributing symmetric keys to several computers (i.e., communications groups). These techniques, generally, rely on a key distribution center (KDC) to act as a go between in setting up the symmetric key groups. Military systems, such as BLACKER, STU-II/BELLFIELD, and EKMS, and commercial systems, such as X9.17 and Kerberos, all operate using dedicated KDCs. A group key request is sent to the KDC via various means (on- or off-line) The KDC acting as an access controller decides whether or not the request is proper (i.e., all members of a group are cleared to receive all the data on a group). The KDC would then call up each individual member of the group and down load the symmetric key. When each member had the key the KDC would notify the requester. Then secure group communication could begin. While this was certainly faster then anything that requires human intervention. It still requires quite a bit of set-up time. Also, a third party, whose primary interest isn't the communication, needs to get involved.

Pairwise keys can be created autonomously by the host on a network by using any number of key generation protocols (FireFly, Diffe-Hellman, RSA). These protocols all rely on cooperative key generation algorithms to create a cryptographic key. These algorithms rely on random information generated by each host. These algorithms also rely on peer review of permissions to ensure that the communication partners are who they claim to be and have authorization to receive the information being transmitted. This peer review process relies on a trusted authority assigning permissions to each host in the network that wants the ability to create these keys. The real beauty of these pairwise key [illegible] is that they can be integrated into the communication

protocol or the application. This means that the key management becomes relatively invisible to the people in the system [10] .

2.2 MULTICAST KEY DISTRIBUTION

The group key distribution problem has been studied extensively in the larger context of key management for secure group communications mainly on balancing the storage complexity and the communication complexity. There are two trivial schemes for distributing a session key to a group of n members. The first one is that the GC shares an individual key with each group member, which can be used to encrypt a new group session key. In this scheme, the communication complexity is O(n), whereas the GC needs to store O(n) key information, each member stores O(1) key information, and O(n) encryption and decryption operations are needed.

In the second scheme, the GC shares an individual key with each subset of the group, which can then be used to multicast a session key to a designated subset of group members. Now, both the communication complexity and the computation complexity reduce to O(1), but at the cost of increasing the storage complexity to O(2n) for both the GC and each group member. It is easy to see that neither scheme works for practical applications with a reasonable group size n. Thus, research efforts have been made to achieve low communication and storage complexity for group key distribution. However, this threshold-based scheme can only distribute a session key to a designated group of members for one-time use.

Once a session key is distributed to the group, any member can calculate the secret information that other members in the same group hold. Thus, the scheme does not provide forward or backward secrecy. However, its prohibitively high communication complexity and computation complexity make it only practical for a very small group with limited number of members. Along the same line, many research efforts have been made on balancing communication complexity and storage complexity of the group key distribution problems [1].

2.3 GROUP KEY MANAGEMENT PROTOCOL

The group manager interacts with other management functions in the network to provide the group key management protocol (gkmp) with group membership lists and group relevant commands. The gkmp deals strictly with cryptographic key. It relies on external communication and network management services to supply network related information. Primarily, it relies on the network management service to provide it with the addresses of group members (if the group is sender initiated).The gkmp allows an external entity to determine the controller of a group. The controller of the group should be able to handle the additional processing and communication requirements associated with the role. If this is not a necessary function given the implementation, this assignment of controller duties can be set to some automated default. However, even if defaulted some external management entity determines how the role of controller is allocated.The group manager can receive group progress reports from the groupcontroller. The gkmp provides a service for the network. It makes sense that someone in the network is interested in the progress of this service. The gkmp can provide progress reports. It is up to the network management to determine the manner and recipient of the reports.

Creation and distribution of grouped key require assignment of roles.these identify what functions the individual hosts perform in the protocol. The two primary roles are those of controller and receiver. The controller initiates the creation of the key, forms the key distribution messages, and collects acknowledgment of key receipt from the receivers. The receivers wait for a distribution message, decrypt, validate, and acknowledge the receipt of new key. One of the essential concepts behind the gkmp is delegation of group control. Since each host in the network has the capability to act as a group controller, the processing and communication requirements of controlling the groups in the network can be distributed equitably throughout the network. This protocol uses traditional encryption algorithm [10].

2.4 BATCH REKEYING FOR SECURE GROUP COMMUNICATION

In batch rekeying, the key server waits for a period of time, called a rekey interval, collects the entire join and leave requests during the interval, generates new keys, constructs a rekey message and multicasts the rekey message. Batch rekeying reduces the number of rekey messages to be signed: one for a batch of requests. But the server has to wait for a interval to collect the number of join and leave. The key server cannot control which users might leave, but it can control where in the key tree to place the new users.

Given a batch of requests, the main task for the key server is to identify which keys should be added, deleted, or changed. In individual rekeying, all the keys on the path from the request location to the root of the key tree have to be changed. When there are multiple requests, there are multiple paths. These paths form a subtree, called rekey subtree, which includes all the keys to be added or changed. The rekey subtree does not include individual keys. The key server cannot control which users might leave, but it can control where in the key tree to place the new users. Thus, the key server should carefully place the new users (if there were any) so that marking algorithm is used. Marking algorithm is time consuming and more complex [4] .

2.5 ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM

In cryptography, the AES is an encryption standard adopted by the U.S. government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each of these ciphers has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor the Data Encryption Standard (DES) [12].

2.5.1 Description Of The Cipher

AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware. Unlike its predecessor, DES, AES does not use a feistel network.AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits, with a minimum of 128 bits. The blocksize has a maximum of 256 bits, but the keysize has theoretically no maximum.

AES operates on a 4×4 array of bytes, termed the *state* (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a special finite field.The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key [12].

2.5.2 High-Level Description Of The Algorithm

2.5.2.1 Key Expansion

Round keys are derived from the cipher key using Rijndael's key schedule.

2.5.2.2 Initial Round

- AddRoundKey :- Each byte of the state is combined with the round key using bitwise xor.

2.5.2.3 Rounds

- SubBytes:- A non-linear substitution step where each byte is replaced with another according to a lookup table.
- ShiftRows:- A transposition step where each row of the state is shifted cyclically a certain number of steps.
- MixColumns:- A mixing operation which operates on the columns of the state, combining the four bytes in each column
- AddRoundKey

2.5.2.4 Final Round

- SubBytes
- ShiftRows
- AddRoundKey

CHAPTER 3

METHODOLOGY

3.1 OVERVIEW

The approach used can be classified in 3 steps: (i) understanding the problem and existing solutions, (ii) implementing and evaluating proposed solution, and (iii) comparing the computation time with existing solutions.

There has been a large number of existing approaches. It is therefore necessary to understand the fundamental differences of various approaches to solving the problem. Having understood the pros and cons of them, one which works best in most scenarios is chosen.

3.2 UNDERSTANDING EXISTING APPROACHES

Group key management schemes must be able to adjust group secrets subsequent to membership changes, including single-user addition, single-user deletion. Single-user addition means that one member joins the group and single-user deletion means that one member leaves the group. The security requirements with dynamic membership include group key secrecy, forward secrecy, backward secrecy, and key independence. Group key secrecy, which is the most basic property, requires that it should be computationally infeasible for a passive adversary to discover any group key [2].

Forward secrecy requires that a passive adversary who knows a contiguous subset of old group keys cannot discover subsequent group keys, whereas backward secrecy requires that a passive adversary who knows a contiguous subset of group keys cannot discover preceding group keys. Key independence, which is the strongest property, requires that a passive adversary who knows a proper subset of group keys cannot discover any other group key. According to, key independence can be achieved when both forward secrecy and

backward secrecy are achieved. . As a part of security, the keys are encrypted and decrypted using traditional encryption and decryption algorithms. Here Encryption algorithm used is Advanced Encryption Standard (AES) algorithm. Existing system has the overhead of computation cost.

