

MMKM: A Novel Multicast MBMS Key Management Architecture



Acc. No. (FMS) T-1410



DATA ENTERED

Developed by:

Mamoona Asghar
154-CS/MS/03
Moizza Sharmin
158-CS/MS/03

Supervised by:

Prof. Dr. Khalid Rashid
Dr. Tauseef ur Rehman



Department of Computer Science
Faculty of Applied Sciences
International Islamic University, Islamabad
(2006)

Handwritten notes and numbers: 2, 38, 223, 255

MS
006.7
MAM

DATA ENTERED

of
16/02/2022

- 1- Multimedia systems
- 2- computer networks
- 3- SRTP
- 4- MIKEY
- 5- computer software



MMKM: A Novel Multicast MBMS Key Management Architecture



Developed by:

Mamoona Asghar

154-CS/MS/03

Moizza Sharmin

158-CS/MS/03

Supervised by:

Prof. Dr. Khalid Rashid

Dr. Tauseef ur Rehman

Department of Computer Science
Faculty of Applied Sciences
International Islamic University, Islamabad
(2006)



**In the name of ALMIGHTY ALLAH,
The most Beneficent, the
most Merciful**

Department of Computer Science
International Islamic University Islamabad

Final Approval

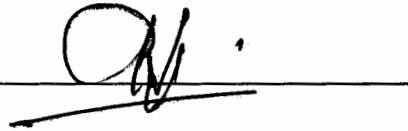
24-8-2006

It is certified, that, we have read the project report submitted by Mamoona Asghar, Registration number 154-CS/MS/03, and Moizza Sharmin, Registration number 158-CS/MS/03, and it is our judgment that this project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the Master of Science Degree in Computer Science.

Committee

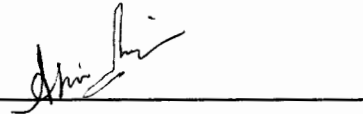
External Examiner

Dr. Abdus Sattar,
Former DG,
Pakistan Computer Bureau,
Islamabad, Pakistan



Internal Examiner

Mr. Asim Munir,
Assistant Professor,
Department of Computer Science
Faculty of Applied Sciences,
International Islamic University,
Islamabad, Pakistan.



Supervisor

Prof. Dr. Khalid Rashid,
Dean,
Faculty of Applied Sciences,
International Islamic University,
Islamabad, Pakistan.



Dedication

**Dedicated to Holy Prophet Muhammad (SAW)
and to our Parents**

After Allah Almighty, we are completely indebted to our parents for this dissertation, whose affection has always been a source of encouragement for us, and whose prayers always turn out to be a key to our success.

A Thesis submitted to the
Department Of Computer Science,
Faculty of Applied Sciences,
International Islamic University, Islamabad
as a partial fulfillment of the requirement
for the award of the degree of
MS in Computer Sciences

Declaration

We hereby declare that we have developed MMKM application and the accompanied report entirely on the basis of our efforts made under the sincere guidance of our teachers and supervisor. No portion of the work presented in this application has been submitted in support of any application for any other degree or qualification of this or any other university or institute. If it found to be copied from any other source, we shall bear the consequences.

Mamoona Asghar

154-CS/MS/03

Moizza Sharmin

158-CS/MS/03

Acknowledgment

We bestow all praises, acclamation and appreciation to *Almighty Allah*, The Most Merciful and Compassionate, The most Gracious and Beneficent, Whose bounteous blessings enabled us to pursue and perceive higher ideals of life. All praises for His *Holy Prophet Muhammad (SAW)* who enabled us to recognize our Lord and Creator and brought to us the real source of knowledge from Allah, *The Qur'an*, and who is a role model for us in every aspect of life.

We would consider it a proud privileged to express our gratitude and deep sense of obligation to our reverend supervisors *Prof. Dr. Khalid Rashid and Dr. Tauseef ur Rehman* for their dexterous guidance and kind behavior during the project.

It will not be out of place to express our profound admiration for our Networks teacher **Mr. Shiraz Baig** for his dedication, inspiring attitude, untiring help and kind behavior throughout the project efforts and presentation of this manuscript. He did a lot of effort for our success.

Finally we must mention that it was mainly due to our family's moral support and financial help during my entire academic career that enabled us to complete our work dedicatedly. We once again would like to admit that we owe all our achievements to our most loving parents, who mean most to us, for their prayers are more precious than any treasure on earth.

Mamoona Asghar

Moizza Sharmin

Project in Brief

Project Title MMKM: A Novel Multicast MBMS Key Management Architecture

Objective To develop an environment that performs secure multimedia audio streaming to multicast receivers, which prevent data hacking and eavesdropping security threat

Undertaken by **Mamoona Asghar**
154-CS/MS/03

Moizza Sharmin
158-CS/MS/03

Supervised by **Prof. Dr. Khalid Rashid**
Dean Faculty of Applied Sciences
International Islamic University Islamabad.

Starting Date January 2005

Completion Date January 2006

Tool Used C language

Operating System Linux Distribution - Red Hat Linux release 9 (Shrike)
Kernel 2.4.20-8

System Used Intel Pentium IV

Abstract

This thesis presents a design of key management architecture for secure multimedia audio streaming to multicast receivers. It highlights the security issues involved, implementation of MIKEY, integration with SRTP and then using it for transmitting live audio data in Multicast Environment. The effort has been more on Security rather than on data compression. MIKEY is used for key management, key distribution, and identification of party through digital signatures. SRTP is responsible for encryption, authentication of packets and finally transmission and playing of live audio data. A number of practical issues were tackled, like request of song from the recipients, password authentication, the actual audio data and separate transmissions of audio file characteristics like bit depth, bitrate, channels etc. Multithreaded application is used for this purpose. Diffie-Hellman technique is used for MIKEY, AES and HMAC for encryption and authentication.

Abbreviations

AES	Advance Encryption Standard
ASRVR	Audio Server
BM-SC	Broadcast Multicast Service Centre
CLNT	Client
CM	Counter Mode
CS	Crypto Master
CSB	Crypto Master Bundle
DH	Diffie-Hellman
DoS	Denial of Service
HDR	MIKEY common header payload
HMAC	Keyed Hash-Based MAC Function
IP	Internet Protocol
IPv4	Internet Protocol version 4, e.g. 192.168.1.10
IPv6	Internet Protocol version 6
MAC	Message Authentication Code
MBMS	Multimedia Broadcast Multicast Services
ME	Mobile Equipment
MIKEY	Multimedia Internet KEYing
MSK	Master Session Key
Nb	AES 32-bit words (number of columns) in the State
Nk	AES key length i.e. $N_k = 4, 6, \text{ or } 8$
PK	Public-Key
PRF	Pseudo Random Function (like Hash)
PSK	Pre-Shared key
RFC	Request for Comments
RR	Receiver Report
RSRVR	Registration Server
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol

SDES	Source Description Report
SIP	Session Initiation Protocol
SRTP	Secure RTP
SRTCP	Secure Real Time Control Protocol
SR	Sender Report
TCP	Transmission Control Protocol
TEK	Traffic-encrypting key
TGK	TEK Generation Key
UDP	User Datagram Protocol
UE	User Equipment
UICC	Universal IC card

Table of Contents

Ch No.	Contents	Page No.
1.	Introduction	01
1.1	Unicast/Multicast communication	01
1.2	Multimedia Broadcast/Multicast Service (MBMS)	02
1.3	Literature Review	03
1.4	The Project.....	05
1.4.1	Problem Definition	05
1.4.2	Objective.....	06
1.4.3	Scope	07
2.	Security Protocols & Algorithms.....	08
2.1	SRTP.....	08
2.1.1	Security Goals.....	08
2.1.2	Features.....	09
2.1.3	Secure RTP (SRTP) Packet.....	09
2.1.4	Secure RTCP (SRTCP) Packet.....	10
2.1.5	SRTP Key Derivation Algorithm.....	11
2.1.6	SRTCP Key Derivation	12
2.1.7	Multicast (one sender).....	12
2.2	MIKEY.....	13
2.2.1	Design Goals.....	14
2.2.2	System Overview.....	14
2.2.3	Basic Key Transport and Exchange Methods.....	15
2.2.4	Generating keys from TGK.....	16
2.2.5	Key data transport encryption.....	17
2.2.6	MAC and Verification Message function.....	17
2.3	AES Algorithm.....	17
2.3.1	AES Algorithm Specification.....	18
2.4	Diffie-Hellman Algorithm.....	18
2.5	HMAC, the Keyed Hash-Based MAC Function.....	19

2.5.1 HMAC Algorithm.....	19
3. Solution.....	20
3.1 Proposed Architecture.....	20
3.2 Design.....	21
3.2.1 View Point Analysis.....	21
3.2.2 Client-Server Emphasized	23
3.2.3 Control Modeling.....	26
3.3 Modular Decomposition.....	27
3.3.1 Registration Server.....	27
3.3.2 Audio Server.....	28
3.3.3 Client.....	28
4. Implementation.....	29
4.1 Sending Audio thru SRTP over Network.....	29
4.1.1 UDP.....	30
4.2 Multithreading.....	31
4.3 Implementation Stages.....	31
4.4 Application modules.....	32
4.4.1 Registration Server.....	32
4.4.2 Audio Server	33
4.4.3 Client	33
4.5 Key Management Mechanism.....	34
4.5.1 Master key (TGK).....	35
4.5.2 Diffie-Hellman Key.....	35
4.5.3 SRTP Keys.....	35
4.5.4 SRTCP Keys.....	36
4.6 Pseudo codes.....	37
4.7 Make file.....	45

5. Results and Discussions	47
5.1 Timings for Encryption.....	47
5.2 Conclusions.....	48
5.3 Access to Threads	52
5.4 Future Enhancements.....	52

References & Bibliography

Appendix A - Glossary	A-1
Appendix B - Data Flow Diagrams	B-1
Appendix C - Linux/Soundcard.h	C-1
Appendix D - User Manual	D-1
Appendix E – Research Paper	E-1

CHAPTER 1
INTRODUCTION

1. Introduction

Wireless market is growing rapidly, and reaching new customers and services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation and fast/flexible service creation WAP Forum defines a set of protocols in transport, security, transaction, session and application layers. Information security is becoming a high priority for businesses around the world for wireless applications. With the dramatic increase in electronic communications and electronic commerce, there has been a corresponding increase in the malicious compromise of that information. Media access is very simple in wireless networks compared with wired networks; because of this, their security concerns are immensely important.

All of us are quite familiar with the term audio conferencing, where we can hear the live songs from hundreds of miles away in some remote corner of the world. The two types of real time contents in this stream of data are real-time audio and real-time video. In the last few years use of interactive multimedia, both media-on-demand and multimedia conferencing systems between users, has grown in the TCP/IP networks and significant developments are taking place in the field of audio-visual communications. It has been observed that transferring real-time audio on the Internet requires powerful techniques to achieve good quality audio streams. The real-time audio has gradually become more and more important as a communication media in Local Area Networks, and for the last two years also on the Internet. The security of data transmission on Internet is a big concern now-a-days. So the emphasize of research is security of multimedia data on network. SRTP and MIKEY Protocols are used for the security and key management purpose.

1.1 Unicast/Multicast communication

Data and network security is recognized as vital to the design of modern communication systems. Much work has been done in developing security solutions and mechanisms for unicast communication. For multicast communication, the issues and problems are substantially more complex, since multicast involves multiple participants that may be spread over many networks and may join and/or leave dynamically. The problem of multicast security has not received much attention. Proposals for securing multicast communication do not address the unique requirements arising from the multicast group communication model. Examples of applications based on internet multicast include transmission of corporate data to employees, communication of stock quotes to brokers, streaming audio and video contents, online conferencing, collaborative work, caching and replication of databases and websites in multiple locations. A secure multicast Research group (SMuG) was formed (in 1998) in the IRTF to standards for multicast security. The basic Internet protocol, in general, and Internet Multicast, in particular, suffers from lack of

security. Furthermore, due to the nature of the basic Internet multicast protocol, receiver-participants can join and leave multicast session at will without notifying the multicast sender or other participants. While much effort has been made in developing security solutions and mechanisms for point to point, or unicast communication, minimum attention has been paid to security issues in multicast communications.

1.2 Multimedia Broadcast/Multicast Service (MBMS)

The broadcast/multicast service provides an ability to transmit the same information For example, text, multimedia (For example, audio, video) and streaming media stream to multiple users simultaneously. The motivation for this service is to achieve the most efficient use of air interface and network resources when sending the same information to multiple users. For the deployment of multicast services many factors need immense attention, For example, questions like how a system admit multicast services, how radio access network selects between the use of point-to-point and point-to multipoint radio bearers to deliver MBMS (Multimedia Broadcast/Multicast Services) services, how the network alert MBMS subscribed users on the availability of MBMS streams etc would arise. Mobility management of MBMS users needs to be considered too. Since users are expected to charge for such multicast issues, authentication and security issues related to MBMS needs to be worked out. There has recently been work to define a security protocol for the protection of real-time applications running on RTP named SRTP. However, a security protocol needs a key management solution to exchange keys and related security parameters. There are some fundamental properties that a key management scheme has to fulfill to serve streaming and real-time applications (such as unicast and multicast), particularly in heterogeneous (mix of wired and wireless) networks.

Some of the common security threats and types of attacks on unicast and multicast communications are Eavesdropping, Impersonation, Data Manipulation, Denial of submission, Denial of Receipt, Repudiation, Replay Attacks, Denial of services, and theft of services or content. The additional security threats and attacks for multicast communication are Uncontrolled Multicast Group Membership, Leakage of Security Stage, Security state Revocation, Security Management, Multi-Point Attacks, Multicast session Publicity.

This research work is a humble effort to develop an environment that performs secure multimedia (audio .wav files) streaming to multicast receivers, which avoid data hacking and eavesdropping security threats.

1.3 Literature Review

Fern Lavitt pointed out that the basic data communication protocols incorporate very few security features, if any. Typically, security is handled at the level of the system that uses the communication protocols. [1, 2] In document SA3#25 Ericsson presented contributions which proposed to adopt SRTP and MIKEY for MBMS. [3, 4] Ericsson discussed MIKEY and it was compared to two other multicast key management protocols, namely Group Domain of Interpretation (GDOI) and Group Security Association and Key Management Protocol (GSAKMP-light). [5, 6] It was found that MIKEY is the most efficient and best suitable for MBMS since it is using pre-shared keys with symmetric cryptography. So, MIKEY was proposed for MBMS.

SRTP was compared against IP sec and radio level multicast security [3]. The contribution proposed SRTP as security protocol for streaming applications for MBMS. However, document SA3 #25 concluded that there were too many open issues regarding security architecture and also pointed out additional contributions were requested to progress the technical specification. At that time TS was not mature enough for specific protocols to be included. In document #27 and #28 SA3 agreed on some important security architecture issues, namely the encryption of MBMS traffic shall be done between UE and BM-SD. BM-SC shall be the entity to generate and distribute traffic encryption key (TEK) to the UEs [7]. Today many issues are still to be specified, regarding user authentication, charging models (and related keying mechanisms) and interface functionality. Ericsson presented a discussion paper [3] on the status of MIKEY and SRTP [8, 9] in IETF in document SA3#28. These protocols are strong candidates for key management and security protocols for MBMS and got RFC status during 2003. In document SA3#29 Ericsson presented contribution and related pseudo CR, which proposed to adopt SRTP and MIKEY for MBMS [10]. This proposal was not accepted due to concerns that the key management protocol was not inline with the proposal discussed in 3GPP2. However since there is a lack of indication of what protocols to use it is difficult to make any detailed judgment of the whole system impacts based on [11]. It was also not clear what type of redundancy (if any) is required. For example, use of FEC (Forward Error Corrections) or similar techniques to mitigate the issue related with bit errors. This contribution describes that SRTP can be used as security protocol for streaming MBMS media. Ericsson pointed out the issue that should the Ad Hoc support the working hypothesis to include SRTP for protection of streaming services for MBMS. Ericsson volunteers received feedback by the Ad Hoc meeting to the SA3#30 document [10, 12].

J. Arkko et. al., describes a key management scheme that can be used for real-time applications (both for peer-to-peer communication and group communication). In particular, its use to support the Secure Real-time Transport Protocol is described in detail. Security

protocols for real-time multimedia applications have started to appear. This has brought forward the need for a key management solution to support these protocols [8].

Vesa Lehtovirta, Ericsson SA3 agreed that MBMS key management should be based on extensions to the MIKEY-protocol. MIKEY has been designed together with SRTP, which was proposed for the protection of MBMS streaming data [13].

Finally Vesa Lehtovirta, Ericsson SA3 confirmed that the SRTP is a security protocol that can be used for protecting MBMS streaming data. It is proposed that SRTP is chosen as security protocol for MBMS streaming data [14].

Ericsson in document S3#31 presents that MIKEY enables both ME and UICC based key management. This document and accompanying document [15, 16] describe how MIKEY is used in ME and UICC based key management. Ericsson has studied the issue and came to the conclusion that the problem is mitigated by providing a solution in key management level, i.e. in MIKEY (Multimedia Internet KEYing) protocol. [8, 17]

Ericsson and Nokia in document S3#33 presents a document which focuses on the case where all key handling is done in the ME. The MIKEY presents in document S3-040081 [18] introduced unnecessary deviation from original MIKEY. To minimize the impact on the MIKEY protocol, this document describes how the same information can be conveyed to the ME by means of the extension payload presented in MIKEY. The purpose of this payload is, as the name suggests, enabling extensions to the protocol in a controlled fashion. The document gives a slightly different and more clean (with respect to MIKEY) way of achieving the same task as described in [18]. It describes how to use MIKEY in the case all key handling is done in the ME. [19]

Ericsson and Nokia in document S3#33 [20] present a document which focuses on that MIKEY has been proposed earlier for MBMS key management. [20, 21, 22] This contribution discussed the suitability of MIKEY for MBMS key management. In document #32, SA3 decided to adopt a two-tiered keying mechanism for MBMS. Although the MIKEY developed in IETF but it does not support this, that's why MIKEY need enhancements to support two-tiered keying and meet other MBMS specific requirements. The enhancements to MIKEY should be specified in IETF, but due to the time constraints of Rel-6 the enhancements are specified in 3GPP and possibly later in IETF in the form of an RFD. The contribution has shown that MIKEY offers a complete solution for MBMS key management including migration from ME based solution to UICC solution. It is proposed that the enhanced MIKEY is chosen as key management protocol for MBMS. The detailed descriptions of MIKEY in MBMS are found in companion contribution [23].

1.4 The Project

The Project presents the design of key management architecture for secure multimedia audio streaming to multicast receivers. It highlights the security issues involved, implementation of MIKEY, integration with SRTP and then using it for transmitting live audio data in Multicast Environment. The emphasis is not on compression of audio data but on its security. First MIKEY is used for key distribution taking care of identification of party through digital signatures, then encryption with AES and authentication of SRTP packets, finally transmission of live audio data, receiving and then playing it. A number of practical issues are tackled, like request of audio file from the recipients, password authentication, separate transmissions of audio file characteristics like bit depth, channels and bit rate etc. and the actual audio data. Multithreaded application is used for this purpose. Two servers are used, Registration server for user registration and key management, while audio server for handling the audio transmissions. Diffie-Hellman technique is used for MIKEY.

1.4.1 Problem Definition

The problematic scenarios of secure multicast transmission are briefly mentioned below, on the basis of which we started our research.

- i. In addition to the general threats to security, multicast communication is subject to some additional security threats and types of attacks.
- ii. IP Multicast protocols provide no means to specify, control, or limit the membership of a multicast session group.
- iii. In multicast environment, the security state of a multicast session may be shared among multiple participants, thereby increasing the risk of security state leakage.
- iv. For unicast point-to-point communication, satisfactory security solutions for the threats listed in section 1.2 have been developed. There is no simple mechanism to revoke the validity of a security state and to notify all multicast participants of the change. This may be necessary in order to restrict access of a multicast group member who has left the group.
- v. Multicast Protocol does not provide mechanisms for managing the security attributes of multicast group members. In particular, the problem of key distribution and re-keying group are of major concern. Another significant problem is the lack of synchronization among the group members.

- vi. In multicast, it is easier for an attacker to pose as one of many users, or to attack at several points in the network at the same time, thereby increasing the vulnerability of the system.
- vii. Multicast sessions are usually well advertised thereby leaking some data in advance and making it easier for an attacker to target an attack.
- viii. In an attack on a multicast environment, a large number of users may be affected, and the scope of the damage usually unknown.
- ix. It is doubtful to recommend that whether a key deletion function can be useful or not in MBMS.
- x. Multicast is inherently a receiver-based concept. The sender is not consulted about the addition of a multicast receiver. Receivers can join and/or leave a particular multicast session, whether or not it is currently active, at will. [24] There is no record of receivers during the session.