Proposed system drastically reduces the computation complexity of group controller compared to existing system employing traditional encryption algorithms.

3.3 PROPOSED SYSTEM

Key is distributed for the secure communication among the group members. There is a Group Controller to distribute the session keys among the group members. The group members change dynamically i.e, some new members will join to the group while some old members will be excluded. So session keys are changed dynamically in order to ensure both forward secrecy and backward secrecy. Forward secrecy is maintained when an old member has been excluded from the current session and not allowed to access the current session. Also backward is maintained when a new member join to the current session and not allowed to recover the communication of past session. Due to this dynamic change of member's, session key is changed for each current session. Session key is a secret key that is shared between the group controller(GC) and group members. GC also distribute individual seed key to each group members. Individual seed key is shared between the GC and the corresponding member. Seed key is considered as an random number. Seed key is denoted as 'S_i' [1].

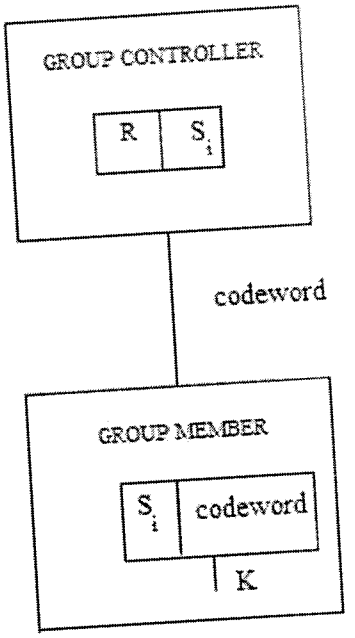


Figure 3.1 Block Diagram Of Proposed System

For a multicast group with a large number of members, key tree based scheme is introduced to decompose a large group into multiple layers of subgroups with smaller sizes. It has been assumed that encryption and decryption algorithms are used to distribute session keys, but it is time consuming. In the proposed system, a new dynamic group key distribution scheme that reduces the time taken to distribute session keys. In our scheme, information related to session keys are encoded using reed solomon code rather than encryptions. In general, encoding and decoding using reed solomon code have much lower computation time than existing encryption and decryption algorithms [1].

3.3.1 Basic Scheme

The Group Controller generates random numbers and these numbers are passed to the reed solomon code. The reed solomon code encodes these numbers to get the resultant is in an encoded form. By using seed key performing shift and xor operation on the resultant to form codeword. When a new member joins the group, GC first verifies the ID of that member then only it sends the encoded session key to that member. GC also sends seed key to that member. After receiving the codeword, the group member performs shift and xor operation by using its seed key. Then the resultant value is passed to the reed solomon code for decoding. After decoding the original session key is retrieved.

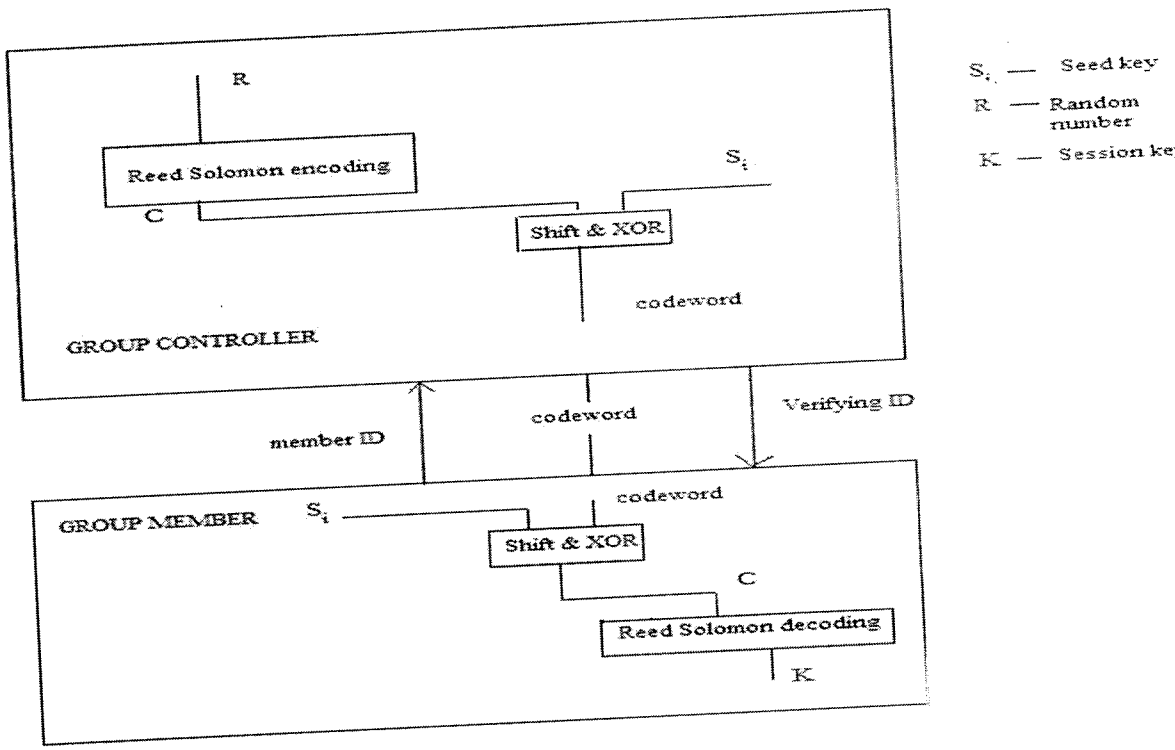


Figure 3.2 Generation of session key

3.3.2 Rekeying Scheme

Rekeying is done whenever some new members join or some old members leave a multicast group, the GC needs to distribute a new session key to all the current members. Seed key remains valid until the corresponding member leaves the multicast group permanently. Thus the seed key is unicast only once to the corresponding member. In the rekeying procedure, each time GC generates new session for the particular group. The random number 'r' is used to guarantee that the new session key is different from all the old keys used. After an old member leaves, the GC needs to distribute a new session key to the remaining members to achieve both forward and backward secrecy of the session key. In this the group controller executes the rekeying process and sends the key to the member and when the authorized member of the group receives a codeword from the group controller, it can decode the key that is send to it by the group controller [1].

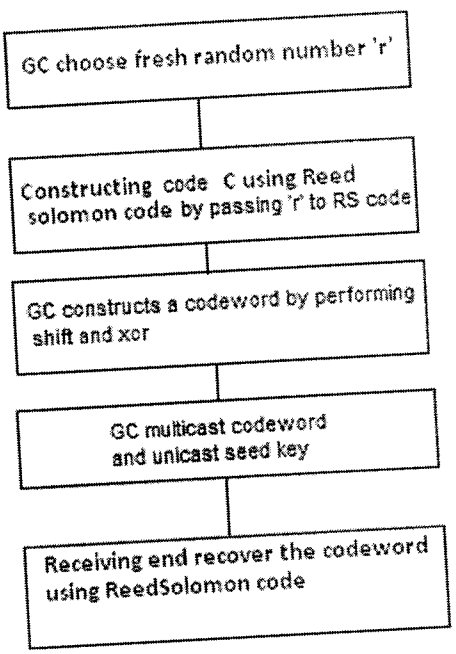


Figure 3.3 Block Diagram for Rekeying Process

3.3.3 Key-Tree Based Rekeying Scheme

To reduce the time taken by rekeying operation, a key-tree based scheme is introduced. For that the GC distribute subgroup keys in addition to individual seed keys and the session key. These keys are arranged in a tree like structure, where the session key serves as the root, the individual seed keys are the leaves, and the subgroup keys correspond to intermediate nodes. Each member stores all the keys along the path from the corresponding leaf to the root in the tree. Then each subgroup key can be used to securely multicast to the members that are leaves of the corresponding sub tree [1].

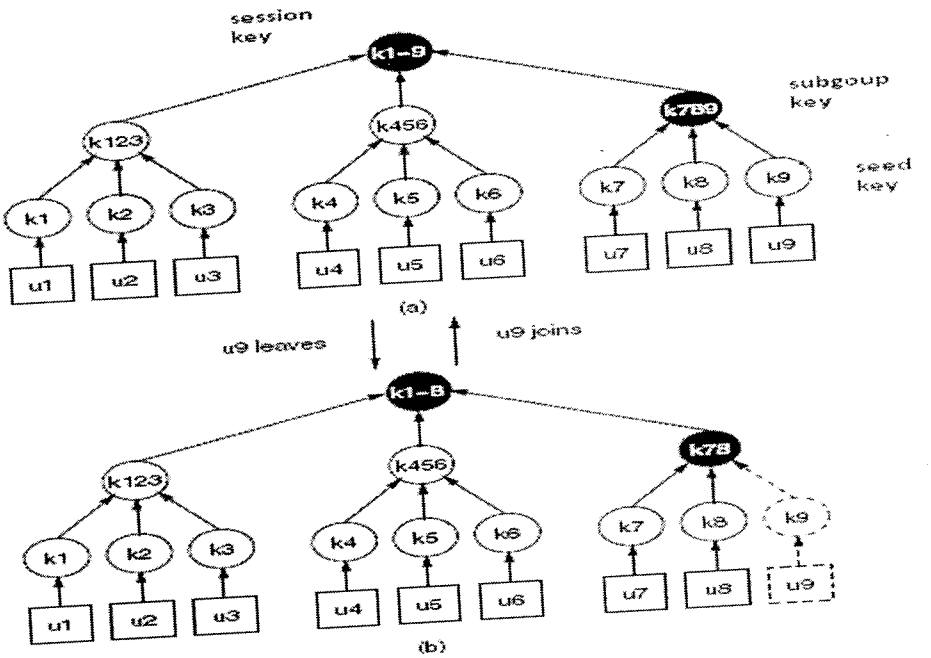


Figure 3.4 An example for key tree

During the rekeying process, the GC can thus multicast to a subgroup of members using their shared subgroup key instead of individual seed key. In a key tree, the root is the session key, leaf nodes are individual seed keys, and the other nodes are subgroup keys. Consider a group of 9 users u1... u9. A key tree

and circles represent key nodes. User u9 is given 3 keys on the path from u9 to k1-9 are k9, k789, and k1-9. k1-9 is the session key and k9 is u9's individual seed key, which is shared by only u9 and the Group Controller. Subgroup key k789 is shared by u7, u8, and u9. Suppose u9 wants to leave the group, then the GC needs to change the keys that u9 has to change k1-9 to a new key k1-8 and k789 to k78 [4].

3.3.4 Reed Solomon Code

Reed Solomon codes are class of MDS code. It is a linear code and a coding scheme. Reed Solomon code contains encoding part and decoding part. Reed-Solomon encoding part adds extra bits to the original data. Reed-Solomon decoding part recovers the original data. A Reed-Solomon code is specified as RS (n,k) with s-bit symbols. This means that the encoder takes 'k' data symbols of 's' bits each and adds parity symbols to make an 'n' symbol codeword. Decoder will recover the data from codeword. GC constructs the codeword by using reed solomon encoding part and group member recovers the original data by using reed solomon decoding portion [5].

Reed Solomon (RS) codes are invented by Irving S. Reed and Gustave Solomon. They described a systematic way of building codes that could detect and correct multiple random symbol errors. By adding t check symbols to the data, an RS code can detect any combination of up to t erroneous symbols, and correct up to $\lfloor t/2 \rfloor$ symbols. As an erasure code, it can correct up to t known erasures, or it can detect and correct combinations of errors and erasures.

In Reed-Solomon coding, source symbols are viewed as coefficients of a polynomial p(x) over a finite field. The original idea was to create n code symbols from k source symbols by oversampling p(x) at $n > k$ distinct points, transmit the sampled points, and at the receiver it will recover the original message. That is not how RS codes are used today. Instead, RS codes are viewed as cyclic BCH codes, where encoding symbols are derived from the coefficients of a polynomial

constructed by multiplying p(x) with a generator polynomial. This gives rise to an efficient decoding algorithm, which was discovered by Elwyn Berlekamp and James Massey, and is known as the Berlekamp-Massey decoding algorithm.

Reed-Solomon codes have since found important applications from deep-space communication to consumer electronics. They are prominently used in consumer electronics such as CDs, DVDs, Blu-ray Discs, in data transmission technologies such as DSL & WiMAX, in broadcast systems such as DVB and ATSC, and in computer applications such as RAID 6 systems.

Overview of the method

The data points are sent as an encoded block. The total number of m-bit symbols in the encoded block is $n=2^m-1$. Thus a Reed-Solomon code operating on 8-bit symbols has $n=2^8-1 = 255$ symbols per block. The number k, $k < n$, of data symbols in the block is a design parameter. A commonly used code encodes k = 223 8-bit data symbols plus 32 8-bit parity symbols in a n = 255-symbol block; this is denoted as a (n, k) = (255,223) code.

Properties of Reed-Solomon codes

The error-correcting ability of any Reed-Solomon code is determined by n−k, the measure of redundancy in the block. If the locations of the error symbols are not known in advance, then a Reed-Solomon code can correct up to (n−k)/2 error symbols, i.e., it can correct half as many errors as there are redundant symbols added to the block. Sometimes error locations are known in advance, these are called erasures. A Reed-Solomon code is twice as powerful at erasure correction than at error correction, and any combination of errors and erasures can be corrected as long as the equation $2E + S \leq (n-k)$ is satisfied, where E is the number of errors and S is the number of erasures in the block [5] .

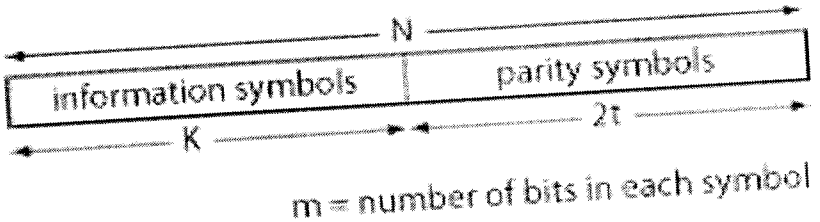


Figure 3.5 RS (n,k)

3.3.4.1 Reed Solomon Encoding Process

An overview of RS encoding process with a short data as an example, i.e. RS (7,3). It means that the RS codes have seven codeword symbols, three of which are the original information symbols and the rest are parity symbols. Let each is 3-bit symbol. The information that will be transmitted is divided into three 3-bit blocks, each of which is added by four 3-bit symbols to make 7 codewords symbols. The field elements for m = 3 using primitive polynomial $x^3 + x + 1$ are presented in Table 1.