1.4.2 Objectives

- i. Implementation in GCC under Linux environment.
- ii. Installation of Sound card at the Client through which audio file will be played. Installation of TCP/IP based network between two computers, namely server and the client.
- iii. SRTP/SRTCP packetization, encryption of packets at Server side and decryption at Client side and report data at sender and receiver respectively. The SRTP packets contain encrypted audio data. SRTCP packets contain transmission statistics.
- iv. Transmission of SRTP/SRTCP packets over the network based on UDP Linux sockets.
- v. Display of SRTP and SRTCP reports at server and client ends, showing statistics of the server and client itself and statistics received in real time from both sides.
- vi. Multimedia Security will be the major concern rather than compression of data.
- vii. Diffie-Hellman with MIKEY Protocol will be used for Key distribution and management issues.
- viii. Playing of audio streams at client end.

1.4.3 Scope

- i. The system will be Multicast i.e. Server-Multiple Clients Model where there is a single sender and multiple receivers.
- ii. Implementation is half duplex i.e. only in one direction, Server can only send data and clients can only receive data. The transmissions are Synchronous and at every time data will move from sender.
- iii. Two Servers are used, which will run on the same machine.
- iv. The work deals with transmission of audio stream only and not with the video capture and transmission. Thus video issues are not resolved in it.
- v. Diffie-Hellman with MIKEY is used for the key distribution and management issues.
- vi. The system is implemented only on LAN. However it can be used on the Internet also.
- vii. Since the drivers of the sound card are not very accurate therefore sound quality may be suffer little bit.
- viii. The data will be received from one source only therefore, for the implementation of SRTP, only one synchronization source (SSRC) will be used. This will be the Server itself.
- ix. The architecture used will be for Intel PC. (Not aiming at Macintosh and Alpha etc).
- x. IPv4 and not IPv6 will be used.

CHAPTER 2
SECURITY PROTOCOLS &
ALGORITHMS

2. Security Protocols & Algorithms

The security protocols provide confidentiality, message authentication, and replay protection to the network traffic and control the traffic to protect actual data from security threats. The project is based on security concern of multicast multimedia transmission. To provide security to multimedia data, we need security protocols. The Security protocols define a set of default cryptographic algorithms to protect the data traffic. The cryptographic algorithms use different keys to encrypt and decrypt the actual data for its protection. For the sake of keys generation and distribution, security protocols need appropriate key management algorithms. Basic Network security protocols and their appropriate algorithms used in this project are:

2.1 SRTP

The Secure Real-time Transport Protocol (SRTP), a profile of the Real-time Transport Protocol (RTP), which can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the control traffic for RTP, RTCP (the Real-time Transport Control Protocol). [25]

SRTP provides a framework for encryption and message authentication of RTP and RTCP streams. SRTP defines a set of default cryptographic transforms. With appropriate key management, SRTP is secure for unicast and multicast RTP applications. SRTP can achieve high throughput and low packet expansion. SRTP proves to be a suitable protection for heterogeneous environments (mix of wired and wireless networks). [9]

2.1.1 Security Goals

The security goals for SRTP are to ensure:

- The confidentiality of the RTP and RTCP payloads, and
- The integrity of the entire RTP and RTCP packets, together with protection against replayed packets.
- A framework that permits upgrading with new cryptographic transforms,
- Low bandwidth cost, i.e., a framework preserving RTP header compression efficiency,
- And, asserted by the pre-defined transforms.

- A low computational cost, a small footprint (i.e., small code size and data memory for keying information and replay lists),
- Limited packet expansion to support the bandwidth economy goal,
- Independence from the underlying transport, network, and physical layers used by

RTP, in particular is high tolerant to packet loss and re ordering. These properties ensure that SRTP is a suitable protection scheme for RTP/RTCP in both wired and wireless scenarios.

2.1.2 Features

Besides the security goals mentioned in section 2.1.1, SRTP provides some additional features. They have been introduced to reduce the burden on key management and to further increase security. They include:

- A single "master key" can provide keying material for confidentiality and integrity protection, both for the SRTP stream and the corresponding SRTCP stream. This is achieved with a key derivation function, providing "Master keys" for the respective security primitive, securely derived from the master key.
- In addition, the key derivation can be configure to periodically refresh the Master keys, which limits the amount of cipher text produced by a fixed key, available for an adversary to crypt analyze.
- "Salting keys" are used to protect against pre-computation and time-memory tradeoff attacks.

2.1.3 Secure RTP (SRTP) Packet

The format of an SRTP packet is given fig. 2.1.

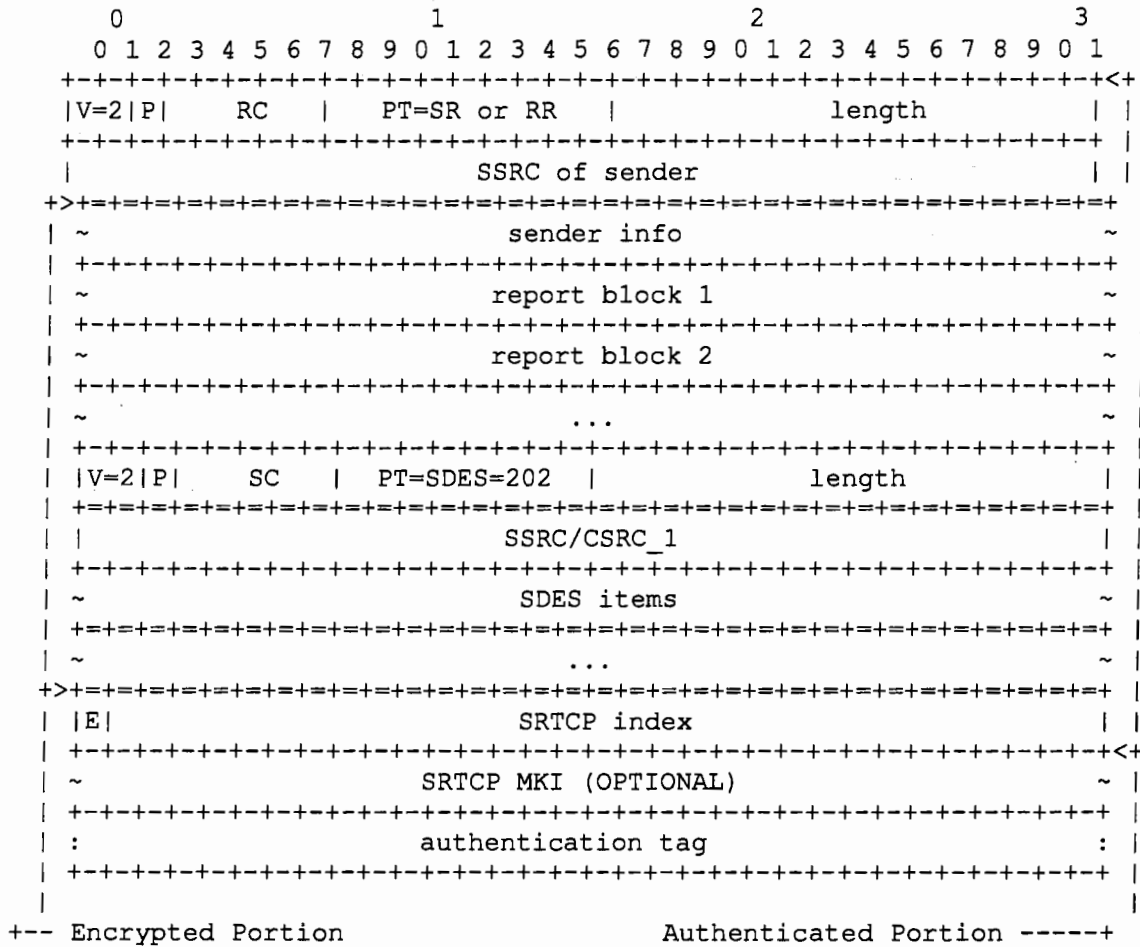


Fig. 2.2 Format of a Secure RTCP packet

The Encrypted Portion of an SRTCP packet consists of the encryption of the RTCP payload of the equivalent compound RTCP packet, from the first RTCP packet, i.e., from the ninth (9) octet to the end of the compound packet. The Authenticated Portion of an SRTCP packet consists of the entire equivalent (eventually compound) RTCP packet, the E flag, and the SRTCP index (after any encryption has been applied to the payload).

2.1.5 SRTP Key Derivation Algorithm

Regardless of the encryption or message authentication transform that is employed (it may be an SRTP pre-defined transform or newly introduced), interoperable SRTP implementations must use the SRTP key derivation to generate Master keys. Once the key derivation rate is properly signaled at the start of the session, there is no need for extra communication between the parties that use SRTP key derivation.

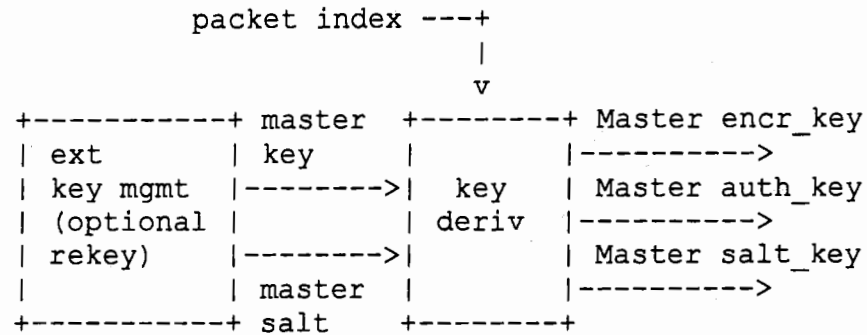


Fig. 2.3 SRTP key derivation

At least one initial key (first key) derivation shall be performed by SRTP, as it is required. Further applications of the key derivation may be performed, according to the values defined in the cryptographic context. The key derivation function SHALL initially be invoked before the first packet.

2.1.6 SRTCP Key Derivation

SRTCP shall, by default, use the same master key (and master salt) as SRTP. To do this securely, following changes shall be done, when applying Master key derivation for SRTCP:

Use <label> = 0x03 for the SRTCP encryption key,

<label> = 0x04 for the SRTCP authentication key and,

<label> = 0x05 for the SRTCP salting key.

Key derivation reduces the burden on the key establishment. As many as six different keys are needed per crypto context (SRTP and SRTCP encryption keys and salts, SRTP and SRTCP authentication keys), but these are derived from a single master key in a cryptographically secure way. Thus, the key management protocol needs to exchange only one master key (plus master salt when required), and then SRTP itself derives all the necessary Master keys (via the first, mandatory application of the key derivation function).

2.1.7 Multicast (one sender)

Just as with (unprotected) RTP, a scalability issue arises in big groups due to the possibly very large amount of SRTCP Receiver Reports that the sender might need to process. In SRTP, the sender may have to keep state (the cryptographic context) for each receiver, or more precisely, for the SRTCP used to protect Receiver Reports. The overhead increases proportionally to the size of the group. In particular, re-keying requires special concern.

Considering SRTCP and key storage, it is recommended to use low-rate (or zero) key derivation (except the mandatory initial one), so that the sender does not need to store too many Master keys (each SRTCP stream might otherwise have a different Master key at a given point in time, as the SRTCP sources send at different times). Thus, in case key derivation is wanted for SRTP, the cryptographic context for SRTP can be kept separately from the SRTCP crypto context. [9]

2.2 MIKEY

Multimedia Internet Keying Protocol (MIKEY) is mainly intended to be used for peer-to-peer, simple one-to-many, and small-size (interactive) groups. One of the main multimedia scenarios considered when designing MIKEY has been the conversational multimedia scenario, where users may interact and communicate in real-time. In these scenarios it can be expected that peers set up multimedia sessions between each other, where a multimedia session may consist of one or more secured multimedia streams (e.g., SRTP streams).

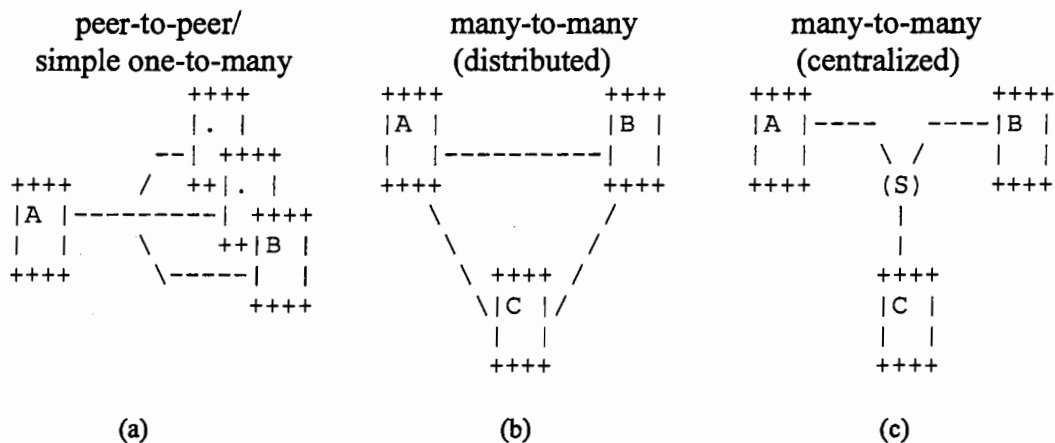


Fig. 2.4 Three Scenarios

Following are some typical scenarios which involve the multimedia applications.

- peer-to-peer (unicast), e.g., a SIP-based call between two parties, where it may be desirable that the security is either set up by mutual agreement or that each party sets up the security for its own outgoing streams.
- simple one-to-many (multicast), e.g., real-time presentations, where the sender is in charge of setting up the security.
- many-to-many, without a centralized control unit, e.g., for small-size interactive groups where each party may set up the security for its own outgoing media. Two basic models may be used here. In the first model, the Initiator of the group acts as the group server (and

is the only one authorized to include new members). In the second model, authorization information to include new members can be delegated to other participants.

d) many-to-many, with a centralized control unit, e.g., for larger groups with some kind of Group Controller that sets up the security.

2.2.1 Design Goals

The key management protocol is designed to have the following characteristics:

- End-to-end security. Only the participants involved in the communication have access to the generated key(s).
- Simplicity.
- Efficiency. Designed to have:
 - low bandwidth consumption,
 - low computational workload,
 - Small code size, and
 - minimal number of roundtrips.
- Tunneling. Possibility to "tunnel"/integrate MIKEY in session establishment protocols (e.g., SDP and RTSP).
- Independence from any specific security functionality of the underlying transport.

2.2.2 System Overview

One objective of MIKEY is to produce a Data SA for the security protocol, including a traffic-encrypting key (TEK), which is derived from a TEK Generation Key (TGK), and used as input for the security protocol.

MIKEY supports the possibility of establishing keys and parameters for more than one security protocol (or for several instances of the same security protocol) at the same time. The concept of Crypto Session Bundle (CSB) is used to denote a collection of one or more Crypto Sessions that can have common TGK and security parameters, but which obtain distinct TEKs from MIKEY. The procedure of setting up a CSB and creating a TEK (and Data SA), is done in accordance with Figure below:

1. A set of security parameters and TGK(s) are agreed upon for the crypto Session Bundle (this is done by one of the three alternative key transport/exchange mechanisms).
2. The TGK(s) is used to derive (in a cryptographically secure way) a TEK for each Crypto Session.
3. The TEK, together with the security protocol parameters, represent Data SA, which is used as the input to the security protocol.

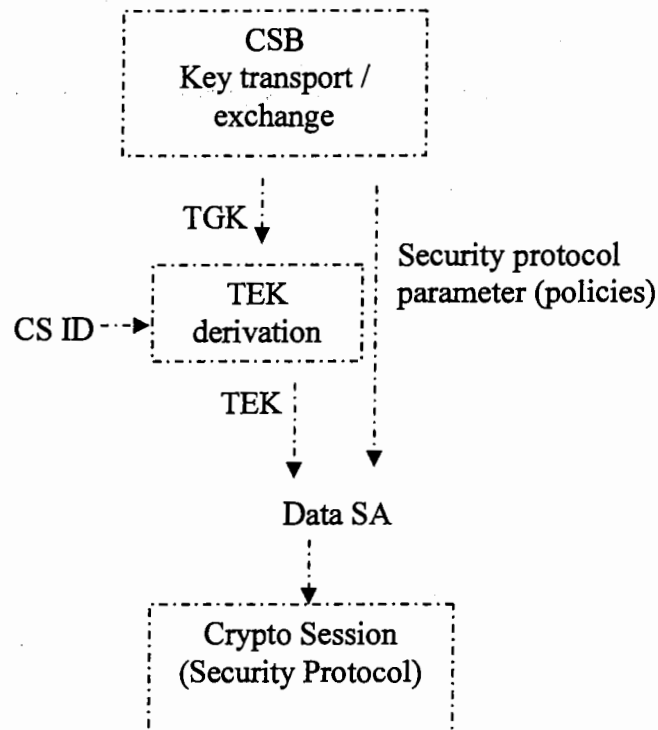


Fig. 2.5 Overview of MIKEY key management procedure

The security protocol can then either use the TEK directly, or, if supported, derive further Master keys from the TEK [9]. It is however up to the security protocol to define how the TEK is used.

MIKEY can be used to update TEKs and the Crypto Sessions in a current Crypto Session Bundle. This is done by executing the transport/exchange phase once again to obtain a new TGK (and consequently derive new TEKs) or to update some other specific CS parameters.

2.2.3 Basic Key Transport and Exchange Methods

Following sub-sections define three different methods of transporting/establishing a TGK: with the use of a pre-shared key, public-key encryption, and Diffie-Hellman (DH) key

exchange. In the following, we assume unicast communication for simplicity. In addition to the TGK, a random "nonce", denoted RAND, is also transported. In all three cases, the TGK and RAND values are then used to derive TEKs. A timestamp is also sent to avoid replay attacks.

The pre-shared key method and the public-key method are both based on key transport mechanisms, where the actual TGK is pushed (securely) to the recipient(s). In the Diffie-Hellman method, actual TGK is instead derived from the Diffie-Hellman values exchanged between the peers.

The **pre-shared case** is, by far, the most efficient way to handle the key transport due to the use of symmetric cryptography only. This approach also has the advantage that only a small amount of data has to be exchanged. Of course, the problematic issue is scalability as it is not always feasible to share individual keys with a large group of peers. Therefore, this case mainly addresses scenarios such as server-to-client and also those cases where the public-key modes have already been used.

Public-key cryptography can be used to create a scalable system. A disadvantage with this approach is that it is more resource consuming than the pre-shared key approach. Another disadvantage is that in most cases, a PKI (Public Key Infrastructure) is needed to handle the distribution of public keys. Of course, it is possible to use public keys as pre-shared keys (by using self-signed certificates).

In general, the **Diffie-Hellman (DH) key agreement method** has a higher resource consumption (both computationally and in bandwidth) than the previous ones, and needs certificates as in the public-key case. However, it has the advantage of providing perfect forward secrecy (PFS) and flexibility by allowing implementation in several different finite groups.

Note that by using the DH method, the two involved parties will generate a unique unpredictable random key. Therefore, it is not possible to use this DH method to establish a group TEK (as different parties in the group would end up with different TEKs). It is not the intention of the DH method to work in this scenario, but to be a good alternative in the special peer-to-peer case.

2.2.4 Generating keys from TGK

MIKEY describes how keying material is derived from a TGK. The key derivation method SHALL be executed using PRF with the following input parameters:

inkey : TGK

inkey_len : bit length of TGK

label : constant || cs_id || csb_id || RAND

outkey_len : bit length of the output key.

The constant part of label depends on the type of key that is to be generated. The table below summarizes the constant values, used to generate keys from a TGK.

Constant | derived key from the TGK

0x2AD01C64 | TEK

0x1B5C7973 | authentication key

0x15798CEF | encryption key

0x39A2C14B | salting key

Table 2.1 Constant values for the derivation of keys from TGK

2.2.5 Key data transport encryption

The default and mandatory-to-implement key transport encryption is AES in counter mode, as defined in RFC of SRTP, using 128-bit key.

2.2.6 MAC and Verification Message function

MIKEY uses 160-bit authentication tag, generated by HMAC with SHA-1 as mandatory implementation method. The authentication key size should be equal to the size of the hash function's output [8].