Elemen	Polynomial	α_2	α_1	α_0
0	0	0	0	0
α^0	1	0	0	1
α^1	α	0	1	0
α^2	α^2	1	0	0
α^3	$\alpha+1$	0	1	1
α^4	$\alpha(\alpha+1)= \alpha^2+\alpha$	1	1	0
α^5	$\alpha^3+\alpha^2=\alpha^2+\alpha+1$	1	1	1
α^6	α^2+1	1	0	1

Table 1. Elements of GF(2^3)

Let a block of information is 100011110. By dividing it into three 3-bit symbols (100 011 110), we have the information symbols: a^2 a^3 a^4 where 'a' equals α.

A Reed Solomon codeword is constructed using a special polynomial called generator polynomial. All valid codewords are divisible by the generator polynomial. The general form of a generator polynomial is:

$$g(x) = (x + a^i)(x + a^{i+1})...(x + a^{i+2t-2})(x + a^{i+2t-1})$$

and the codeword is constructed using:

$$c(x) = g(x).i(x)$$

where g(x) is the generator polynomial, i(x) is the information block, c(x) is a valid codeword.

3.3.4.2 Reed Solomon Decoding Process

In the decoding part, transmit s(x) divisible by generating polynomial g(x).Received r(x) will have errors: r(x) = s(x) + e(x).The received polynomial is evaluated at the roots of g(x); these values are the syndromes. Since g(x) divides s(x), s(x) has no effect on the syndromes and is essentially eliminated. Only e(x) affects the syndromes, and the goal becomes using the syndromes to determine e(x) (which can be used to reconstruct s(x) = r(x) − e(x)).

The Berlekamp–Massey algorithm is an algorithm that will find the minimal polynomial of a linearly recurrent sequence in a field. The algorithm became the key to practical application of the now ubiquitous Reed–Solomon code. Berlekamp–Massey algorithm is specialized for the typical binary finite field and GF(2). The field elements are 0 and 1. The field operations + and − are identical and become the exclusive or operation (xor). The multiplication operator becomes the logical and operation. The division operator reduces to the identity operation (i.e., field division is only defined for dividing by 1, and x/1 = x).

The algorithm for the binary field

1. Start
2. Let s0,s1,s2.....sn be the bits of the stream.
3. Initialise two arrays b and c each of length n to be zeroes, except b0=1, c0=1.
4. Assign L=0, m=-1.
5. For N=0 step1 while N<n :

   - Let d be $S_N+C1S_{N-1}+.....+C_LS_{N-L}$
   - If d=0, then C is already a polynomial which annihilates the portion of the stream from N-L to N.
   - Else

     i)Let t be a copy of C

     ii) Set $c_{N-m} \leftarrow c_{N-m} \oplus b_0, c_{N-m+1} \leftarrow c_{N-m+1} \oplus b_1, \ldots$ up to $c_{n-1} \leftarrow c_{n-1} \oplus b_{n-N+m-1}$(where $\oplus$is the Exclusive or operator).

     iii) If L=N/2, set L=N+1-L, set m=N, and let b=t ; otherwise leave L,m,b alone.

6. Stop

3.3.5 Reed Solomon Code Implementing On Rekeying

3.3.5.1 Steps for constructing codeword

1. Start
2. Generating random numbers.
3. Passing random numbers to an array.
4. [illegible]ration on the data stored in the array.

5. Multiplying some integer values with the step 3 result value to form codeword.
6. Stop

3.3.5.2 Steps for recovering codeword

1. Start
2. Dividing the codeword using some integer values that is used for the construction of codeword.
3. Perform shifting operation on the result got from step1.
4. From the step 2 original data is recovered.
5. Passing the data to an array.
6. Stop

CHAPTER 4

TECHNOLOGY INFRASTRUCTURE

4.1 CORE JAVA

Java can be used to create two types of programs: application and applet. An application is a program that runs on your computer, under the operating system of that computer. That is, an application created by java is more or less like one created using C or C++. When used to create application, java is not much different from any other computer language. Rather, it is java's ability to create applets that makes it important. An applet is an application designed to be transmitted over the internet and executed by a java-compatible Web Browser. An applet is actually a tiny java program, dynamically downloaded across the network, just like an image, sound file, or video clip. The important difference is that an applet is an intelligent program, not just an animation or media file. In other words, an applet is a program that can react to user input and dynamically change-not just run the same animation or sound over and over [8].

Java having a major role in internet and the intranet application. The reason for this is quite simple: Java expends the universe pf objects that can move about freely in cyberspace. In a network, two very broad categories ob objects are transmitted between the server and your personal computer: passive information and dynamic, active programs.

Security

As you are likely aware, every time that you download a "normal" program, you are risking viral infection. Prior to java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Even so, most users still worried about the possibility of infecting their system with a virus. When you use a java-compatible web browser, you can [illegible] without fear of viral infection or malicious intent.

Java achieves this protection by confining a java program to the java execution environment and not allowing it access to other parts of computer.

Portability

Many types of computers and operating systems are in use throughout the world-and many are connected to the internet. For program to be dynamically downloaded to all the various type of platforms connected to the Internet, some means of generating portable executable code is needed.

Bytecode

The key that allows java to solve both the security and the portability problems just described is that output of a java compiler is not executable code. Rather, it is BYTECODE. Byte code is a highly optimized set of instruction designed to be executed by the java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreted code.

Simple

Java was designed to be easy for the professional programmer to learn and use effectively. Assuming that you have some programming experience, you will not find java hard to master. If we know the basic concept of object-oriented programming, learning java will be even easier.

Object-Oriented

Object-Oriented programming is the core of java. In fact, all java programs are object-oriented-this isn't an option the way that it is in C++, for example. OOP is so integral to java that you must understand its basic principles before you can write even simple java programs.

Abstraction

The essential element of object-oriented programming is abstraction. Humans manage complexity through abstraction. For example, people do not think of a car as a set of ten of individual parts. They think of it as a well-defined object with its own unique behavior. So this ignore the details of how the engine, transmission, and braking systems work.

4.2 SWING

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit. Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform.

Swing introduced a mechanism that allowed the look and feel of every component in an application to be altered without making substantial changes to the application code. The introduction of support for a pluggable look and feel allows Swing components to emulate the appearance of native components while still retaining the benefits of platform independence. This feature also makes it easy to make an application written in Swing look very different from native programs if desired. originally distributed as a separately downloadable library, Swing has been included as part of the Java Standard Edition since release 1.2. The Swing classes and components are contained in the javax.swing package hierarchy[8].

Architecture

Swing is a platform-independent, Model-View-Controller GUI framework for Java. It follows a single-threaded programming model, and possesses the following traits:

Platform independence

Swing is platform independent both in terms of its expression (Java) and its implementation (non-native universal rendering of widgets).

Extensibility

Swing is a highly partitioned architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations. In general, Swing users can extend the framework by extending existing (framework) classes and/or providing alternative implementations of core components.

Component-oriented

Swing is a component-based framework. The distinction between objects and components is a fairly subtle point: concisely, a component is a well-behaved object with a known/specified characteristic pattern of behaviour. Swing objects asynchronously fire events, have "bound" properties, and respond to a well-known set of commands (specific to the component.) Specifically, Swing components are Java Beans components, compliant with the Java Beans Component Architecture specifications.

Customizable

Given the programmatic rendering model of the Swing framework, fine control over the details of rendering of a component is possible in Swing. As a general pattern, the visual representation of a Swing component is a composition of a standard set of elements, such as a "border", "inset", decorations, etc. Typically, users will programmatically customize a standard Swing component (such as a JTable) by assigning specific Borders, Colors, Backgrounds, opacities, etc., as the properties of that component. The core component will then use these property (settings) to determine the appropriate renderers to use

in painting its various aspects. However, it is also completely possible to create unique GUI controls with highly customized visual representation.