2.3 AES Algorithm

In cryptography, the **Advanced Encryption Standard (AES)**, also known as **Rijndael**, is a block cipher adopted as an encryption standard by the US government. Unlike its predecessor DES, Rijndael is a substitution-permutation network. AES is fast in both software and hardware, is relatively easy to implement, and requires little memory. As a new encryption standard, it is currently being deployed on a large scale. [26]

2.3.1 AES Algorithm Specification

For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This shows that block size denoted by N_b is 4, which reflects the number of 32-bit words (number of columns) in the State. For the AES algorithm, the length of the Cipher Key, K is 128, 192, or 256 bits. The key length is denoted by N_k is 4, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is denoted by N_r is 10, 12, 14. The only Key-Block-Round combinations that conform to this standard are given in Fig. 2.6.

	Key Length (N_k words)	Block Size (N_b words)	Number of Rounds (N_r)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Fig. 2.6 Key-Block-Round Combinations

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations:

- i) Byte substitution using a substitution table (S-box)
- ii) Shifting rows of the State array by different offsets
- iii) Mixing the data within each column of the State array
- iv) Adding a Round Key to the State

2.4 Diffie-Hellman Algorithm

Diffie-Hellman was the first public-key algorithm invented, and dates back to 1976. It can be used to generate a secret key, but not for encrypting and decrypting messages. Assume parties A and B are involved and want to generate a secret key. A and B agree on a large prime number, n and generator g , such that g is primitive mod n . The two integers, n and g do not have to be secret. A and B can agree to them over an insecure channel.

1. A chooses a large random number x , and computes

$$X = g^x \bmod n$$
 A then sends X to B.
2. B chooses a large random, y and computes

$$Y = g^y \bmod n$$
 B sends Y to B.
3. A computes

$$k = Y^x \bmod n$$
4. B computes

$$k' = X^y \bmod n$$

After the initial exchange between A and B, A knows x and Y , and B knows X and y . Since this is exchanged over an insecure link, X and Y could be found out by a third-party and are public. But either x or y is not obtainable by the third party. Steps 3 and 4 yield the Master key Z ,

$$Z = k = k' = g^{xy} \bmod n$$

The computed key Z can then be used by either the encryption or the authentication algorithm.

The Diffie-Hellman algorithm derives its security from the difficulty of calculating discrete logarithms in a finite field, as compared with the ease of exponentiation in the same field. It is difficult for a third-party to compute the discrete logarithm and recover x or y . The choice of g and n is important, and n must be large. The security of the system is based on the difficulty of factoring numbers the same size of n . [27]

2.5 HMAC, the Keyed Hash-Based MAC Function

HMAC is merely a specific type of MAC function. It works by using an underlying hash function over a message and a key. It is currently one of the predominant means to ensure that secure data is not corrupted in transit over insecure channels (like the internet).

Any hashing function could be used with HMAC, although more secure hashing functions are preferable. An example of a secure hash functions are SHA-14 and MD5. As computers become more and more powerful, increasingly complex hash functions will probably be used. [28]

2.5.1 HMAC Algorithm

HMAC generates a Message Authentication Code by the following formula:

$$\text{HMAC}(M) = H[(K+\text{opad})\|H[(k+\text{ipad})\|M]]$$

M = Message

H[] = Underlying Hash function

K = Shared Secret Key

opad = 36hex, repeated as needed

ipad = 5Chex, repeated as needed

|| = concatenation operation

+ = XOR operation

The HMAC(M) is then sent as any typical MAC(M) in a message transaction over insecure channels. For a graphical illustration, see diagram of the MAC algorithm.

CHAPTER 3

SOLUTION

3. Solution

The problems mentioned in section 1.3 lead us to propose a novel Architecture MMKM to resolve the mentioned basic and advance problems of Multicast environment in MBMS.

3.1 Proposed Architecture

The basic features of multicasting are joining and leaving the group by users while the advance features are efficiency in key delivery, error robustness, load distribution, extra network administration, join/leave notification to other members and mobility. In multicast environment the data need to shift from one member to another to travel in network that make the data vulnerable because the attacker can easily attack at several points in the network at the same time. Some times, data faces the eavesdropping security threat.

Purposed MMKM architecture consists of two servers and multiple clients. The whole Project has following three modules;

1. Registration Server
2. Audio Server
3. Client

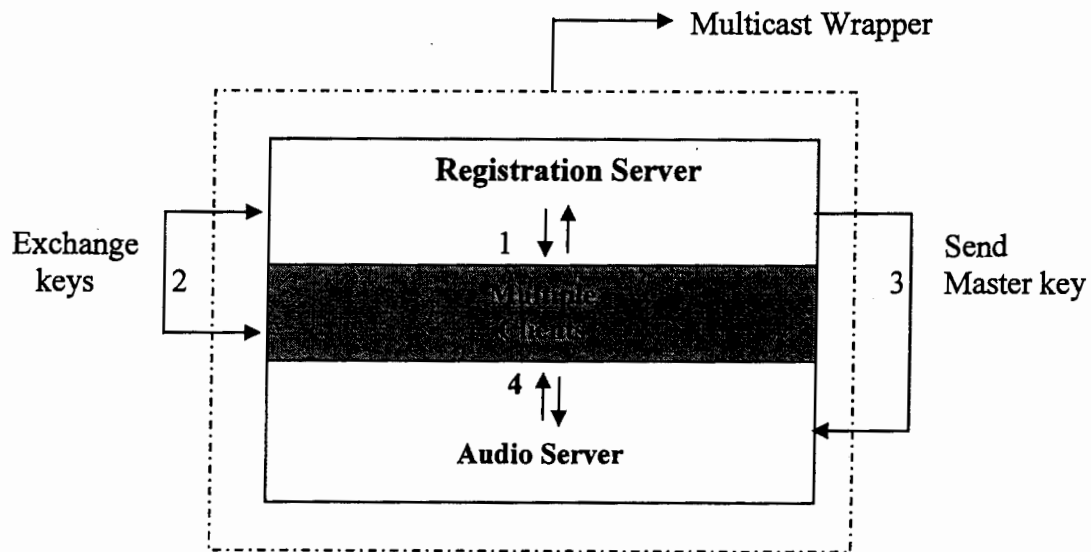


Fig. 3.1 MMKM Architecture
Communication flow of MMKM components

When a member wants to interact with advanced features i.e. want to listen Audio file of the architecture after joining the group, the clients will communicate with Audio Server, and audio server will communicate with Registration server for receiving Master keys to establish session among clients and audio server.

From several contributions, it seems possible to carry out several MSKs updates in the same MIKEY packet. Similarly, it is possible to carry out multiple key/file updates in the same secured packet, which reduces some overhead in header fields. The overhead reduction is a key concern of MMKM Architecture.

The MMKM architecture will provide complete authentication to registered members which automatically provide security and also provide better Quality of Services to users which is the aim of MMKM architecture.

MMKM architecture will maintain a list of join and/or leave notification of all registered users according to their joining and leaving time. Data Packet loss during communication will be minimal. Therefore, it should be possible to provide better service continuity in the communication flow of MMKM components.

3.2 Design

Good design is the key to effective engineering. Design is a creative process requiring insight and flair on the part of the designer. It must be practiced and learned by experience and study of the existing systems. We need design to study, understand the problem and describe each abstraction used in the solution.

The design process involves developing several models of the system at different levels of abstraction. If a design is decomposed, Errors and omissions in earlier stages are discovered. These feed back allow earlier design models to be improved.

3.2.1 View Point Analysis

A system is designed on the basis of Architecture mention in section 3.1 with subsequent flow diagrams. The system is structured into a number of principal subsystems as shown in Fig. 3.2, 3.3.

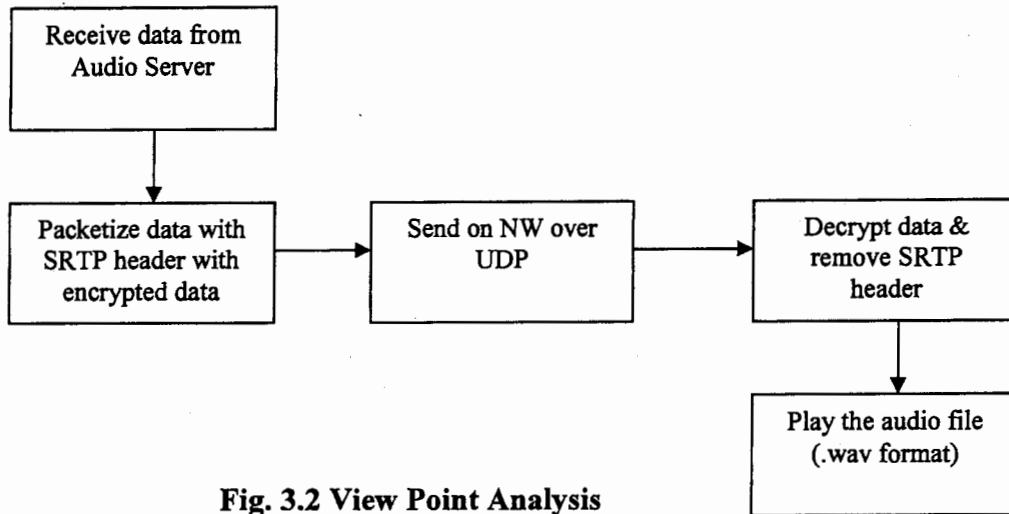


Fig. 3.2 View Point Analysis

A client-server architectural model a distributed system model has been followed. It shows how data and processing is distributed across a range of processors. But since the focus is on two servers (Registration server, Audio server) and a multiple clients therefore the model is simplified with processing divided just between the server and clients.

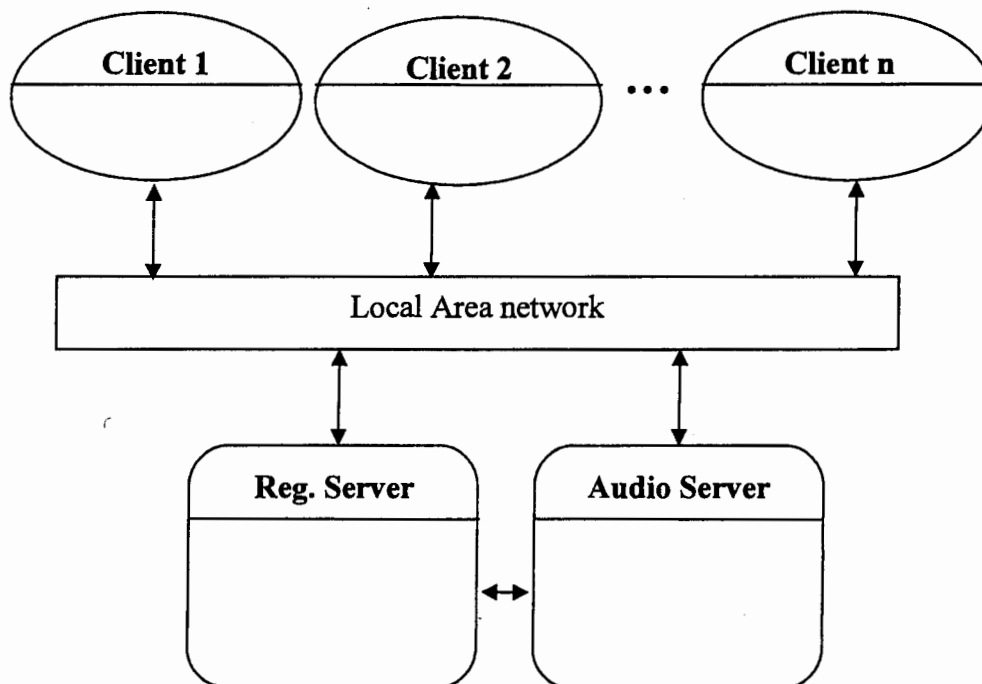


Fig. 3.3 System Structure

3.2.2 Client-Server Emphasized

The architecture described in section 3.1 fulfills solutions for the problems in section 1.3 based on client-server environment. First server is Registration Server with attributes of Communication through FDSET on UDP Ports 9505 and 9507 and input from Keyboard and do processing on events like get user information, save Information in file and exit. Registration Server is designed to provide services like Register New User, Generate DH key for new user, Generate Master Key for session establishment, Login registered users and Change Password. The initial start and exit processing of Registration server is shown in fig. 3.4, 3.5.

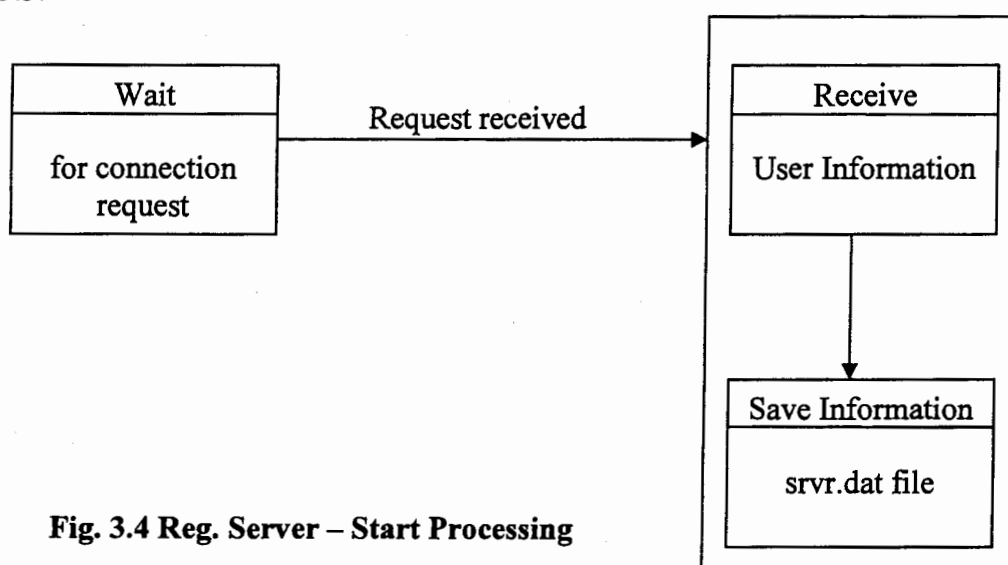


Fig. 3.4 Reg. Server – Start Processing

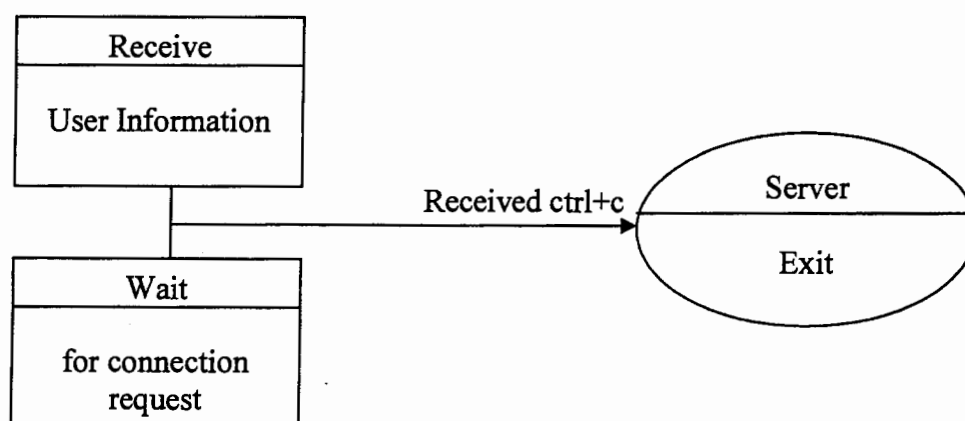


Fig. 3.5 Reg. Server – Exits

First server is Audio Server with attributes of Communication through FDSET on SRTP & SRTCP Ports i.e. 9501, 9503 and 9504 and it will do processing on events mention in Fig. 3.6, 3.7, 3.8. Audio Server is designed to provide services like Receive Audio Choice from Client, Initialize Sound device (dsp), Send SRTP Packets with encrypted audio payload, Produce and display Sender Report SR, Receive SRTCP packets from client, including, Receiver report, and BYE packet if any, Display SRTP & SRTCP (RR, SR, BYE) info on screen.

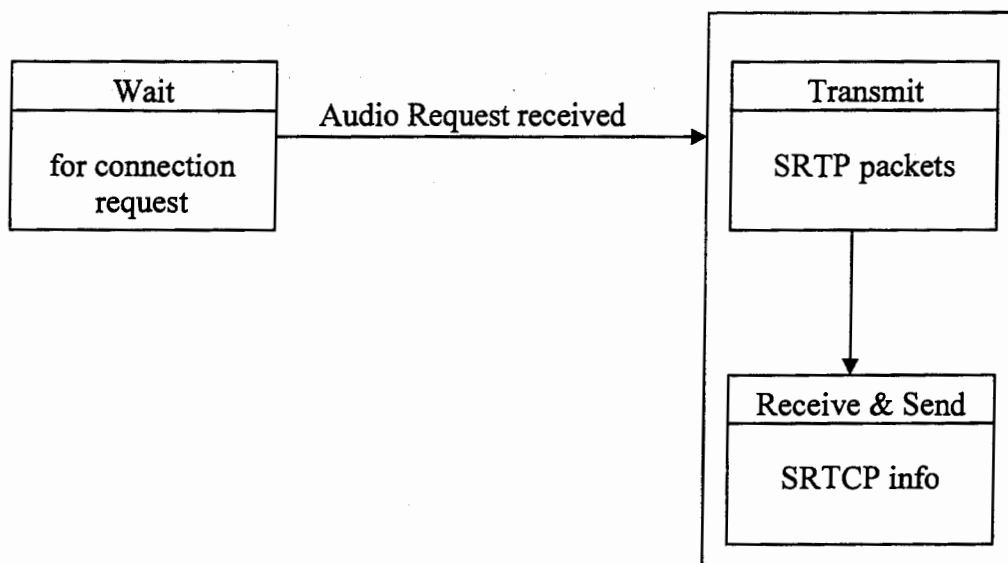


Fig. 3.6 Server – Start Transmission

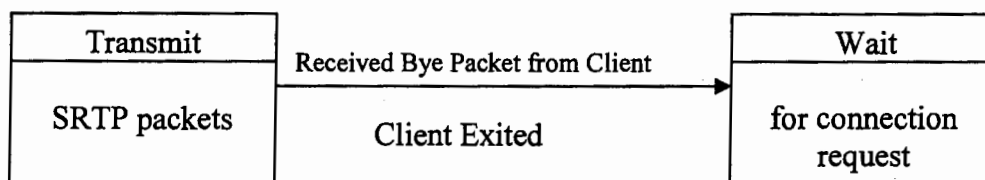


Fig. 3.7 Server – End Transmission

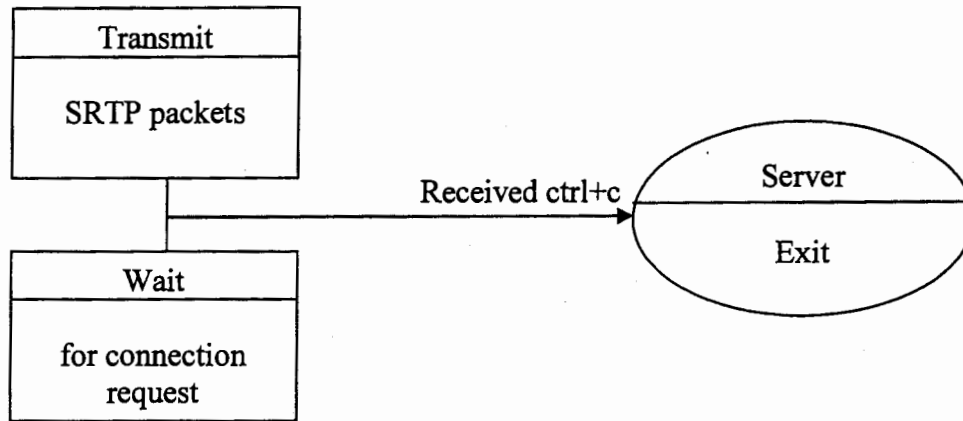


Fig. 3.8 Server – Exits

Multiple Clients can communicate with both Registration and Audio server. The single client has attributes of Communication through FDSET on SRTP, SRTCP Ports 9501, 9503 and 9504 and input from keyboard. Does processing is done on events like Send Request of Connection to server, Send BYE packet on exit, End Transmission. The Client of application is designed to provide services like Receive SRTP packets from server, Play audio streams, Produce Receiver report and BYE packet of client if any, Send SRTCP packets to server.