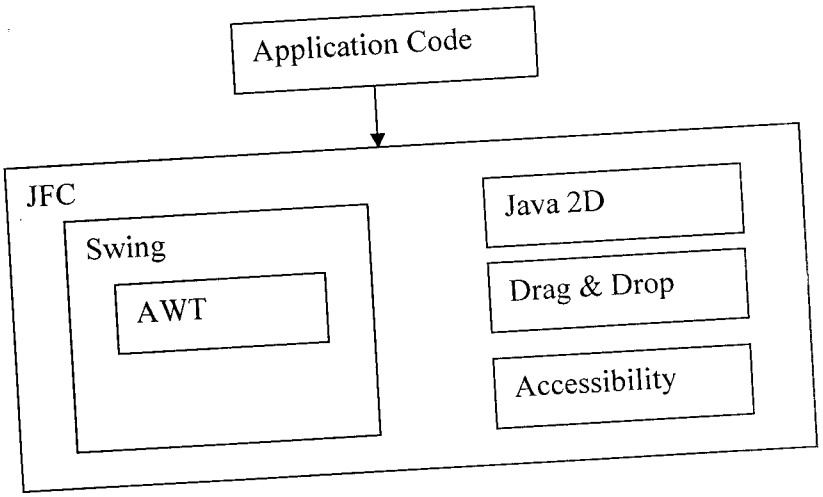
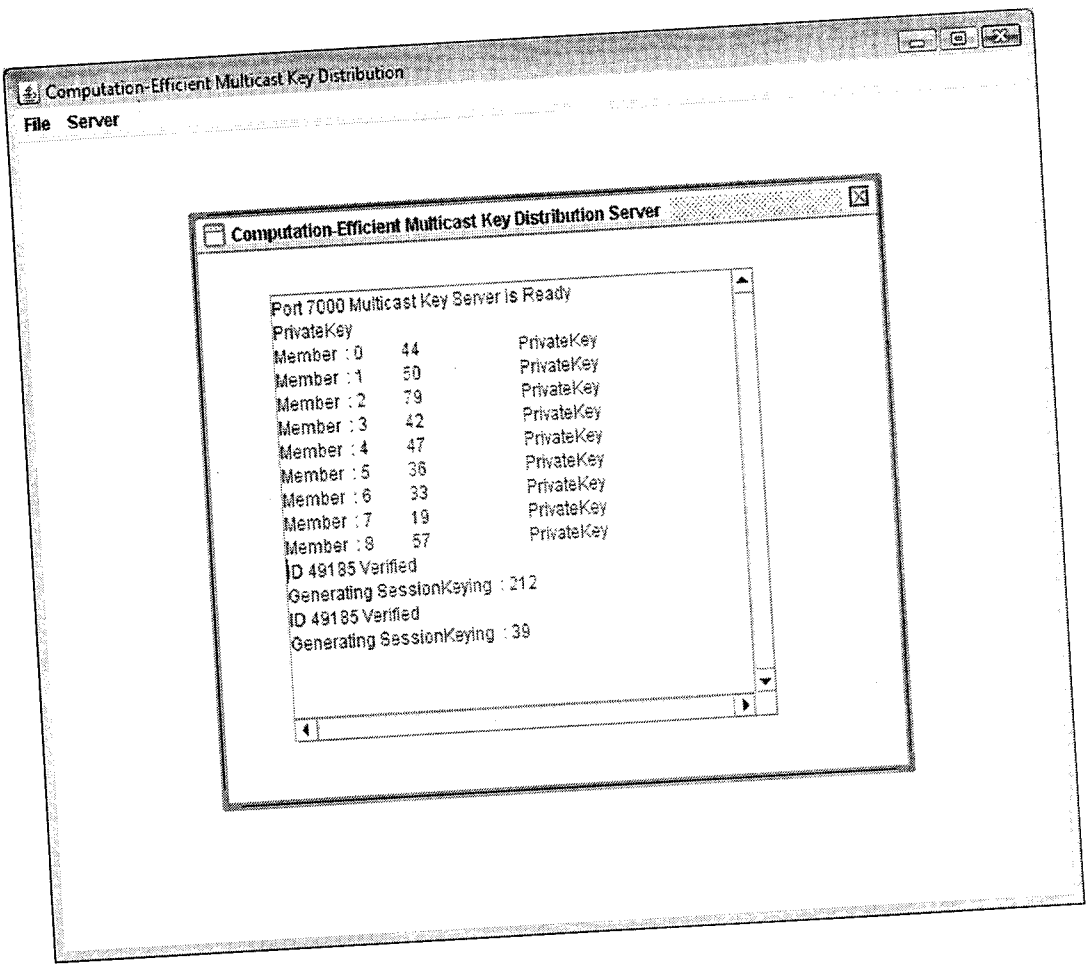