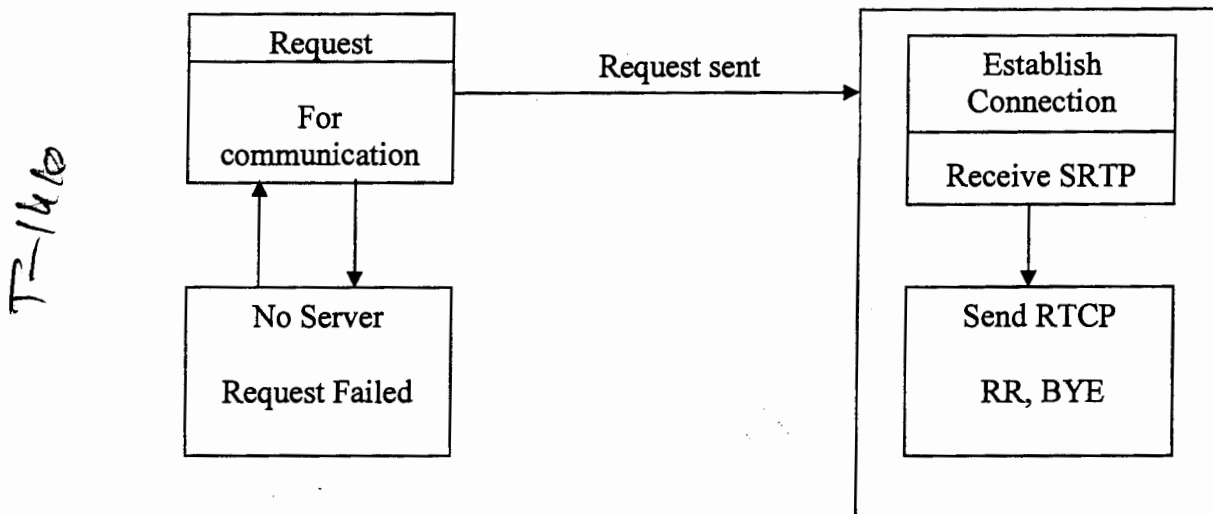


Fig. 3.9 Client starts Transmissions

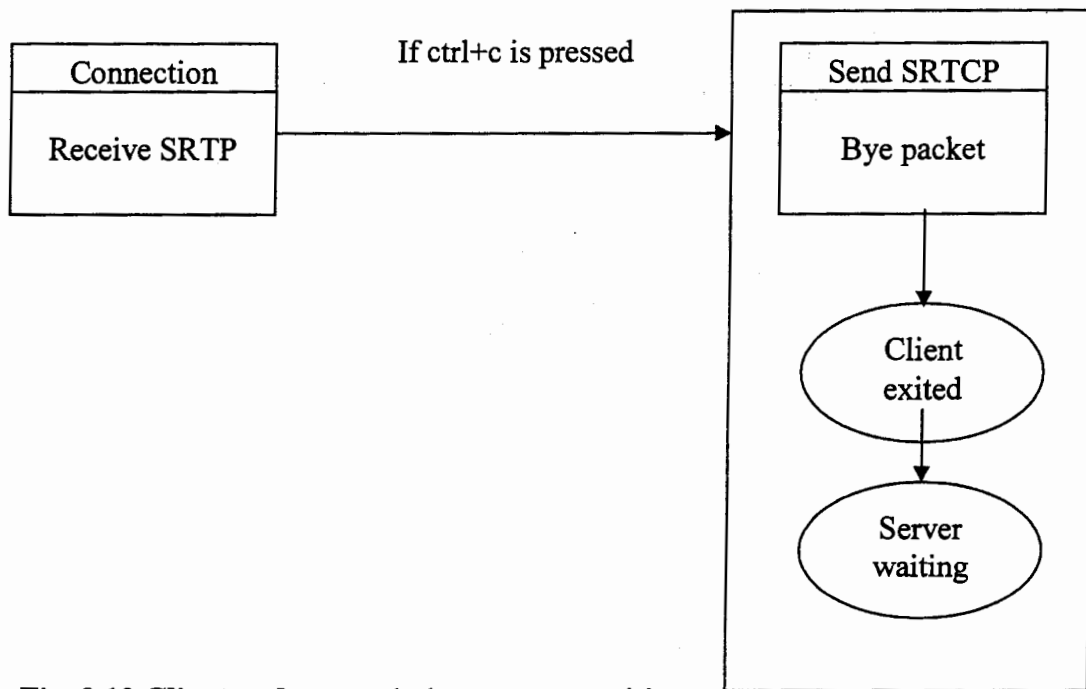


Fig. 3.10 Client end transmission server awaiting

3.2.3 Control Modeling

The general model of the control relationships between different parts of this system is:

Centralized Control: The Audio Server has central control of sending packetized audio data.

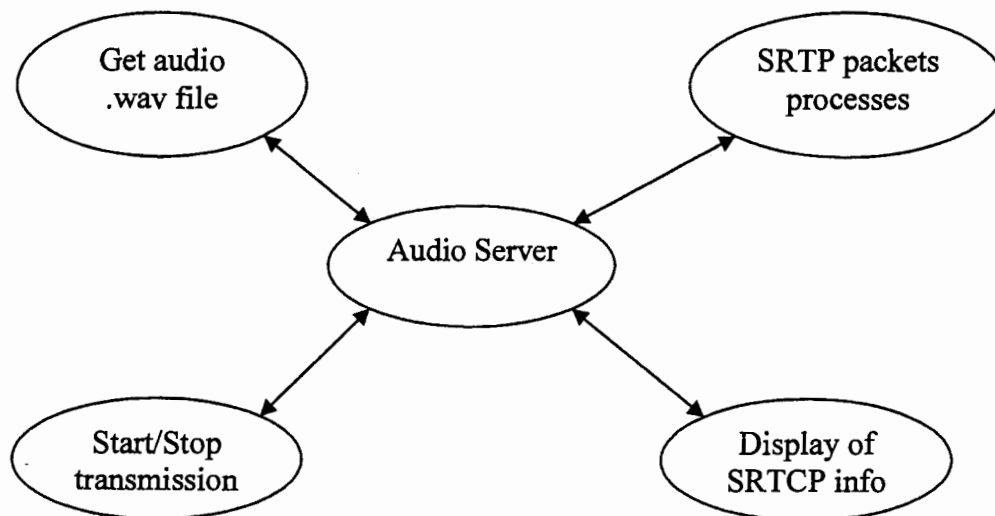


Fig. 3.11 A centralized control model for a real time audio server

Audio stream is transmitting to the multiple clients. During transmission, it can make the client exit by force or can make it quit, make the client exit as well at the same time. It is the responsibility of audio server to get audio stream and transmit it over the network. Furthermore the server also displays the SRTCP information on its own end.

Event-based Control: Audio server is always in running state so when client is logged in, he adds himself in the multicast group and sends his desired request for audio stream then he receives an audio stream according to audio queue maintained on server. After every 5 sec the SRTCP data is sent from the client to the server and reports are updated at server end. If client wants to send bye packet to the server computer and end transmission it can do so by pressing ctrl+c keys. The client can exit and stop receiving network data. The client is responsible for playing audio stream on receiving it from the server computer.

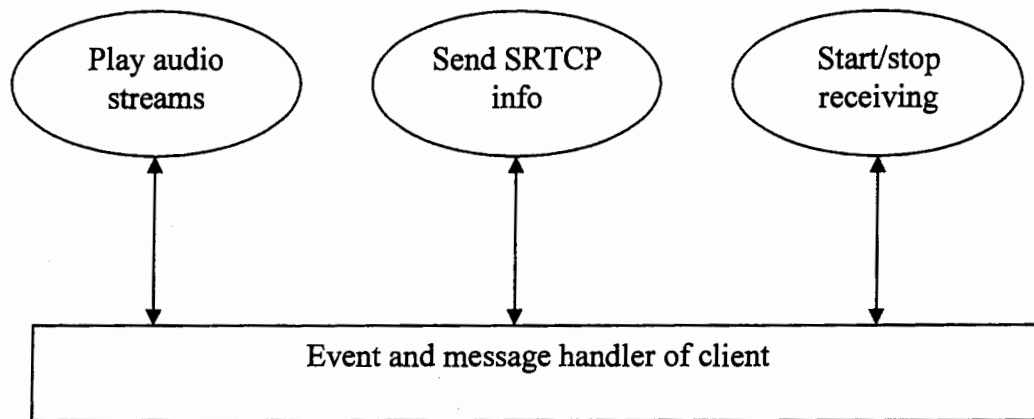


Fig. 3.12 Control model based on selective event handling

3.3 Modular Decomposition

Overall Design specifications lead us to decompose proposed system into following three modules.

3.3.1 Registration Server

- i. Establish/open sockets on two ports 9505 for client registration, login and 9506 for sending key to Audio server.
- ii. Wait for connection from client in the form of new client registration, login or change password request.

3.3.2 Audio Server

- i. Establish/open sockets on many ports, basic SRTP socket on Port 9500 and SRTCP socket on 9503, and many other UDP ports i.e. for keys exchange and handling client requests are also used.
- ii. Open audio device and access .wav file from disk.
- iii. Packetized and encrypt audio streams.
- iv. Attach SRTP header with encrypted payload.
- v. Allot sequence no. and time stamp and send data to client
- vi. After every 5 sec, it displays SRTCP reports on screen i.e SR and RR.
- vii. On receiving bye message from client, display bye packet along with the source of Bye packet and reason to go.

3.3.3 Client

- i. Establish sockets
- ii. Send SRTCP RR packet to server.
- iii. Receive each SRTCP SR packet
- iv. Remove SRTP header from encrypted payload and decrypt payload.
- v. Play audio streams.
- vi. Send SRTCP reports after every approximately 5 seconds
- vii. When finished listening then send bye packet to server.

CHAPTER 4
IMPLEMENTATION

4. Implementation

The work has been implemented in Linux environment for which Red Hat Linux 8.0 / 9.0 are used. The project is implemented in GCC under Linux environment. All network communication is through standard Linux sockets and is according to ANSI/C standards for UDP networking.

Sound device is installed in Linux. The encrypted audio stream is sent over the network via SRTP packets. On the other end, again in Linux, SRTP packets are decrypted, and played on Client's end. SRTP and SRTCP Packets status are displayed on both Server and Client screen.

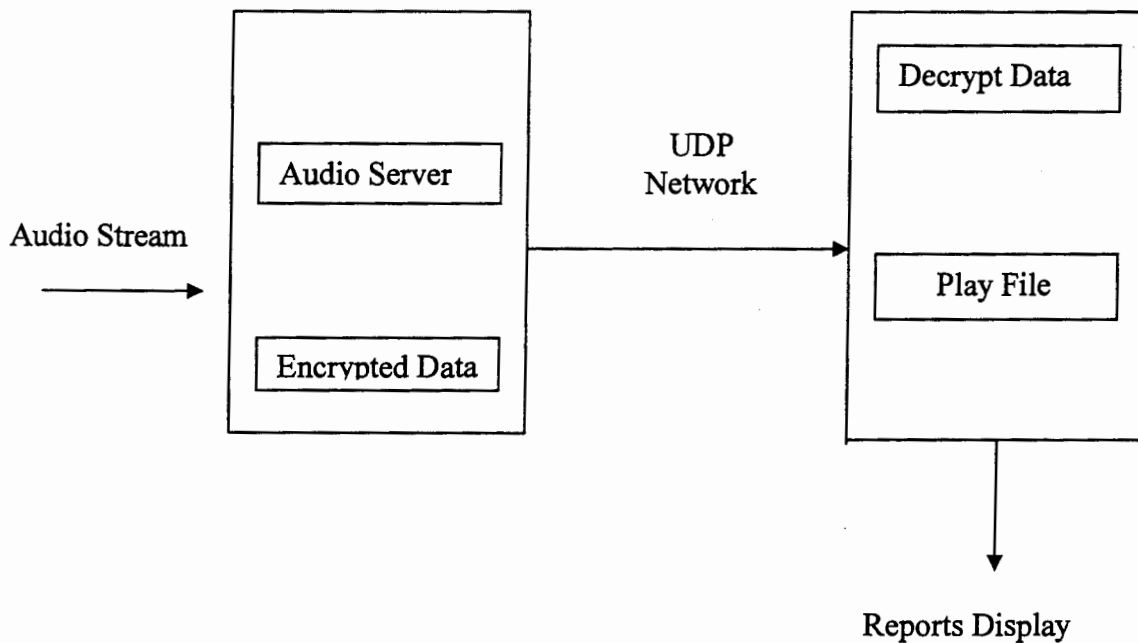


Fig. 4.1 Playing Process

4.1 Sending Audio through SRTP over Network

MMKM application obtains audio streams (.wav file), packetizes it into an SRTP packet. Then it is handed over to the networking module, which is responsible for sending the packet over the network. At the client end, the networking module receives the packet, and renders it. This section deals with the networking issues connected with this application.

4.1.1 UDP

Secure RTP typically runs on top of UDP to make use of its multiplexing and checksum functions. UDP provides a connectionless and unreliable datagram service over the network. UDP was chosen as a target transport protocol for RTP [29] and then similarly for Secure RTP because of three reasons.

- i. First, RTP is primarily designed for multicast, while the connection-oriented TCP does not scale well and therefore is not suitable. TCP cannot support multicast.
- ii. Second, for real-time data, reliability is not as important as timely delivery. Even more, reliable transmission provided by retransmission as in TCP is not desirable. For example, due to network congestion, some packets might get lost and the application would result in lower but acceptable quality. If the protocol insists on a reliable transmission, the retransmitted packets could possibly increase the delay, jam the network and eventually starve the receiving application. Thus reliable transmission is inappropriate for delay-sensitive data such as real-time audio and video.
- iii. The TCP congestion control mechanisms decrease the congestion window when packet losses are detected (“slow start”). Audio and video, on the other hand, have “natural” rates that cannot be suddenly decreased without starving the receiver. For example, standard PCM audio requires 64 kb/s, plus any header overhead, and cannot be delivered in less than that. Video could be more easily throttled simply by slowing the acquisition of frames at the sender when the transmitters send buffer is full, with the corresponding delay. The correct congestion response for these media is to change the audio/video encoding, video frame rate, or video image size at the transmitter, based, for example, on feedback received through RTCP receiver report packets.
- iv. An additional small disadvantage is that the TCP header is larger than a UDP header (40 bytes for TCP compared to 8 bytes of UDP).
- v. Also, these reliable transport protocols do not contain the necessary timestamp and encoding information needed by the receiving application, so that they cannot replace RTP. (They would not need the sequence number as these protocols assure that no losses or reordering takes place).

4.2 Multithreading

There is a need to continuously play the audio file and also we need an infinite loop to watch the file descriptors for requests of other login clients. The only efficient solution was to use separate threads for concurrent execution of many tasks.

This application uses a thread to play the audio file. The application begins and after initialization, it starts a new thread and starts playing audio file. When the signal is received from the client to start transmission, value of a global variable is changed and the transmission of audio packets starts over the network. This has made the application quite efficient.

4.3 Implementation Stages

The numbers of stages listed below are being followed to accomplish the project.

- i. **Playing of Audio file:** In this stage it was studied how to play the audio files in text mode of Linux.
- ii. **Cryptography:** Advance Encryption Standard (AES) Algorithm was chosen for encryption and decryption of data.
- iii. **Networking:** Transmission of data over network and implementation of SRTP.
- iv. **Keys generation and distribution:** Diffie-Hellman key exchange algorithm is used to generate and exchange keys of two parties. HMAC with MIKEY used for other keys generation and distribution.
- v. All the processes were integrated. It comprised of following stages.
 - At the Registration server, new users registered then login to interact with Audio server.
 - The Audio server, access audio file from disk then formed SRTP packets with encrypted audio payload and transmitted over the network.
 - Audio file was received in SRTP packets on the other end, decrypt packets one by one and played. Only disk space and access was required at the server because audio file was not written on client disk.
- vi. Developed code for reports of SRTCP for preparing, transmitting the network performance statistics.
- vii. Developed interface for the Audio Server and client to receive and display reports.

4.4 Application Modules

The whole application consists of three modules:

- i. Registration Server (RSRVR)
 - ii. Audio Server (ASRVR)
 - iii. Client (CLNT)
-
- i. **RSRVR** Wait for connection from client in the form of new client registration, login or change password request.
 - ii. **ASRVR** Open audio device and access .wav file from disk. Packetized and encrypt audio streams. Attach SRTP header with encrypted payload. Allot sequence no. and time stamp and send data to client. After every 5 sec display stats on screen: SRTCP SR and RR. On receiving bye message from client, display bye packet along with the source of Bye packet and reason to go.
 - iii. **CLNT** It sends SRTCP RR packet to server. Receive each SRTCP SR packet. Remove SRTP header from encrypted payload and decrypt payload. Play audio streams. Send SRTCP reports after every approximately 5 seconds. When finished listening then send bye packet to server.

4.4.1 Registration Server

The client registers himself/herself to the Registration server. If we agree to register him/her, we shall send him a master key, and a username and password. Using Diffie-Hellman method, a key will be generated which will encrypt the username, password and the master key and send it to the user. Its implemented pseudo code is mention in section 4.6.

Registration Process: When the Client registers, he/she has following information:

- a. Username, Password and credit card number.
- b. DH Key (embedded in the Client Software).
- c. Client Software to connect and play audios.

4.4.2 Audio Server

- a. In response, the audio server will send to the user, a list of already queued audios. The Audio server will maintain a queue of 10 audios. So, depending upon the length of the queue, the server will inform the user whether the queue is full or otherwise. If the queue is not full, the server will add the request to the queue and inform the user accordingly. The server will also inform the user, if that audio is already queued. In that case, the request for the same audio will not be queued.
- b. Every packet that is received by the client, it first read its MAC (Message authentication code). It will then calculate MAC itself. This will be calculated over the headers of the SRTP or SRTCP packet. The calculated authentication code and the code received in the packet must match. If they do not, the data is corrupt and may be rejected. The server may be informed accordingly. If the authentication code matches, then the audio data is decrypted and played on the client machine.
- c. At one time, only one audio file will be played by the server because of Multicast environment.
- d. No billing system will be maintained. However, server will maintain a log for any user logging in and logging out and a separate log to maintain list of audio files being played.

Its implemented pseudo code is mention in section 4.6.

4.4.3 Client

The user will install the software. User's computer must have a sound card and the Linux driver. The client software will test the sound card and play a sound. If the user has successfully heard the test sound he/she will be able to listen audio file. If this is not successful then the user must make arrangements to put things right. When user wishes to listen to the audio files he/she will run the client software. Its implemented pseudo code is mention in section 4.6.

- i. The client software will contact with two Servers. One is called Audio Server, used for audio file data (SRTP Packets) receiving, playing and for passing control information (SRTCP Packets) between client and server. The other will be Registration Server, which used to register users by username and password.
- ii. The client software will ask for username and password from the user. A facility is provided to the client software to save the username, password and DH key for matching when client again login after registration.

- iii. The client software will send username and password to the Registration server. The registration server will verify the username and password and send the Master Key to the user.
- iv. If the username and password do not verify, the user will be informed accordingly and no Master key will be sent to the user. He/she will be asked to register first. If the login is successful, a greeting message will be displayed,

“Welcome to IIUI Audio Server”

- v. After receiving the Master key, the audio files i.e audio data option list will display client, client will enter the choice then will receive audio file from Audio server and start listening the audio file which will already multicast. So, he/she may only listen to only a part of the audio file, depending upon the time, he/she has put on her computer and ran the software. After that, he/she will listen to complete audio file, as being played by the Audio server.
- vi. The user can request for a audio file is his/her choice. The client software will provide a menu item, whereby the user can send a request of his/her own audio file to the server. Following mechanism is adopted.
 - a) When the user will login, a complete list of audio file will be displayed on his screen. The Audio server will send the list. It may contain 100 or 200 audio files, or may be more. This list will be changed by the server after a month.
 - b) The user will then send a request for a particular audio file, by giving its serial number. This request will be sent on Audio Server.
- vii. When the user wishes to end the listening to the audio file, it will press ctrl+c to exit. The client software will send a message to the server that it is exiting and then terminate.
- viii. After some time, the user again logs in, on the same day. He/she will again have to give the username and password and will receive a Master key. Then he/she can start listening to the audio file.

4.5 Key Management Mechanism

Application manages eight keys. The implemented pseudo code of key mechanism is mention in section 4.6.

- i. Master key

- ii. Diffie-Hellman key
- iii. Three SRTP keys (Authentication Key, Encryption Key, Salt Key)
- iv. Three SRTCP keys (Authentication Key, Encryption Key, Salt Key)

4.5.1 Master key (TGK)

Type: A 16 byte random number.

Generation: Generated by Reg. server everyday at 12 o' clock midnight.

Distribution Method: Diffie-Hellman

Generation and Distribution Period: Once every day, when customer logs in. If he login more than once on the same day, he/she will again get the same key. The timings will be of the country where main server is located. The distribution of the key will be transparent to the user. The user will only send login information and the key will be received by the client software automatically.

Purpose: Generate Authentication Keys.

4.5.2 Diffie-Hellman Key

Type: Symmetric Key

Generation: Generated by Reg. server by using Diffie-Hellman Algorithm

Distribution Method: Diffie Hellman

Generation and Distribution Period: Generate and distribute once at the time of registration.

Purpose: Encrypt and decrypt Master key to make it secure while traveling on network.

4.5.3 SRTP Keys

Authentication Key

Type: Symmetric Key

Generation: Generated by both Audio server and client by using Master key in HMAC method.

Generation Time: At the Application run time before sending and receiving SRTP packets.

Distribution Period: Constantly, on every sending and receiving SRTP packets.

Distribution Method: Send it in SRTP packet.

Purpose: Used as MAC of every SRTP Packet. The headers of the packet will go in clear text. But the authentication code (MAC) will accompany them. The HMAC method will be used for calculation.

Encryption key

Type: Symmetric Key

Generation: Generated by both Audio server and client by using SRTP Authentication key in HMAC method.

Generation Time: At the Application run time.

Purpose: Used in AES method to encrypt the audio data on Audio server and to decrypt the audio data on client.

Salt key

Type: Symmetric Key

Generation: Generated by both Audio server and client by using SRTP Encryption key in HMAC method.

Generation Time: At the Application run time.

Purpose: Used to alter some bytes of encryption key to enhance security.

4.5.4 SRTCP Keys

Authentication Key

Type: Symmetric Key

Generation: Generated by both Audio server and client by using Master key in HMAC method.

Generation Time: At the Application run time before sending and receiving control SRTCP packets.

Distribution Period: Constantly, on every sending and receiving SRTCP (RR and SR) packets.

Distribution Method: Send it in every SRTCP (RR and SR) packets.

Purpose: Used as MAC of every SRTCP (RR and SR) packet. The headers of the packet will go in clear text. But the authentication code (MAC) will accompany them. The HMAC method is used for calculation.

Encryption key

Type: Symmetric Key

Generation: Generated by both Audio server and client by using SRTCP Authentication key in HMAC method.

Generation Time: At the Application run time.

Purpose: Used in AES method to encrypt control headers on Audio server and to decrypt on client.

Salt key

Type: Symmetric Key

Generation: Generated by both Audio server and client by using SRTCP Encryption key in HMAC method.

Generation Time: At the Application run time.

Purpose: Used to alter some bytes of encryption key to enhance security.

4.6 Pseudo Codes

Registration Server

procedure 1.1rs create_regsocket()

begin

define, establish and bind Registration socket(regSocket).

fill sockaddr_in with INADDR_ANY and REGPORT # 9505

end

procedure 1.2rs create_keysocket()

begin

define and establish Key Socket(keySocket).

fill struct sockaddr_in keyServer with INADDR_ANY and KPORT #

9506

end

procedure 1.3rs gensesskey()

begin

Open sesskey.dat file in read mode

Read sesskey from file, if current date and sesskey generation date match, close sesskey.dat file.

If current date and sesskey generation date doesn't match

 Generate new sesskey by using time function

 Overwriting previous sesskey by new in sesskey.dat file

Close sesskey.dat file.

end

procedure 1.4rs readuserfile()

begin

Open srvr.dat file in read mode to read registered users.

loop

 Read all users from file and store in memory

end loop

Close srvr.dat file

End

procedure 1.5rs Registration Process

if Svr Receives Registration Request from Client

begin

Receive userinfo from client

procedure 1.5.1rs verifyccno()

begin

 Verify user's entered credit card no.

end

procedure 1.5.2rs verifyuser()

begin

 Verify user's entered username by matching the already registered username.

 If username matches send failure o.w. send success to CLNT and proceed.

end

procedure 1.5.3rs calcdh()**begin**

Srvr's public key generated by Prime and base no.
 Send Srvr's Public key to CLNT on regSocket.
 Receive CLNT Public key on regSocket.
 Generate Srvr's private key by secret key of CLNT.

end**procedure 1.5.4rs SaveMsg()****begin**

Open srvr.dat file in Append mode to save new user info.
 Write userinfo struct in srvr.dat file
 Close srvr.dat file
 re-read the srvr.dat file to take account of new added user

end**end****procedure 1.6rs chngpasswd()**

if Srvr Receives Changing Password Request from Client

begin

Receive Username, old Password and new Password from CLNT on regSocket.
 Verify username and password
 Write back changes in srvr.dat file
 Send Success to CLNT on regSocket.

end**procedure 1.7rs dologin()**

if Srvr Receives Login Request from Client

begin

Receive userinfo from client on reg Socket.
 Verify userinfo from srvr.dat file
 Get Diffie-Hellman key from srvr.dat file and save in memory
 Send Success or Failure msg to CLNT on acceptance or failure of info on regSocket.
 On Success, send encrypted Session key by dhkey to CLNT.

end**Audio Server****procedure 2.1as create_socket()****begin**

define and establish Audio Socket(audioSocket).

fill struct sockaddr_in localaddr with INADDR_ANY and port 0.
 Fill the destination address struct sockaddr_in destaddr with
 GPADDR and SPORT # 9501.

Define, establish and bind Sync Socket(syncSocket).
 fill struct sockaddr_in localaddr1 with INADDR_ANY and SYCPORT #
 9502

initialize SRTP header.

end

procedure 2.2as audioreq()

handle in Thread "q_thread"

begin

define, establish and bind Audio Queue Socket(qSocket).
 fill struct sockaddr_in qServer with GPADDR and QPORT # 9507

loop

Receive Audio file list request from CLNT on qSocket.
 Send list of Audio files to CLNT on qSocket.
 Receive user Choice input from CLNT on qSocket.
 Enter user choice in audio file list queue.

end loop

end

procedure 2.3as getsesskey()

begin

define, establish and bind Key Socket(keySocket).
 fill struct sockaddr_in keyServer with SRVRADDR and KPORT #
 9506

Receive sesskey from Registrarion Svr on keySocket.

end

procedure 2.4as hmac()

begin

Generate Authentication key, Encryption key, Salt key by using MIKEY key
 generation Method.

end

procedure 2.5as sndwavdata()

handle in Thread "a_thread"

begin

define, establish and bind Control Socket(contSocket).
 fill struct sockaddr_in destaddr1 with GPADDR and CPORT # 9500

loop

Send initial 44 bytes to initialize sound device(dsp)to CLNT on contSocket.

end loop
end

procedure 2.6as clntrep()

handle in Thread "b_thread"

begin

define, establish and bind SRTCP Socket(mySocket2).
fill struct sockaddr_in localaddr with INADDR_ANY and CLPORT # 9503
Multicast mySocket2 on GPADDR "224.1.2.3"

Define and establish Sender Report(SRTCP)Socket(srSocket).
fill struct sockaddr_in srClient with GPADDR and SRPORT # 9504

initialize SRTCP header for Sender Report(SR)

loop

If any new user joins or existing user left the group, Display its IP address
with joining and/or leaving time.

Else

Receive SRTCP RR Packet from CLNT on mySocket2.

Send SRTCP SR Packet to CLNT on srSocket.

end loop

end

procedure 2.7as PlayFile()

begin

loop

Open currently selected audio file from queue in read mode.

Read initial 44 bytes to initialize dsp from the currently opened audio file

loop

Read 16384 bytes packets from remaining audio file till end of file

Encrypted each packet by encdeckey

Receive request for every audio packet from CLNT on syncSocket.

Send SRTP Packets with encrypted audio data to CLNT on
audioSocket.

procedure 1.12.1c calcmac()

begin

mac() calculated on Srvr by using authkey to check the
authenticity of every SRTP packet.

end

Print SRTP headers after every 50 packets.

Close the currently open audio file on end of file

Send 0 byte packet to CLNT as an indicator of end of file
 Audio list queue decrement at every end of audio file

```

    end loop
  end loop
end

```

Client

procedure 3.1c create_regsocket()

```

begin
  define and establish Registration socket(regSocket).
  fill sockaddr_in with SRVADDR and REGPORT # 9505
end

```

procedure 3.2c readuserfile()

```

begin
  Open clnt.dat file in read mode to read registered users.
  loop
    Read all users from file and store in memory
  end loop
  Close clnt.dat file
end

```

procedure 3.3c sendregnreq()

if cmnd line arguments "-R"

```

begin
  loop
    Enter Username, Password and Credit card no. for
    Registration.
    Send information on Srvr on regSocket.
    Receive Success or Failure msg from Srvr on acceptance or failure of info on
    regSocket.
  end loop

```

```

end

```

procedure 3.4c calcdh()

```

begin
  Client's public key generated by Prime and base no.
  Send CLNT's Public key to Srvr on regSocket.
  Receive Srvr Public key on regSocket.
  Generate CLNT's private key by Srvr's secret key
end

```

procedure 3.5c SaveMsg()**begin**

Open clnt.dat file in Append mode to save msg in file
 Write userinfo struct in clnt.dat file
 Close clnt.dat file
 re-read the clnt.dat file to take account of new added user.

end**procedure 3.5c chngpasswd()**

if cmd line arguments "-P"

begin

Enter Username, old Password and new Password to change password.
 Verify New Password
 Write back changes in clnt.dat file
 Send new password to Server on regSocket.
 Receive response Success or Failure msg from Server on acceptance or failure of info.

end**procedure 3.6c dologin()****begin**

Enter Username, Password for normal Login
 Get Deffie Helmen key from clnt.dat file and save in memory
 Send Login info to Srvr on reg Socket.
 Receive Success or Failure msg from Srvr on acceptance or failure of info on regSocket.
 Get encrypted Session key from srvr after successful login.
 Decrypt sesskey by dhkey, proceed to listen the audio file.

procedure 3.6.1c hmac()**begin**

Generate Authentication key, Encryption key, Salt key by using MIKEY key generation Method.

end**end****procedure 3.8c create_socket()****begin**

define, establish and bind Control Socket(contSocket).
 fill struct sockaddr_in localaddr with INADDR_ANY and CPORT #
 9500

define, establish and bind Audio Socket(audioSocket).
 fill struct sockaddr_in localaddr1 with INADDR_ANY and SPORT #


```

9501
Multicast audioSocket on GPADDR "224.1.2.3"

define and establish Sync Socket(syncSocket).
fill struct sockaddr_in syncServer with SRVADDR and SYCPORT #
9502

define, establish and bind Sender Report(SRTCP)Socket(srSocket).
fill struct sockaddr_in srClient with INADDR_ANY and SRPORT #
9504
Multicast audioSocket on GPADDR "224.1.2.3"

```

end

procedure 3.9c audioreq()

begin

```

define and establish Audio Queue Socket(qSocket).
fill struct sockaddr_in qServer with GPADDR and QPORT # 9507
Send Audio list request to Srvr on qSocket.
Receive list of Audios from Srvr on qSocket.

```

loop

```

Take input choice no. from user through keyboard.
Send Choice input to Srvr on qSocket.

```

end loop

end

procedure 3.10c clntrep()

handle in "a_thread" Thread

begin

```

define and establish SRTCP Socket(mySocket2).
fill struct sockaddr_in destaddr2 with GPADDR and CLPORT # 9503
initialize SRTCP header for Receiver Report(RR)

```

loop

```

Send SRTCP RR Packet to Srvr on mySocket2.
Receive SRTCP SR Packet from Srvr on srSocket.

```

end loop

end

procedure 3.11c capture()

handle in Signal Call(press ctrl+c)

begin

```

Send SRTCP BYE Packet to Srvr on mySocket2 and Exit.

```

end

procedure 3.12c PlayFile()

begin

```

loop
    Receive 44 initial bytes to initialize sound device(dsp)on contSocket.
loop
    Send request for every audio packet to srvr on syncSocket.
    .Receive SRTP Packets with encrypted audio data on audioSocket.
    Print SRTP headers after every 50 packets.

    procedure 1.12.1c calcmac()
    begin
        mac() calculated on CLNT by using authkey to check the
        authenticity of every SRTP packet.
    end
    decrypt the audio data by encdeckey.
    play audio packet if srvr & CLNT mac matched o.w.
    Exit.
    end loop
end loop
end

```

4.7 Makefile

Linux provides a very useful means of compiling programs. You can include all the libraries and all the files and clean old files and so many other actions. We have used the following Makefile, which we created our self for compiling our programs. Here is the extract of Makefile:

```

# Makefile
OPT= -O2
# For profile: -pg
all: rs as c
rs: rsvr.o hint.o arith.o dh.o
    gcc $(OPT) -o rsvr rsvr.o hint.o arith.o dh.o aes.o
as: ASRVR.o arith.o
    gcc $(OPT) -o ASRVR ASRVR.o aes.o -lpthread -lcurses -lcrypto
c: clnt.o hint.o arith.o dh.o
    gcc $(OPT) -o clnt clnt.o hint.o arith.o dh.o aes.o -lcurses -lpthread -lcrypto
hint.o: hint.h hint.c
    gcc $(OPT) -c hint.c
ASRVR.o: ASRVR.c
    gcc $(OPT) -c ASRVR.c

```

```
rsrvr.o: rsrvr.c
    gcc $(OPT) -c rsrvr.c
clnt.o: clnt.c hint.h
    gcc $(OPT) -c clnt.c
dh.o: dh.c hint.h dh.h
    gcc $(OPT) -c dh.c
arith.o: arith.s
    as -o arith.o arith.s
clean:
    rm rsrvr.o asrvr.o clnt.o
    rm rs as c
```

CHAPTER 5
RESULTS AND DISCUSSIONS

5. Results & Discussions

For analyzing the critical timings, we have used the function `gettimeofday()` to find out the results for the transmission of packets. The function `gettimeofday()` displays current time after every 50th packet in seconds and microsecond. Thus we can calculate the time required to transmit 50 packets. (This time is included in the Time Stamp of the SRTP header). The main purpose was to calculate the timing difference between non-encrypted packets and the encrypted packets. Two scenarios were examined. In one scenario we full encryption and key generation was carried out. In scenario II, key generation was not done and the unencrypted audio packets were dispatched.

5.1 Timings for Encryption

When MMKM application (the name of our application) was sending **encrypted audio packets**, the time taken to transmit 50 packets was in order of 5-second i.e. the values returned by `gettimeofday` varied from 1132844404 to 1132844409 in seconds. We also noted the microseconds. We took last two digits of the seconds and all the digits of the microseconds to show the results.

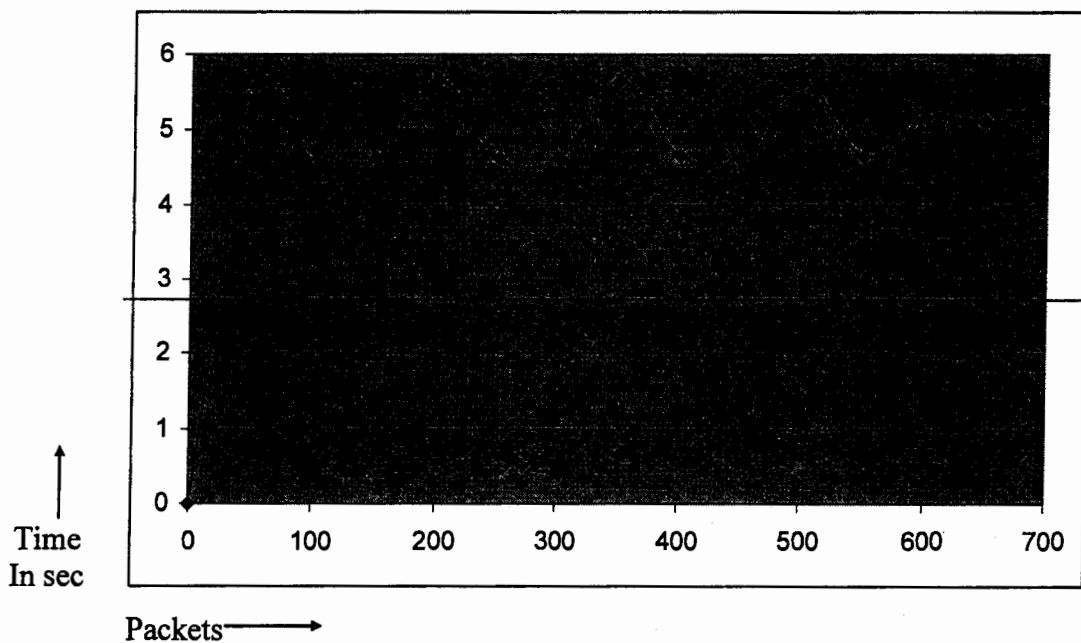


Fig. 5.1 Encrypted Audio Packets

When MMKM application was sending **audio packets without any key generation and without encryption**, the time varies in order of 4 to 5 sec and vice versa. However, the quality is a quite better.

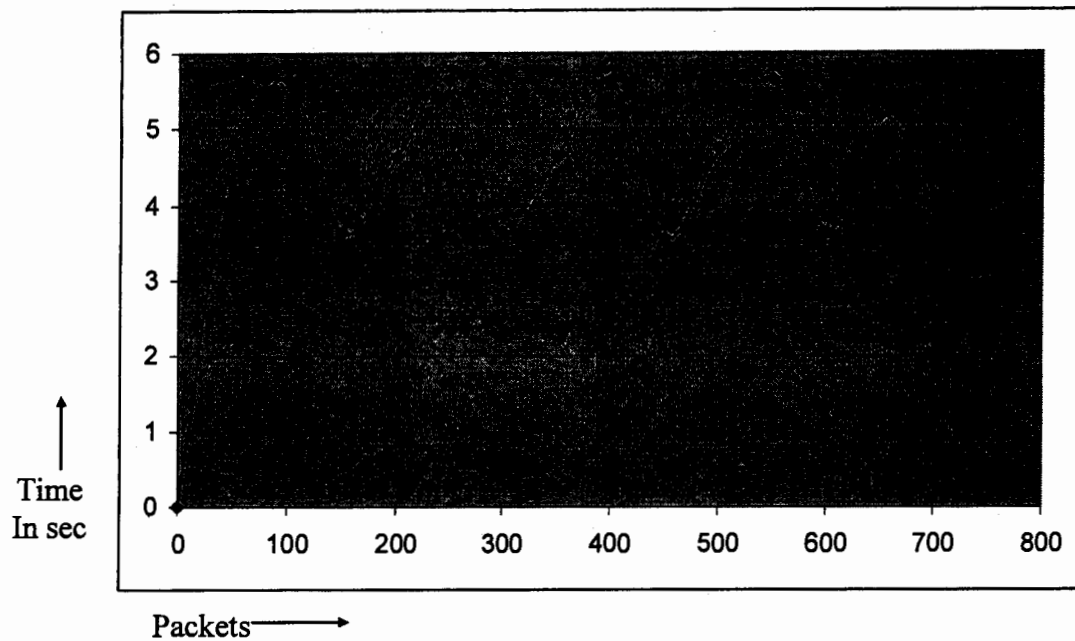


Fig. 5.2 Audio Packets without Encryption

5.2 Conclusions

The main Purpose of MMKM project is to develop an environment that performs secure multimedia streaming to multicast receivers, which prevent data hacking and eavesdropping security threat. We emphasize on security, as well as the sound quality.

SRTP & MIKEY has been used for audio streaming and data security. SRTCP has been used for control information of the packets [8, 9]. We found very little implementations of MIKEY and SRTP on the Internet. But in our application MMKM, the combination of the two has been implemented, which basically fulfills the security purpose.

Conclusions and Recommendations:

1. Discussion on RFCs
2. Discussion on buffers
3. Multithreaded application

1. MIKEY and SRTP RFCs are both related to each other in the security context. MIKEY is used for key management and SRTP manages the actual transmission. But their roles are not clearly defined in the RFCs [8, 9]. It causes implementation ambiguities for the users.