Figure 4.1 Swing class

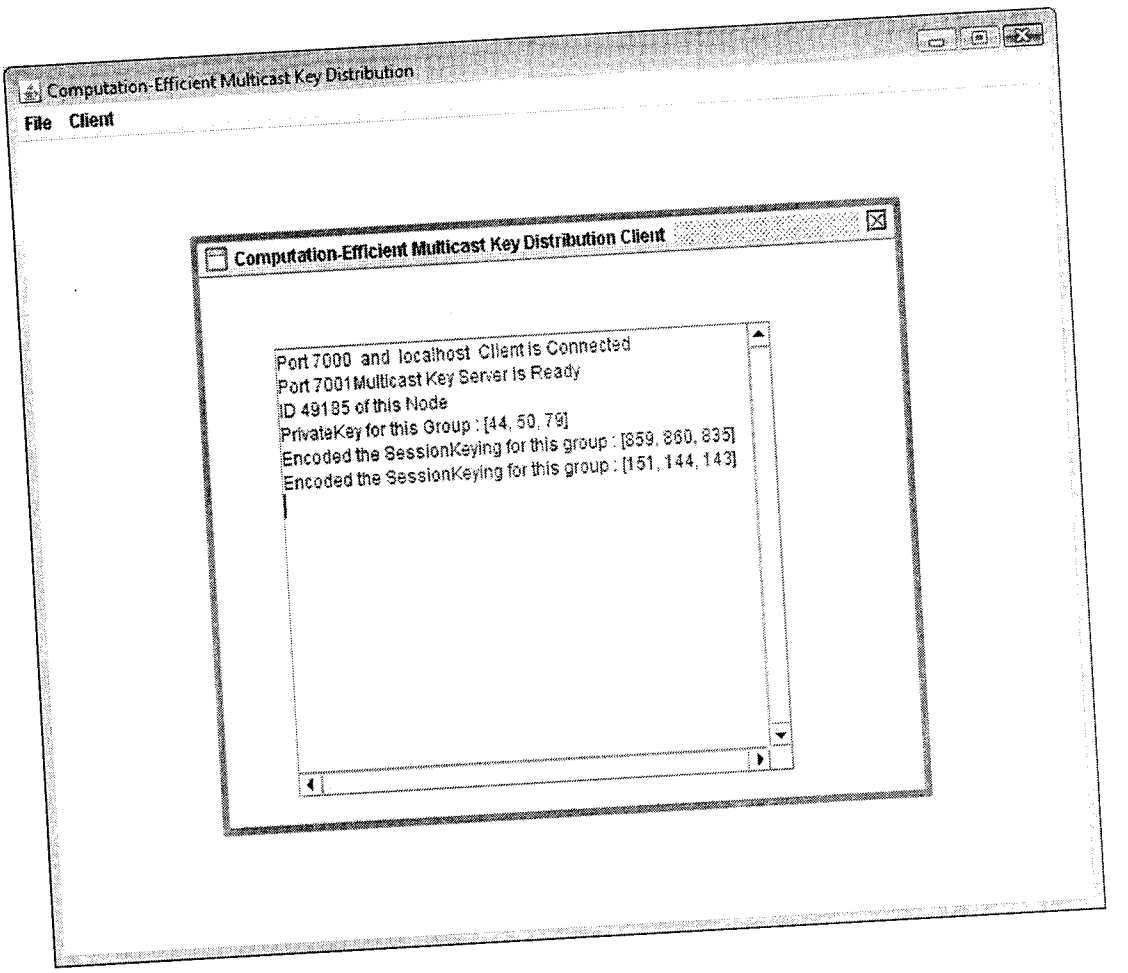
CHAPTER 5

EXPERIMENTAL RESULTS

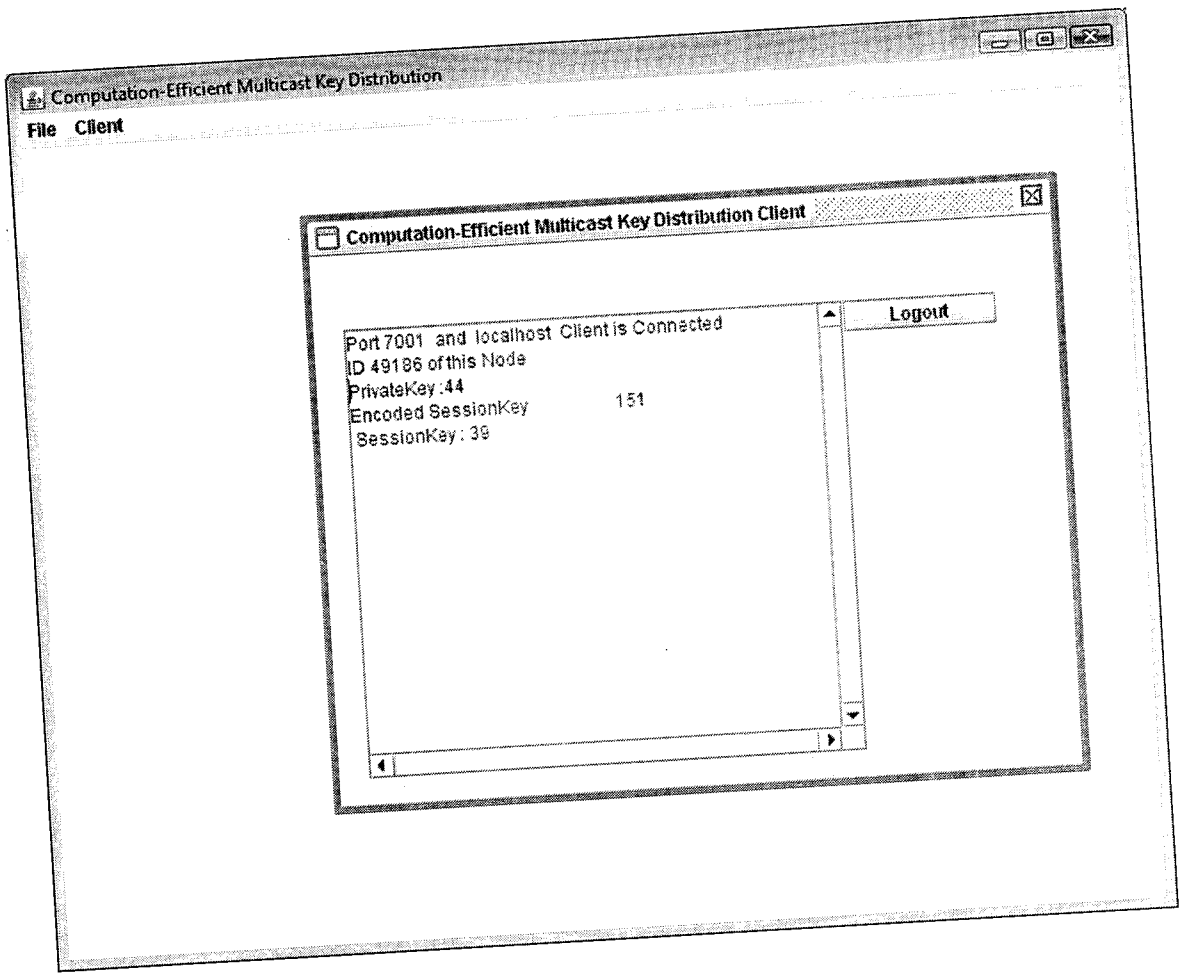
5.1 GROUP CONTROLLER



5.2 INTERMEDIATE



5.3 MEMBER


Computation-Efficient Multicast Key Distribution
File Client
Computation-Efficient Multicast Key Distribution Client
Port 7001 and localhost Client is Connected
ID 49186 of this Node
PrivateKey :44
Encoded SessionKey 151
SessionKey : 39
Logout

5.4 PERFORMANCE COMPARISON GRAPH

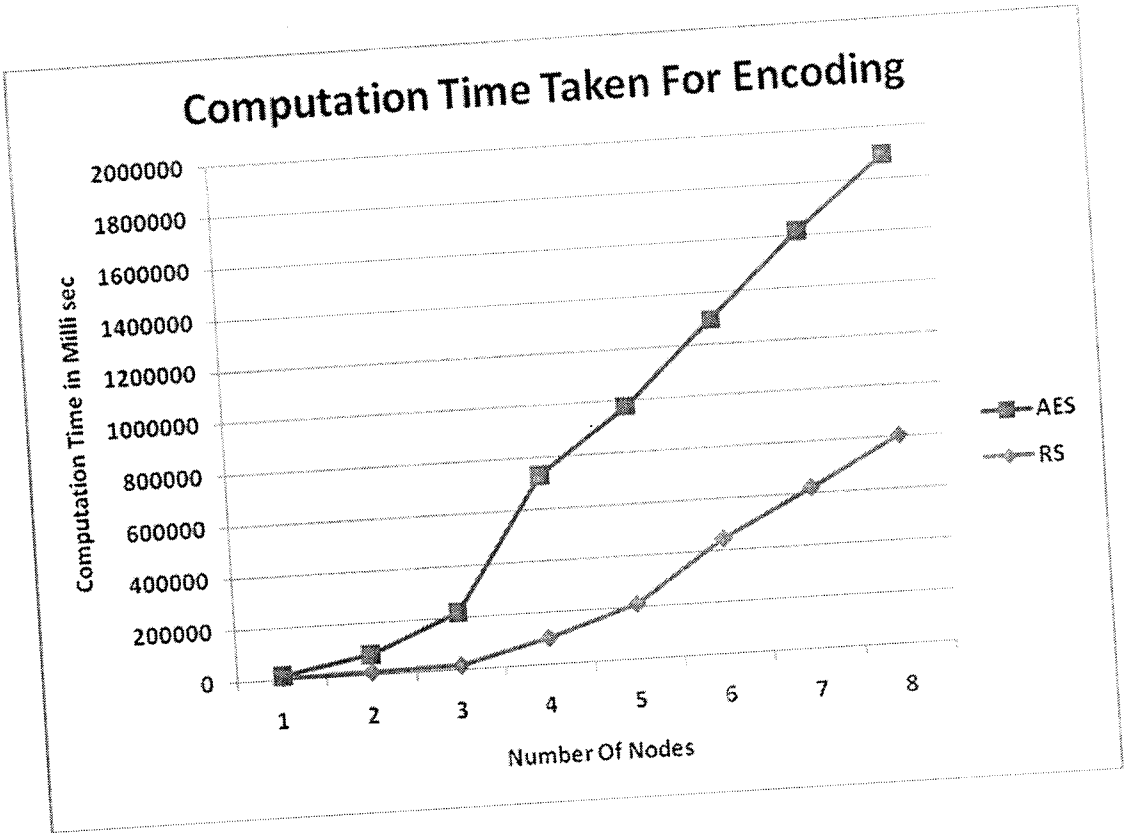
Reed Solomon code is a linear code and which is a class of MDS code, to distribute multicast key dynamically that drastically reduces the computation load of each group member compared to existing schemes employing traditional encryption algorithms. Comparison of computation time of existing system and proposed system are shown below