For example, MIKEY uses the words TEK (Traffic Encryption Key) and TGK (Traffic Generation Key), meaning that TGK is a kind of master key, from which we derive TEK. But it does not clearly define other keys. On the other hand, the SRTP discusses two keys, the Master Key and the Master Salt Key, leaving the choice with the user for the latter. MIKEY does not discuss Salt Keys. Yet, it derives other keys. It is not clear that the derivation of keys is only for initial security context parameters exchange or subsequently, it will also be used for the actual data transmission.

The confusion arises in deciding how do MIKEY and SRTP integrate with each other. We shall explain it further in subsequent parameters.

It is well understood that there will be two phases of any communication. In the first phase, we establish communication and in the second phase we transmit and receive the data. We have assumed that in the first phase, it is MIKEY which comes into play and in the next phase it is the SRTP that comes into play. Working on this assumption, first of all we should have a Pre-Shared key. This key will be considered as TGK for the purpose of MIKEY. With the help of this key we generate TEK. Thus we establish a session and exchange security parameters. Now we move on to second phase. Here we are entering the SRTP. The RFC says that we should derive six keys from our master key. Here we have three choices, i) we generate a random number and treat it as our Master key. ii) We use the pre-shared key as our master key that we had used for MIKEY, iii) we use Diffie- Hellman or Public key algorithms to generate the master key. The RFC does give a choice but is silent on this aspect. Let us assume that we generate a new key. Note, that if we adopt this solution, we had a pre-shared key we used it for MIKEY only, and generated a number of keys from that to be used in the phase when MIKEY is going on. Then we generated another master key and used it to derive further keys to carry on with our data transmission. So, in all we could end up with 12 keys, six for the MIKEY Phase and six for the SRTP (data transmission) phase. It makes too much overhead.

Our own suggestion is that we should use one key, which is either pre-shared key, or generated by Diffie-Helman or Public key algorithms and use it for both phases. It will reduce the overhead of calculation of keys and simplify implementation. We have used this concept in our implementation.

Another point is about Salt Keys. SRTP RFC leaves it to the choice of the user whether he/she wishes to generate the Salt Keys or not. One of the headers should specify whether Salt keys are being used or not, so that both sides (transmitter and receiver) should know whether salt keys are to be used or not. Similarly, if the Salt keys are to be used, the RFC does not clarify how many bits are to be changed at what frequency. The application has to

build it on its own choice. We feel, it is leaving too many things to the implantation. The result will be that an application designed by one will not be able to talk to the other.

In summary, the integration between SRTP and MIKEY should be more clearly defined. Moreover, use of Salt Keys should be further elaborated. At the end of the SRTP RFC, there is a relationship described between SRTP and MIKEY but it does not clarify the concepts completely.

2. It is well known that in case of Multimedia applications, we need to transmit data in real time. Buffering is one of the techniques that are used for this purpose. Normally, the receiving side creates a number of buffers and then stores the data into those buffers, before playing it. Thus when the data starts being played, at that time there are a number of buffers that have been filled with the data. Thereafter, the playing of the data and reception of the data go on simultaneously. The purpose of filling the buffers in the beginning is that in subsequent transmission, if there is a network delay (transmission errors or congestion) the data being fed to the playing device is available. Because there are number of buffers which are filled with data and they can continue to supply data for playing even when the reception of new data, from the network, is being delayed.

We had applied this concept in our application. But there was one problem. (This problem has already been described in the text above). The data had been sampled at 44100 samples per second. But the sound card, that we had, could play only 48000 samples per second. So, in spite of the fact that the buffers were full in the beginning, soon they would become empty and the data player had to wait for more data to arrive, before it could be played. We devised a novel solution for that.

We ran an artificial local client on the same machine (or on the same LAN) where our server was running. Thus the client and server were running on same machine. The local client was playing the data it had received. The client would request a packet, receive it and play it. Then it would request for the next packet, receive it and play it. There was no buffering involved. This process would go on, till the server would send zero byte packet to show that the data has ended ie one song has been played. This technique ensured that any other client (on the Internet or on LAN) could not play the data faster than the local client. This system addressed the problem of playing too fast by the client. This solution would work beautifully on LAN or WLAN where there were no network delays and no problems of bandwidth. But this solution had the limitation on the Internet. Because, it would not buffer the data, so if there were network delays, the quality of sound would suffer. Another advantage of this technique was that we would be locally monitoring the playing of song, which is essential to see what is being transmitted on the Internet.

We have used both methods in our applications. But we have mostly relied on request method instead of buffering.

Here is a graph which shows how quickly the buffers were depleted, while using buffering technique. We had used in total 50 buffers. When 45 buffers were full, on the client side, then the client would start playing the song. Thereafter, one thread would be receiving the data and re-filling the buffers while other thread was playing the data from the filled buffers. In the graph below, the wave file was sampled at 44100 samples per second, while the sound card was playing 48000 samples per second.

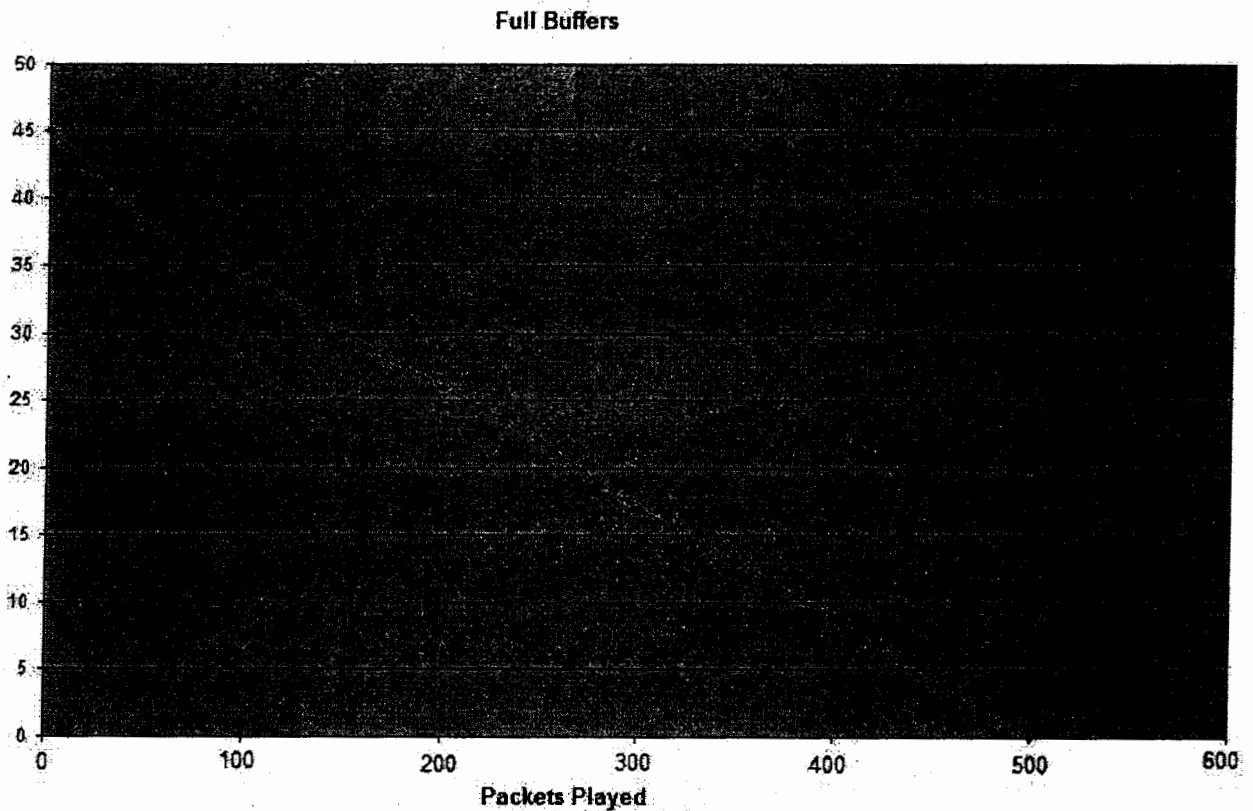


Fig. 5.3 Audio Packets depleted

The X axis shows the serial number of packet being played. The Y Axis shows the number of full buffers, when a particular packet was being played. Note that when we start playing (ie when serial number of packet played is zero) at that time, we had 45 full buffers. When we go on playing packets, the number of full buffers keeps decreasing. By the time we have played 100th packet, the full buffers left are only 34 instead of 45. Because data being played is faster than the data being received. And by the time we have played 200th packet, the full

buffers left are only 27. When we are playing 500th packet, at that there are is only one full buffer left. And very soon, all the buffers are exhausted and there is no filled buffer which can be played. This shows how fast the buffers are being depleted.

5.3 Access to Threads

The application is thread intensive and we noted that the quality of sound had no effect because of that. The threads used were:

- a. One thread is for sending wave data to all the clients. A separate thread was necessary for this because certain clients would join when the song has already been playing for a minute or two. If they wished to join, at this stage, the server should be able to provide them with the sound characteristics.
- b. The Thread for sending data of the song was the main work horse. It kept sending the data as multicast.
- c. The third thread was the one which was handling reports from the clients. It was deliberately slowed down, to avoid large volumes of data in the form of client reports. So, we have only allowed reports at 5 seconds intervals.
- d. Then was another thread which handles the song requests. When the user wishes to listen to a particular song, it sends that request and it is handled here.

The conclusion is that for a Multimedia application, we need to have a fast machine to be able to handle all these threads.

5.4 Future Enhancements

MMKM modules (RSRVR, ASRVR, and CLNT) are flexible enough to accommodate and incorporate multiple functions in it depending upon day to day needs and future aspects. Some possible enhancements could be

MP3 Audio Format

The effort will not be as much on compression of audio data but more on its security. So implementation comprises on wave (.wav) file format. It could be better to implement on mp3 format because minimal disk space utilization is preferred now a days.

Billing system

Billing System may maintain to enhance security. User will enter his/her credit card number. Credit card should be valid. If wrong credit card number will be entered by user then he/she will not be registered to get audio streams. Implementation of billing system can also eliminate the problem of stealing client software.

User Identification

A check can be implemented on user login that one user cannot login simultaneously on two machines. If one logout from one machine then he will again login to another machine. This user identification can work well in billing environment.

SDES SRTCP Packet

SDES stands for Source Description items. They contain information to describe the sources e.g. Canonical Name of the host (CNAME), User Name, User Email address, User Phone, Location and the Software tool used. Both Server and Client should generate SDES control packets.

In SDES packet, the IP address of the client is dynamically read by the server and display on screen. It does not hard code in the Client data file. It forms a security check whether the client is the same as he/she is showing through his/her data. It will be evident from the IP address, which will dynamically read by the Server from the UDP packets.

The SDES packet can be useful to get user information for authenticity in case of billing system. The database of billing system must have records of user information like email, phone etc. to make user trustworthy. SDES packet can handle like RR and SR packets. However, we did not emphasize on SDES because of the fact that it is not required in our project.

BIBLIOGRAPHY & REFERENCES

Bibliography & References

- [1] Gregory W. White, et. al, "Computer System and Network Security", CRC Press Computer Engineering, 1995.
- [2] Internet Multicast Security Overview of Issues and Technologies by Fern Levitt
- [3] Ericsson TD S3-020533, Security protocol.
- [4] Ericsson TD S3-020534, Key management.
- [5] <http://www.ietf.org/html.charters/msec-charter.html>
- [6] TD S3-030250, Status of SRTP and MIKEY in IETF, Ericsson
- [7] TD S3-030249, Key generation and distribution in MBMS, Ericsson
- [8] MIKEY: Multimedia Internet KEYing, Arkko et. al., Request for Comments: 3830, August 2004.
- [9] The Secure Real-time Transport Protocol, M. Baugher et. al., Request for Comments: 3711
- [10] TD S3-030368, Pseudo CR Introducing MIKEY and SRTP to TS 33.246, Ericsson
- [11] TD S3-030356, MBMS Security Framework, Qualcomm
- [12] 3GPP TSG SA WG3 Ad hoc (3 - 4 September 2003)- Introducing SRTP in TS 33.246
- [13] 3GPP TSG SA WG3 Security - S3#33 (10 - 14 May 2004)-LS on Protection of streaming and download MBMS data
- [14] 3GPP TSG SA WG3 Security - SRTP for streaming protection in MBMS
- [15] TD S3-030xxx: ME based MBMS key management with MIKEY, Ericsson, SA3#31, Munich
- [16] TD S3-030xxx: UICC based MBMS key management with MIKEY, Ericsson, SA3#31, Munich
- [18] 3GPP TSG SA WG3 Security - S3#31 (18 - 21 November 2003)- Migration of MIKEY in MBMS key management
- [19] Nokia, "S3-040081: Use of MIKEY in Combined method"

- [21] 3GPP TSG SA WG3 Security - S3#33 (10 - 14 May 2004)-Extension payloads to MIKEY to support MBMS
- [22] 3GPP TSG SA WG3 Security - S3#33 (10 - 14 May 2004) - MBMS key management with MIKEY
- [23] TD S3-040081, MIKEY in MBMS, SA2#32, Nokia
- [24] Internet Multicast Security, Overview of Issues and Technologies, Fern Levitt
- [25] Real Time Transport Protocol, Request for Comments: 3550
- [26] <http://en.wikipedia.org/wiki/AES>
- [27] <http://en.wikipedia.org/wiki/Diffie-Hellman>
- [28] <http://en.wikipedia.org/wiki/HMAC>
- [29] User datagram Protocol (UDP), Request for Comments: 768

APPENDIX A
GLOSSARY

Glossary

Authentication

The process of identifying an individual usually based on a user name and password. Authentication usually requires something a person has (such as key, badge or token), something a person knows (such as a password, ID number or mothers maiden name), or something a person is (represented by a photo, fingerprint or retina scan etc). When authentication requires two of those three things, it is considered strong authentication.

Block

Sequence of binary bits that comprise the input, output, State, and Round Key. The length of a sequence is the number of bits it contains. Blocks are also interpreted as arrays of bytes.

Block Ciphers

The block cipher operates on a group of bits (a "block") of a certain length, and trasmit whole blok of data in one go.

Client/Server

A network computing system in which individual computers (clients) use a central computer (server) for services such as file storage, printing and communication.

Cipher

A cipher is an algorithm for encryption and decryption. The exact operation of ciphers is normally controlled by a key some secret piece of information that customises how the ciphertext is produced.

Cipher Key

Secret, cryptographic key that is used by the Key Expansion routine to generate a set of Round Keys; can be pictured as a rectangular array of bytes, having four rows and N_k columns.

Ciphertext

Data output from the Cipher or input to the Inverse Cipher.

Crypto Master (CS)

It is the uni or bi-directional data stream(s), protected by a single instance of a security protocol. For example, when SRTP is used, the Crypto Master will often contain two streams, an RTP stream and the corresponding RTCP, which are both protected by a single SRTP Cryptographic Context.

Crypto Master Bundle (CSB)

It is the collection of one or more Crypto Masters, which can have common TGKs and security parameters.

Crypto Master ID

It is unique identifier for the CS within a CSB.

Crypto Master Bundle ID (CSB ID)

It is unique identifier for the CSB.

Data Security Association (Data SA)

The information for the security protocol, including a TEK and a set of parameters / policies.

Denial of Service Attack (Dos)

A type of attack aimed at making the targeted system or network unusable, often by monopolizing system resources.

Decryption

Decryption is the reverse process, recovering the plaintext back from the ciphertext. Enciphering and deciphering are alternative terms (sometimes considered desirable as 'encryption' also refers to certain burial practices).

Eavesdropping

It is type of attack in which an unauthorized party gains access to an asset (data).

Encryption

The original information which is to be protected by cryptography is called the plaintext. Encryption is the process of converting plaintext into an unreadable form, termed ciphertext, or, occasionally, a cryptogram.

Inverse Cipher

Series of transformations that converts cipher text to plaintext using the Cipher Key.

Jitter

It is an estimate of the statistical variance of the RTP data packet inter-arrival time. It is measured in timestamp units and expressed as a 32 bit unsigned integer.

MAC

MAC stands for Message Authentication Code. In general, a MAC can be thought of as a checksum for data passed through an unreliable (or more importantly, insecure) pipeline. A sender will typically generate a MAC code by first passing their message into some MAC algorithm. The sender will then send their message M with the MAC (M). The receiver can then generate their own MAC (M) and verify that MAC (M) sent by the receiver matches the MAC (M) they themselves generated.

Multicast Environment

In multicast scenarios, the sender is in charge of setting up the security, e.g., real-time presentations.

Packet

A unit of information formatted according to specific protocols that allow precise transmittal of data from one node in a network to another. Also called a datagram or a data packet, it contains two parts: a header and a payload. The header is like envelop, the payload is the contents. In Internet protocol, any message that is larger than 1500 bytes gets fragmented into packets for transmission.

Public key cryptography

It is also known as asymmetric encryption. Two separate key are used, one for encryption and second for decryption.

RTP

Real-time transport protocol (RTP) is an IP-based protocol providing support for the transport of real-time data such as video and audio streams.

RTCP

Real-time transport Control protocol (RTP) is the control packet of RTP. RTCP provides support for real-time conferencing of groups of any size within an internet. This support

includes source identification and support for gateways like audio and video bridges as well as multicast-to-unicast translators. It offers quality-of-service feedback from receivers to the multicast group as well as support for the synchronization of different media streams.

Salting key

A random or pseudo-random string used to protect against some off-line pre-computation attacks on the underlying security protocol.

S-box

Non-linear substitution table used in several byte substitution transformations and in the Key Expansion routine to perform a one for-one substitution of a byte value.

Security Attack

Any action that comprises the security of information owned by an organization.

State

Intermediate Cipher result that can be pictured as a rectangular array of bytes, having four rows and Nb columns.

Symmetric key cryptography

Symmetric key ciphers use the same key for encryption and decryption, or a little more precisely, the key used for decryption is "easy" to calculate from the key used for encryption. Other terms include "private-key", "one-key" and "single-key" cryptography.

Threads

If a process has more than one strand of execution, this means that this process has more than one threads running. Threads are the concurrent execution of more than one task simultaneously.

TEK Generation Key (TGK)

A bit-string agreed upon by two or more parties, associated with CSB. From the TGK, Traffic-encrypting Keys can then be generated without needing further communication.

Traffic-Encrypting Key (TEK)

The key used by the security protocol to protect the CS (this key may be used directly by the security protocol or may be used to derive further keys depending on the security protocol). The TEKs are derived from the CSB's TGK.

TGK re-keying

The process of re-negotiating/updating the TGK (and consequently future TEK(s)).

Unicast

The unicast scenario is the call between two parties, where it may be desirable that the security is either set up by mutual agreement or that each party sets up the security for its own outgoing streams.

APPENDIX B
DATA FLOW DIAGRAMS

Data Flow Diagrams (DFD)

B.1 DFD of MMKM Application

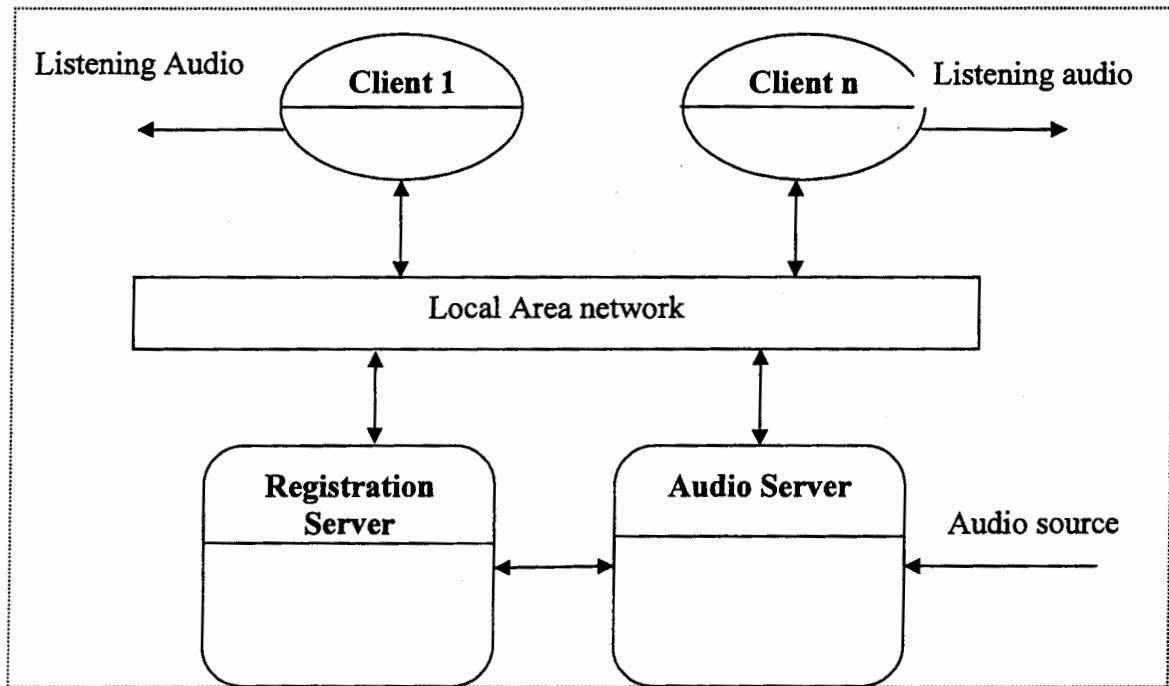


Fig. B.1 Context level data flow diagram

B.2 DFD for Registration Process

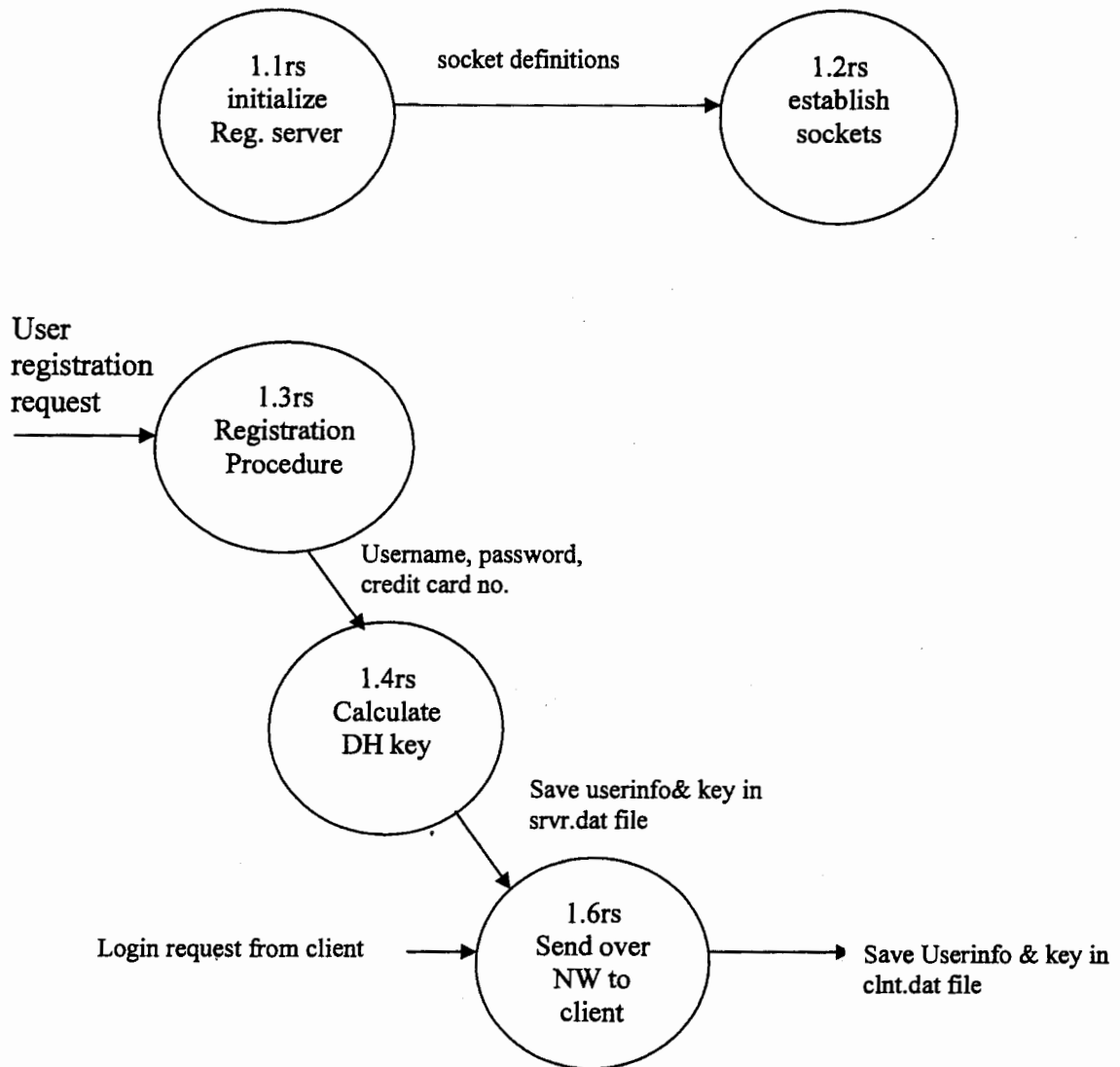
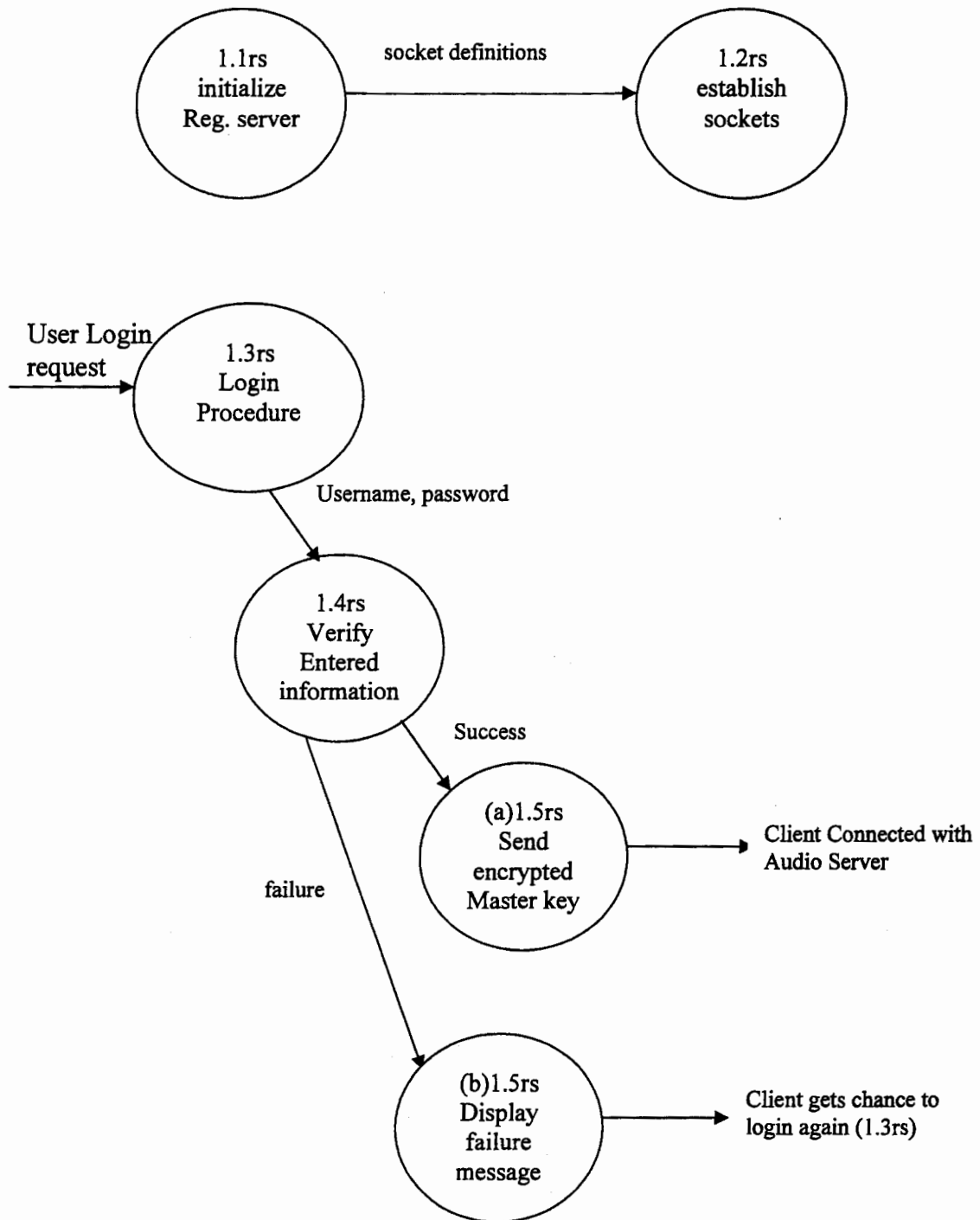


Fig. B.2 Level 1 DFD for Registration server computer – Registration Process

B.3 DFD for Registration server computer – Login Process**Fig. B.3 Level 1 DFD for Registration server computer – Login Process**

B.4 DFD for Registration server computer – Change password Process

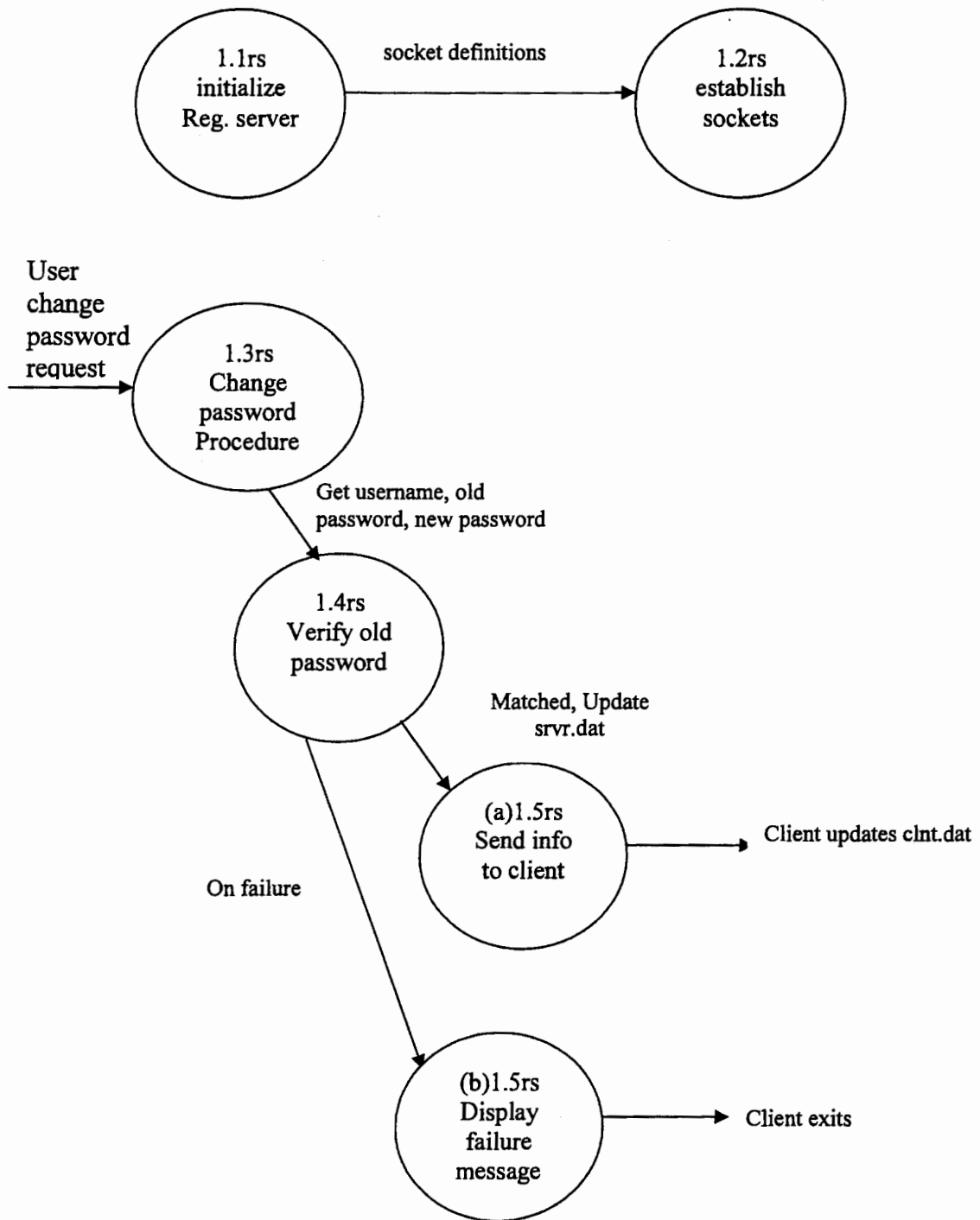


Fig. B.4 Level 1 DFD for Registration server computer – Change password Process

B.5 DFD for Audio Server computer – data sender

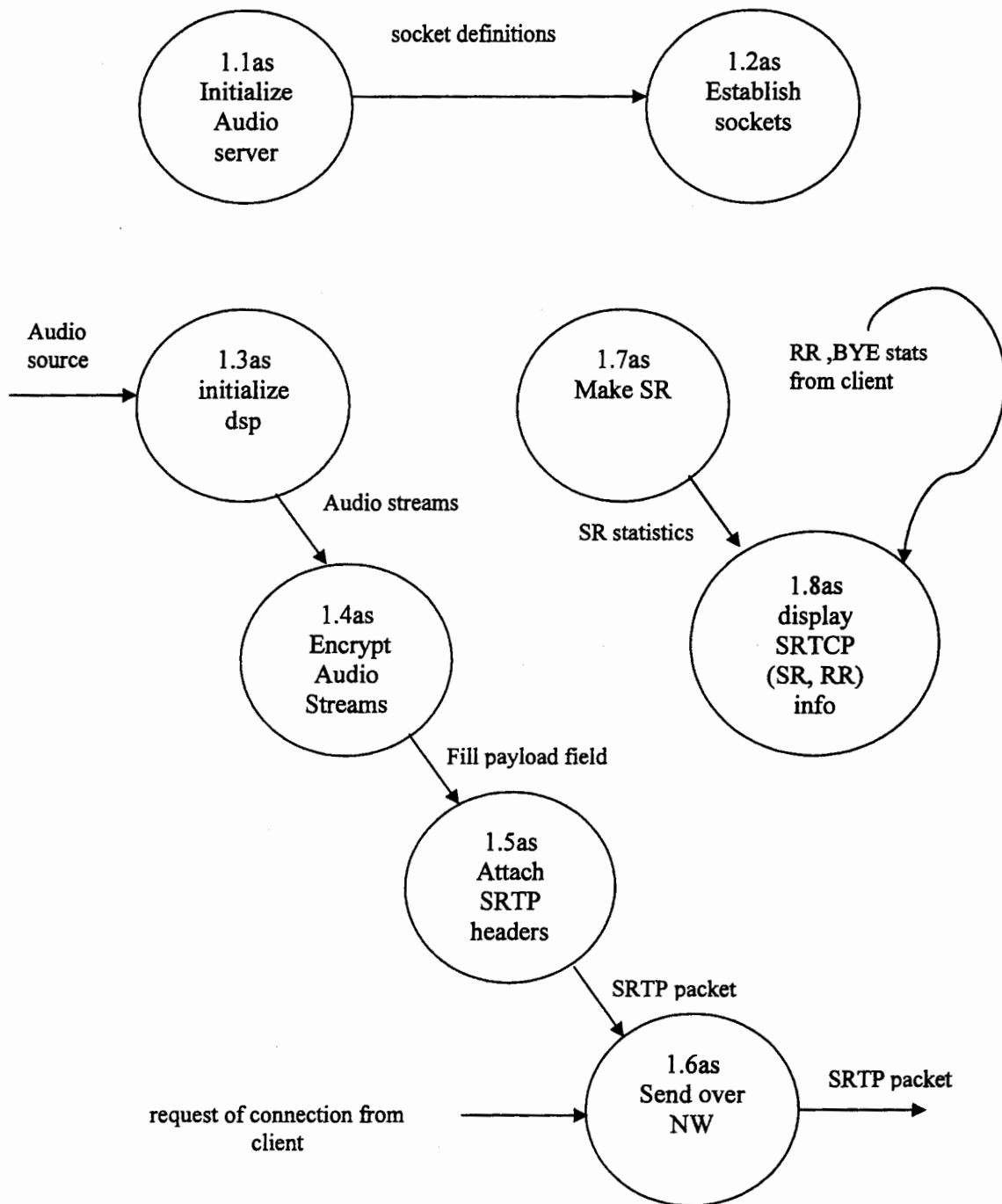


Fig. B.5 Level 1 DFD for Audio Server computer – data sender

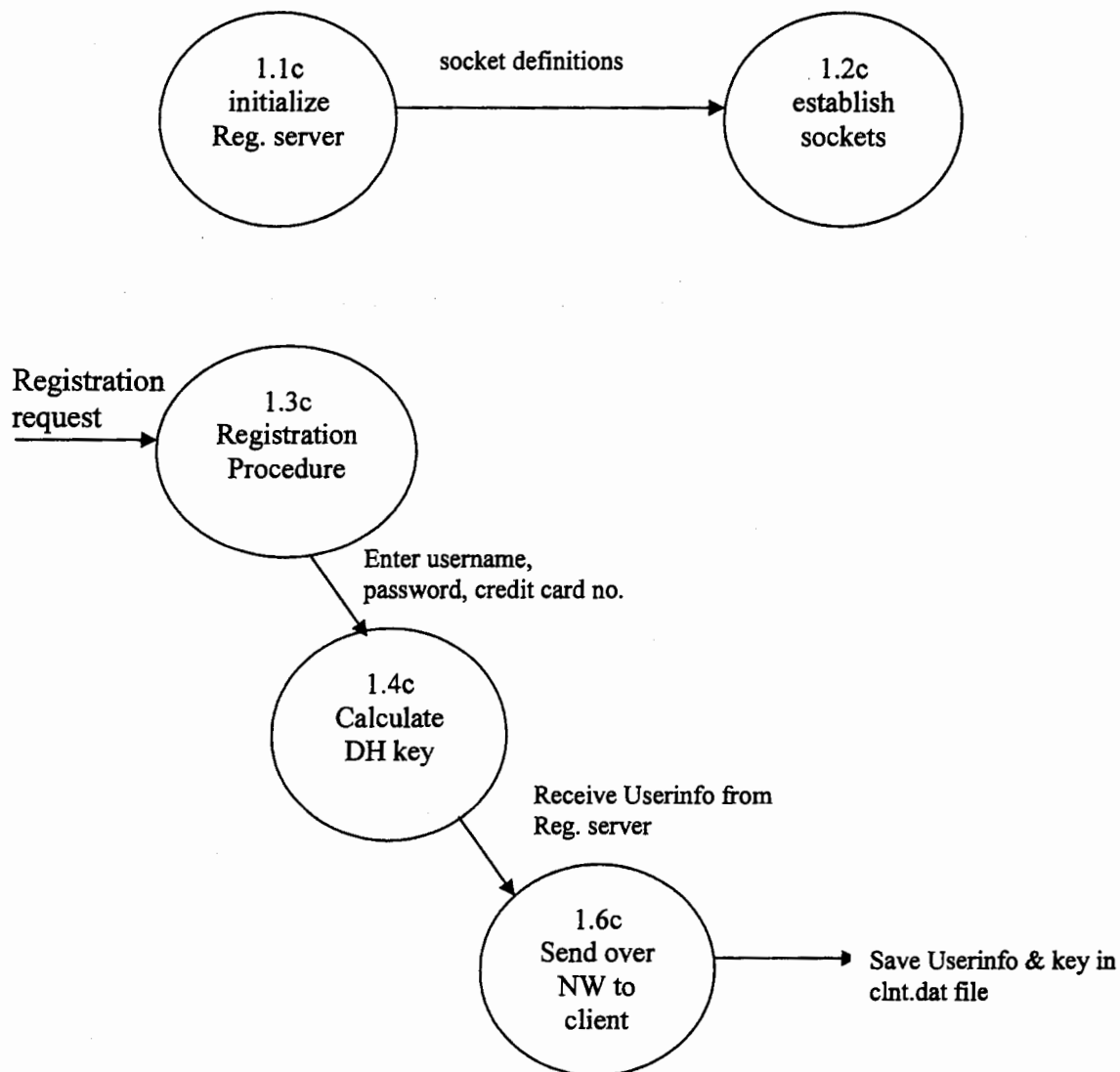
B.6 DFD for Client computer interact with Reg. server Registration Request

Fig. B.6 Level 1 DFD for Client computer interact with Reg. server Registration Request