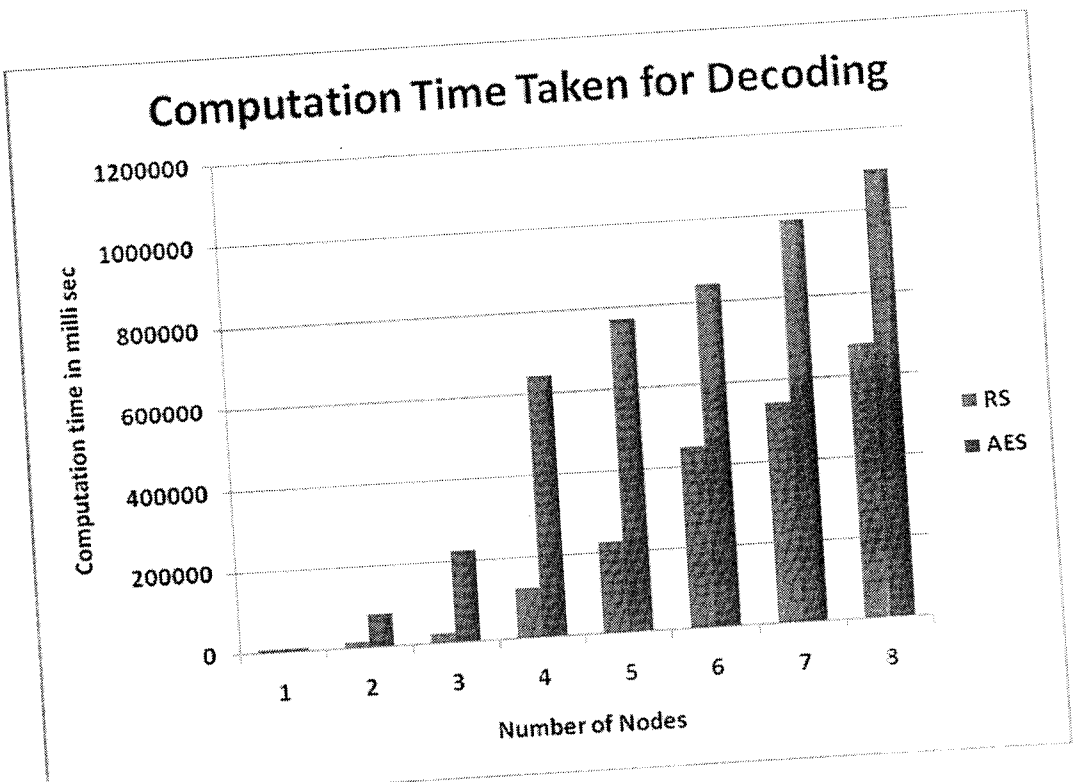
Computation time of RS	8361	13291	19032	102918	215582	447372	624458	804728
Computation time of AES	10373	67775	205441	633028	763823	848077	990872	1089211
No:of Nodes	1	2	3	4	5	6	7	8

Table 1. Computation time taken by RS and AES for encoding

Computation time of RS	9427	14378	24982	121829	226473	446773	548542	679355
Computation time of AES	10458	81875	224566	647542	772338	846173	988345	1099468
No:of Nodes	1	2	3	4	5	6	7	8

Table 2. Computation time taken by RS and AES for decoding


Computation Time Taken For Encoding
Computation Time in Milli sec
2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
1
2
3
4
5
6
7
8
Number Of Nodes
AES
RS


Computation Time Taken for Decoding
Computation time in milli sec
1200000
1000000
800000
600000
400000
200000
0
1
2
3
4
5
6
7
8
Number of Nodes
RS
AES

CHAPTER 6

CONCLUSION AND FUTURE OUTLOOK

6.1 CONCLUSION

I have implemented key distribution using Reed Solomon code. Key Tree scheme is for distributing key among the group members. Reed Solomon code is the widely used MDS code. There is a Group Controller to distribute the session keys among the group members. Forward secrecy and backward secrecy is maintained by dynamically changing the session key. Computation time of both Reed Solomon and AES are observed and plotted performance comparison graph. Key distribution using Reed Solomon code is less time consuming as compared to Advanced Encryption Standard algorithm.

6.2 FUTURE OUTLOOK

Now the group controller and group members are considered as fixed, that is not moving in the network area. In future the movements of the nodes are considered and keys are distributed by using reed solomon code. Also will consider a selection algorithm for the group controller.

APPENDIX 1

Root program is given below:

```
import java.io.*;
import java.net.*;
import javax.swing.*;
import java.util.*;

public class Server extends JInternalFrame
{
        ServerSocket ss;
        Socket s;
        String port=null;
        int mport;
        static int count1=0;
        Vector cv;
        //Swing this class
        public static JTextArea txtareamsg;
        JScrollPane scrollpan1;
        JPanel pan1;
         //creating for hiding mnu
        JMenuItem mnuitexit,mnuitport,mnuitstartserver;
        //creating key
        Reedencode  r1;

        int prikey[],result[];
        public Server(String[] stIpPort,Object mnuobj[])
        {
                super("Computation-Efficient Multicast Key Distribution
    Server",true,true,false,false);
                port=stIpPort[0];
                mport=Integer.parseInt(stIpPort[0]);

                mnuitexit=(JMenuItem)mnuobj[0];
                mnuitport=(JMenuItem)mnuobj[1];
                mnuitstartserver=(JMenuItem)mnuobj[2];
                mnuitstartserver.setEnabled(false);
                cv=new Vector();
```

```
            txtareamsg=new JTextArea();
            scrollpan1=new
JScrollPane(txtareamsg,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrol
lPane.HORIZONTAL_SCROLLBAR_ALWAYS);
            pan1.setLayout(null);
            pan1.add(scrollpan1);
            scrollpan1.setBounds(50,30,350,300);
            setSize(500,400);
            getContentPane().add(pan1);
            setVisible(true);
            new ServerconAccepted(port);
            prikey=new int[9];
            for(int j=0;j<9;j++)
            {
                  Random rn=new Random();
                  int a2=rn.nextInt(100);
                  prikey[j]=a2;
            }
            txtareamsg.append("PrivateKey\n");
            for(int i=0;i<prikey.length;i++)
            {
                  txtareamsg.append("Member :
"+i+""+prikey[i]+"\tPrivateKey\n");
            }

      }

      class ServerconAccepted extends Thread
      {
            int portval=0;
            public ServerconAccepted(String port)
            {
                  portval=Integer.parseInt(port);
                  try
                  {
                        ss=new ServerSocket(portval);
                        txtareamsg.append("Port "+portval+" Multicast Key
Server is Ready\n");
                  }catch (Exception e){e.printStackTrace();}
                  this.start();
```

```
    }

    public void run()
    {
        try
        {
            while(true)
            {
                s=ss.accept();
                Serverfirst cc=new Serverfirst(s);
                cv.addElement(cc);
            }
        }catch (Exception e){e.printStackTrace();}
    }
}

class Serverfirst extends Thread
{
    PrintStream out;
    DataInputStream in;
    String str1=null,str2=null,cp=null;
    Random rn;
    public Serverfirst(Socket b)
    {
        try
        {
            in=new DataInputStream(b.getInputStream());
            out=new PrintStream(b.getOutputStream());
            rn=new Random();
        }catch(Exception e){e.printStackTrace();}
        this.start();
    }
    public int hashencode(int a,int b)
    {
        int c = a << 2;
        int d = b >> 2;
        int e=c^d;
        txtareamsg.append("HashCode SessionKeying\t"+e+"\n");
        return e;
    }
```

```
public void run()
{
        try
        {
                while(true)
                {
                        str1=in.readLine();
if(str1.equals("connected")
{
        str2=in.readLine();
        cp=str2;
        txtareamsg.append("ID "+str2+" Verified\n");
        r1= new Reedencode();
        result=r1.Rencode();
        int temp=rn.nextInt(255);
        txtareamsg.append("Generating SessionKeying  : "+temp+"\n");
        for(int k1=0;k1<cv.size();k1++)
                {
                Serverfirst obj=(Serverfirst)cv.elementAt(k1);
                if(obj.cp.equals(str2))
                {
                if(k1 == 0)
                 {
                   obj.out.println("newconnected");
                   obj.out.println(prikey[0]);
                   obj.out.println(prikey[1]);
                    obj.out.println(prikey[2]);
                 }
                 if(k1 == 1)
                        {
                        obj.out.println("newconnected");
                        obj.out.println(prikey[3]);
                        obj.out.println(prikey[4]);
                        obj.out.println(prikey[5]);
                  }
                  if(k1 == 2)
                  {

obj.out.println("newconnected");
```

```
obj.out.println(prikey[6]);

obj.out.println(prikey[7]);

obj.out.println(prikey[8]);                                    }
                                                          }
                                                    }


 for(int k1=0;k1<cv.size();k1++)
                                                    {
 Serverfirst obj=(Serverfirst)cv.elementAt(k1);
        if(k1 == 0)
         {
        obj.out.println(str1);

  obj.out.println(result.length);
  for(int i=0;i<result.length;i++)
        {

   obj.out.println(result[i]);                                  }


    obj.out.println(hashencode(temp,prikey[0]));

    obj.out.println(hashencode(temp,prikey[1]));

    obj.out.println(hashencode(temp,prikey[2]));
                                                          }
                                                          if(k1 == 1)
                                                          {
                                                                obj.out.println(str1);


     obj.out.println(result.length);
           for(int i=0;i<result.length;i++)
           {

     obj.out.println(result[i]);
```

```
        }
obj.out.println(hashencode(temp,prikey[3]));
obj.out.println(hashencode(temp,prikey[4]));
obj.out.println(hashencode(temp,prikey[5]));
                                                }
                if(k1 == 2)
        {
obj.out.println(str1);
obj.out.println(result.length);
for(int i=0;i<result.length;i++)
                                                {
obj.out.println(result[i]);
                                                }
obj.out.println(hashencode(temp,prikey[6]));
obj.out.println(hashencode(temp,prikey[7]));
obj.out.println(hashencode(temp,prikey[8]));
                                                }
                                        }
                                }
                                if(str1.equals("connected1"))
                                {
                                        str2=in.readLine();
                                        cp=str2;
                                txtareamsg.append("ID "+str2+" Verified\n");
                                        r1= new Reedencode();
                                        result=r1.Rencode();
                                        int temp=rn.nextInt(255);
                                        txtareamsg.append("Generating
SessionKeying  : "+temp+"\n");
```

```
for(int k1=0;k1<cv.size();k1++)
        {
        Serverfirst obj=(Serverfirst)cv.elementAt(k1);
if(obj.cp.equals(str2))
                {
                if(k1 == 0)
                {

 obj.out.println("newconnected1");
 obj.out.println(prikey[0]);
 obj.out.println(prikey[1]);
 obj.out.println(prikey[2]);
                                }
if(k1 == 1)
                {

  obj.out.println("newconnected1");
  obj.out.println(prikey[3]);
  obj.out.println(prikey[4]);
  obj.out.println(prikey[5]);

          }
          if(k1 == 2)
                {
          obj.out.println("newconnected1");
          obj.out.println(prikey[6]);
    obj.out.println(prikey[7]);
    obj.out.println(prikey[8]);
                                                }
                                        }
                                }

 for(int k1=0;k1<cv.size();k1++)
        {
        Serverfirst obj=(Serverfirst)cv.elementAt(k1);
 if(k1 == 0)
                {
                obj.out.println(str1);

  obj.out.println(result.length);
  for(int i=0;i<result.length;i++)
```

```
{
obj.out.println(result[i]);
}
obj.out.println(hashencode(temp,prikey[0]));
obj.out.println(hashencode(temp,prikey[1]));
obj.out.println(hashencode(temp,prikey[2]));
}
if(k1 == 1)
        {
                obj.out.println(str1);
   obj.out.println(result.length);
        for(int i=0;i<result.length;i++)
        {
    obj.out.println(result[i]);
        }
    obj.out.println(hashencode(temp,prikey[3]));
    obj.out.println(hashencode(temp,prikey[4]));
    obj.out.println(hashencode(temp,prikey[5]));
                }
    if(k1 == 2)
          {
                obj.out.println(str1);
                obj.out.println(result.length);
for(int i=0;i<result.length;i++)
      {
      obj.out.println(result[i]);
      }
      obj.out.println(hashencode(temp,prikey[6]));
      obj.out.println(hashencode(temp,prikey[7]));
      obj.out.println(hashencode(temp,prikey[8]));
                                                }
                                            }
                                        }}
                        }catch(Exception e){}
                }}}
```

REFERENCES

[1] Lihao Xu, Cheng Huang, "Computation Efficient Multicast Key Distribution ", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 19, 577-586,MAY 2008.

[2] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication", ACM Computing Surveys, vol. 35,no. 3, pp. 309-329, 2003.

[3] Gerardo del Valle and Roberto Gomez Cardenas, "Overview the Key Management in Ad Hoc Networks", Springer Berlin /Heidelberg, pp.397-406 , VOL.3563,2005.

[4] X. S. Li, Y. R. Yang, M. G. Gouda and S. S. Lam, "Batch Rekeying for Secure Group Communications", Proc. of the 10th International World Wide Web Conference, Hong Kong, China, May 2001.

[5] J. S. Plank and Y. Ding "Note: Correction to the 1997 Tutorial on Reed Solomon Coding" Software, Practice & Experience, vol. 35, no. 2, pp. 189–194, Feb. 2005.

[6] http://computer.howstuffworks.com/home-network2.htm

[7] J. S. Plank and L. Xu, "Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications", Proc. of The 5th IEEE International Symposium on Network Computing and Applications (IEEE NCA06), Cambrid ge, MA, July, 2006.

[8] James Gosling, Bill Joy," Java Language Specification", Sun

[9] R.W. Hamming, "Error detecting and error correcting codes", Bell System Tech.J.,26(1950),pp. 147-160.

[10] H. Harney, C. Muckenhirn,"Group Key Management Protocol (GKMP) Architecture", SPARTA,July 1997,pp. 1-23.

[11] http://en.wikipedia.org/wiki/Network_security

[12] Vincent Rijmen, Joan Daemen,"Advanced Encryption Standard Algorithm", Springer Berlin /Heidelberg, pp.597-806 , VOL.3563,1998.