B.7 DFD for Client computer interact with Reg. server Login Request

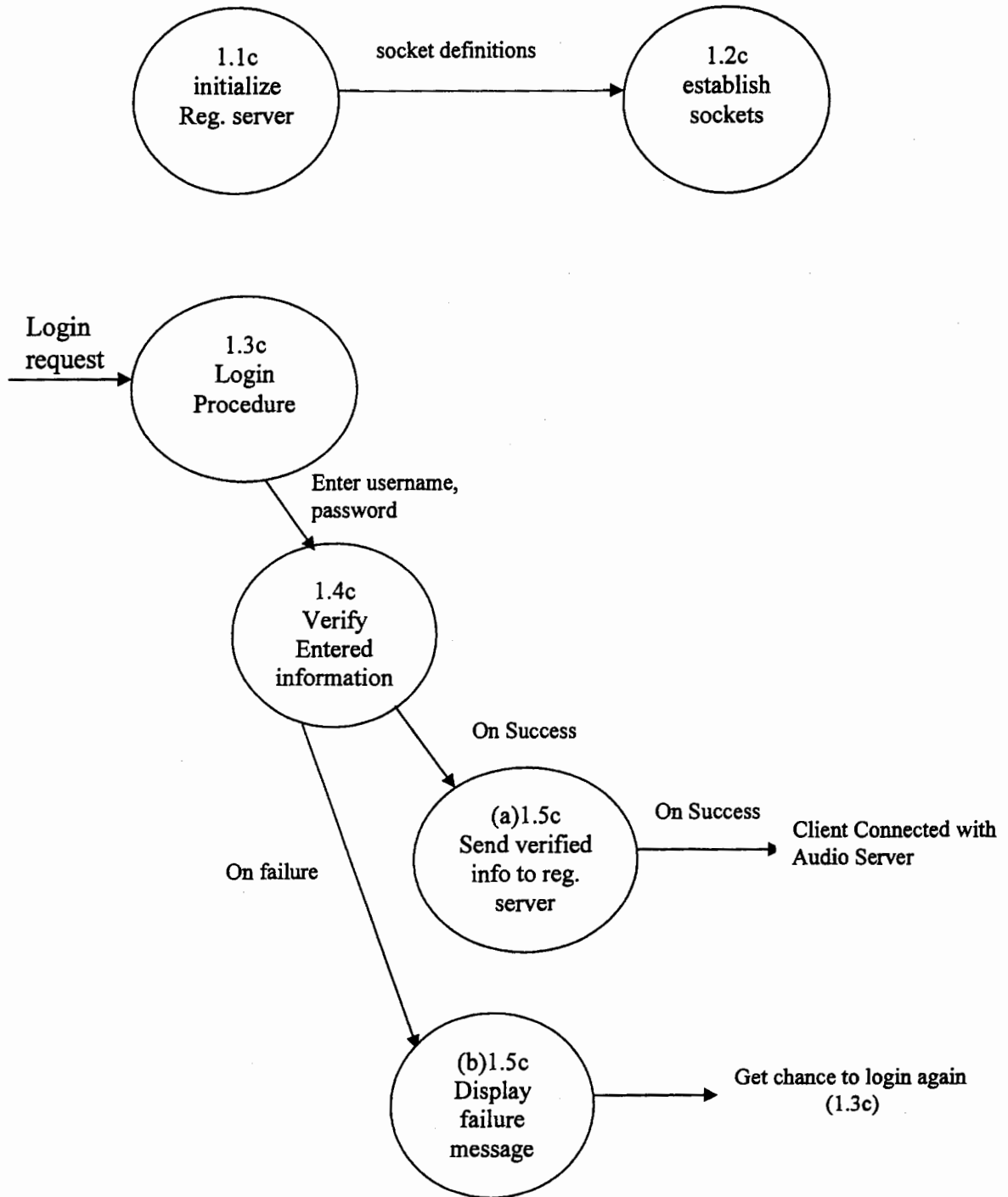


Fig. B.7 Level 1 DFD for Client computer interact with Reg. server Login Request

B.8 DFD for Client computer interact with Reg. server Change password Request

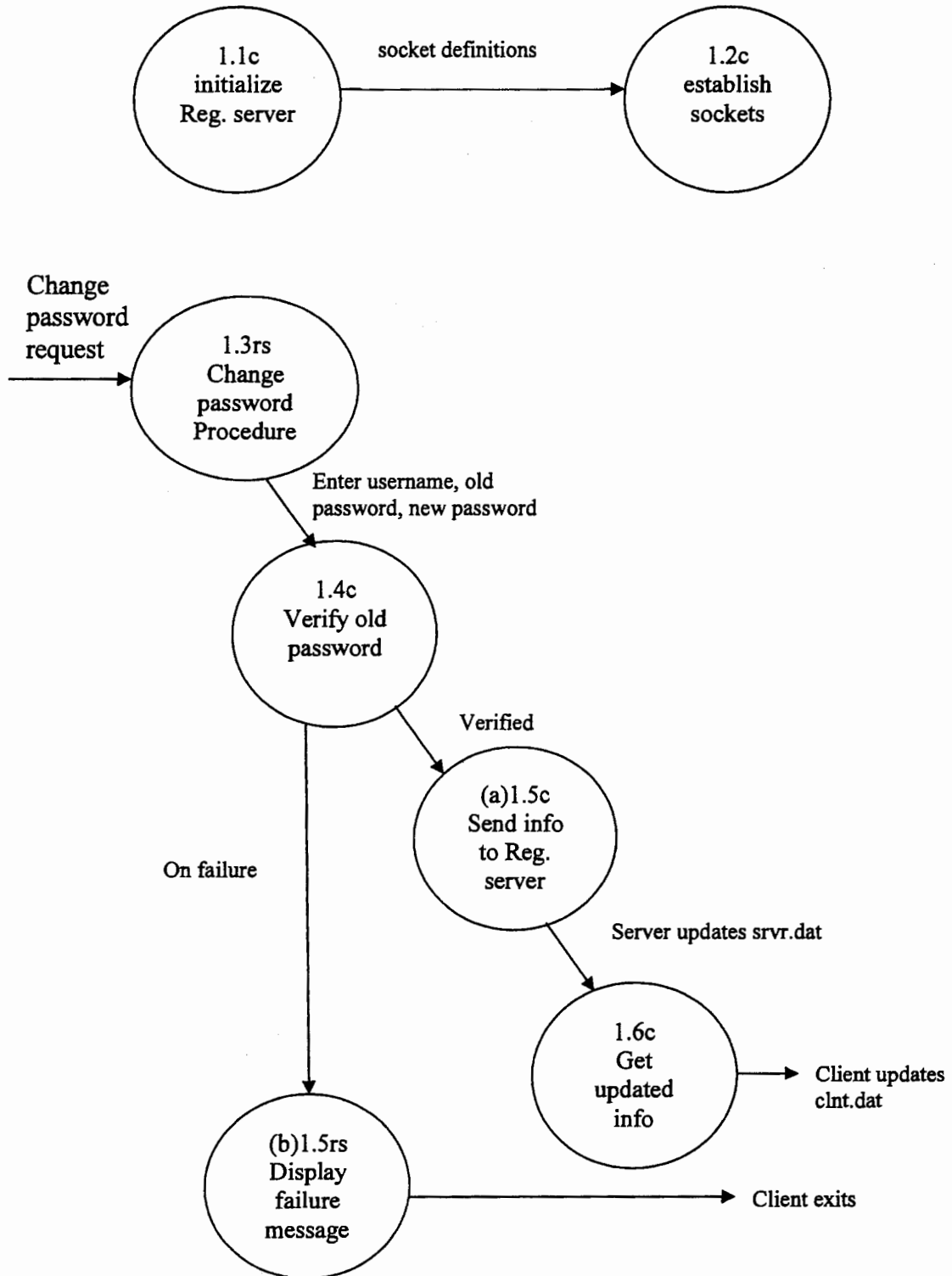


Fig. B.8 Level 1 DFD for Client computer interact with Reg. server Change password Request

B.9 DFD for Client computer interact with Audio server

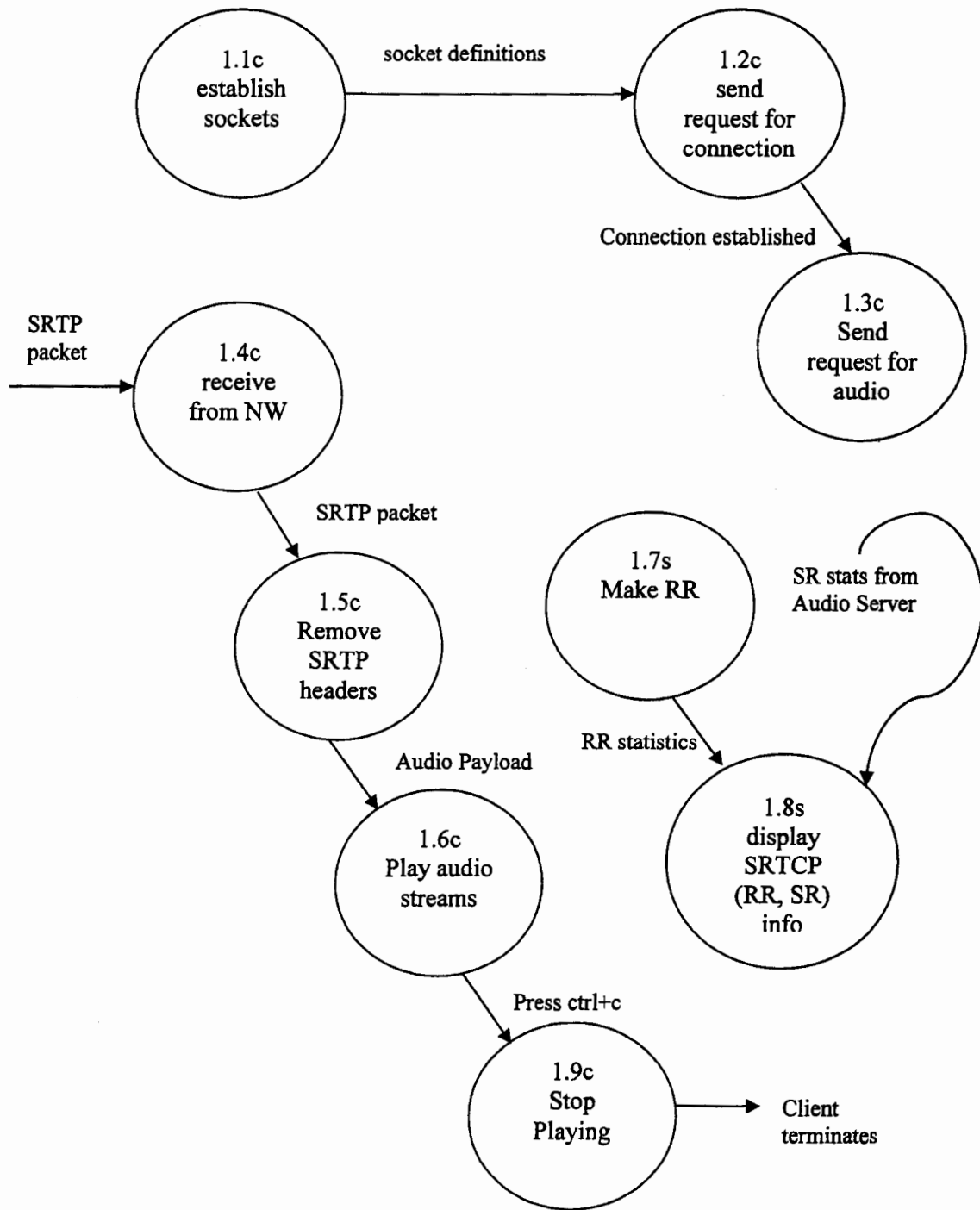


Fig. B.9 Level 1 DFD for Client computer interact with Audio server

APPENDIX C
LINUX/SOUNDCARD.H

Linux/Soundcard.h

/dev/dsp

`/dev/dsp` is the digital sampling and digital recording device, and probably the most important for multimedia applications. Writing to the device accesses the D/A converter to produce sound. Reading the device activates the A/D converter for sound recording and analysis.

The name DSP comes from the term digital signal processor, a specialized processor chip optimized for digital signal analysis. Sound cards may use a dedicated DSP chip, or may implement the functions with a number of discrete devices. Other terms that may be used for this device are digitized voice and PCM.

Some sound cards provide more than one digital sampling device; in this case a second device is available as `/dev/dsp1`. Unless noted otherwise, this device operates in the same manner as `/dev/dsp`.

DSP device

The DSP device is really two devices in one. Opening for read-only access allows you to use the A/D converter for sound input. Opening for write only will access the D/A converter for sound output. Generally speaking you should open the device either for read only or for write only. It is possible to perform both read and write on the device, albeit with some restrictions; this will be covered in a later section.

Only one process can have the DSP device open at a time. Attempts by another process to open it will fail with an error code of EBUSY.

Reading from the DSP device returns digital sound samples obtained from the A/D converter. Analog data is converted to digital samples by the analog to digital converter under control of the kernel sound driver and stored in a buffer internal to the kernel. When an application program invokes the read system call, the data is transferred to the calling program's data buffer. It is important to understand that the sampling rate is dependent on the kernel driver, and not the speed at which the application program reads it.

When reading from `/dev/dsp` you will never encounter an end-of-file condition. If data is read too slowly (less than the sampling rate), the excess data will be discarded, resulting in gaps in the digitized sound. If you read the device too quickly, the kernel sound driver will block your process until the required amount of data is available.

ioctl calls

The input source depends on the mixer setting (which I will look at shortly); the default is the microphone input. The format of the digitized data depends on which ioctl calls have been used to set up the device. Each time the device is opened, its parameters are set to default values. The default is 8-bit unsigned samples, using one channel (mono), and an 8 kHz sampling rate.

Writing a sequence of digital sample values to the DSP device produces sound output. Again, the format can be defined using ioctl calls, but defaults to the values given above for the read system call (8-bit unsigned data, mono, 8 kHz sampling).

If the data are written too slowly, there will be dropouts or pauses in the sound output. Writing the data faster than the sampling rate will simply cause the kernel sound driver to block the calling process until the sound card hardware is ready to process the new data. Unlike some devices, there is no support for non-blocking I/O.

If you don't like the defaults, you can change them through ioctl calls. In general you should set the parameters after opening the device, and before any calls to read or write. You should also set the parameters in the order in which they are described below.

All DSP ioctl calls take a third argument that is a pointer to an integer. Don't try to pass a constant; you must use a variable. The call will return -1 if an error occurs, and set the global variable `errno`.

If the hardware doesn't support the exact value you call for, the sound driver will try to set the parameter to the closest allowable value. For example, with my sound card, selecting a sampling rate of 9000 Hz will result in an actual rate of 9009 Hz being used.

If a parameter is out of range, the driver will set it to the closest value (i.e., the upper or lower limit). For example, attempting to use 16-bit sampling with an 8-bit sound card will result in the driver selecting 8 bits, but no error will be returned. It is up to you, the programmer, to verify that the value returned is acceptable to your application.

About MMKM Audio Player

The .wav file MTM player works in a way that client requests multicast server to play a wav file. Server opens a file and first send the packets of 44 bytes to the client, these 44 bytes are used to initialize the `/dev/dsp` device. After the initialization of sound device Server sends the packets of 16384 bytes to the client until the end of file. Client receives the packets one by one and plays the file until the last packet sent by Server.

DSP Capabilities of MMKM Audio Player

Displaying information about a DSP device (/dev/dsp by default). Determining DSP Capabilities

1. bits (depth) (SOUND_PCM_WRITE_BITS)
2. Channels (number of channels) (SOUND_PCM_WRITE_CHANNELS)
3. rate (sampling rate) (SOUND_PCM_WRITE_RATE)
4. blocksize (block size) (SNDCTL_DSP_GETBLKSIZE)

The above mentioned all capabilities vary according to the different sound devices.

Typical **output** from the MTM Player program looks like this:

Information on /dev/dsp:

Defaults:

sampling rate: 8000 Hz

channels: 1

sample size: 8 bits

block size: 16384 bytes

Modes and Limits

Device	Sample Size	Minimum Rate	Maximum Rate
1	8	8000	16000
2	8	8000	16000

Format of WAV File

The format for a wave file is as follows:

Offset	Description
0x00	chunk id 'RIFF'
0x04	chunk size (32-bits)

0x08	wave chunk id 'WAVE'
0x0C	format chunk id 'fmt '
0x10	format chunk size (32-bits)
0x14	format tag (currently pcm)
0x16	number of channels 1=mono, 2=stereo
0x18	sample rate in hz
0x1C	average bytes per second
0x20	numbers of bytes per sample
0x22	numbers of bits in a sample
0x24	data chunk id 'data'
0x28	length of data chunk (32-bits)
0x2C	Sample data (remaining whole file extending over many kilo bytes)

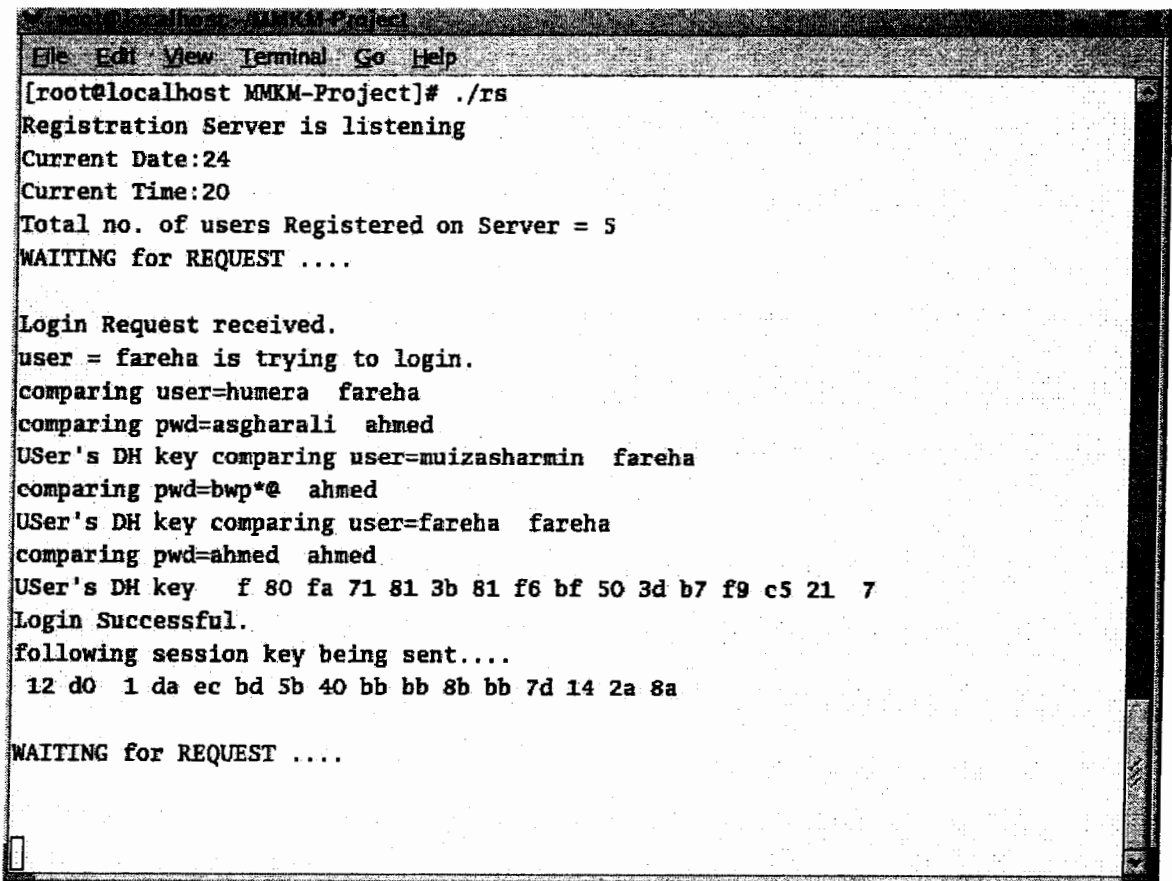
APPENDIX D
USER MANUAL

User Manual

D.1 Registration Server (RSRVR)

RSRVR starts running with following output displayed on screen. Output shows active status of RSRVR with Current time and date and also number of already registered user. Server is waiting for client request to process his/her information.

Login Request is received by RSRVR from client. In reply of client request RSRVR sends Master key to client for further processing.

A terminal window titled "MMKM-Project" with a menu bar "File Edit View Terminal Go Help". The terminal shows the execution of the command './rs' at the prompt '[root@localhost MMKM-Project]#'. The output is as follows:

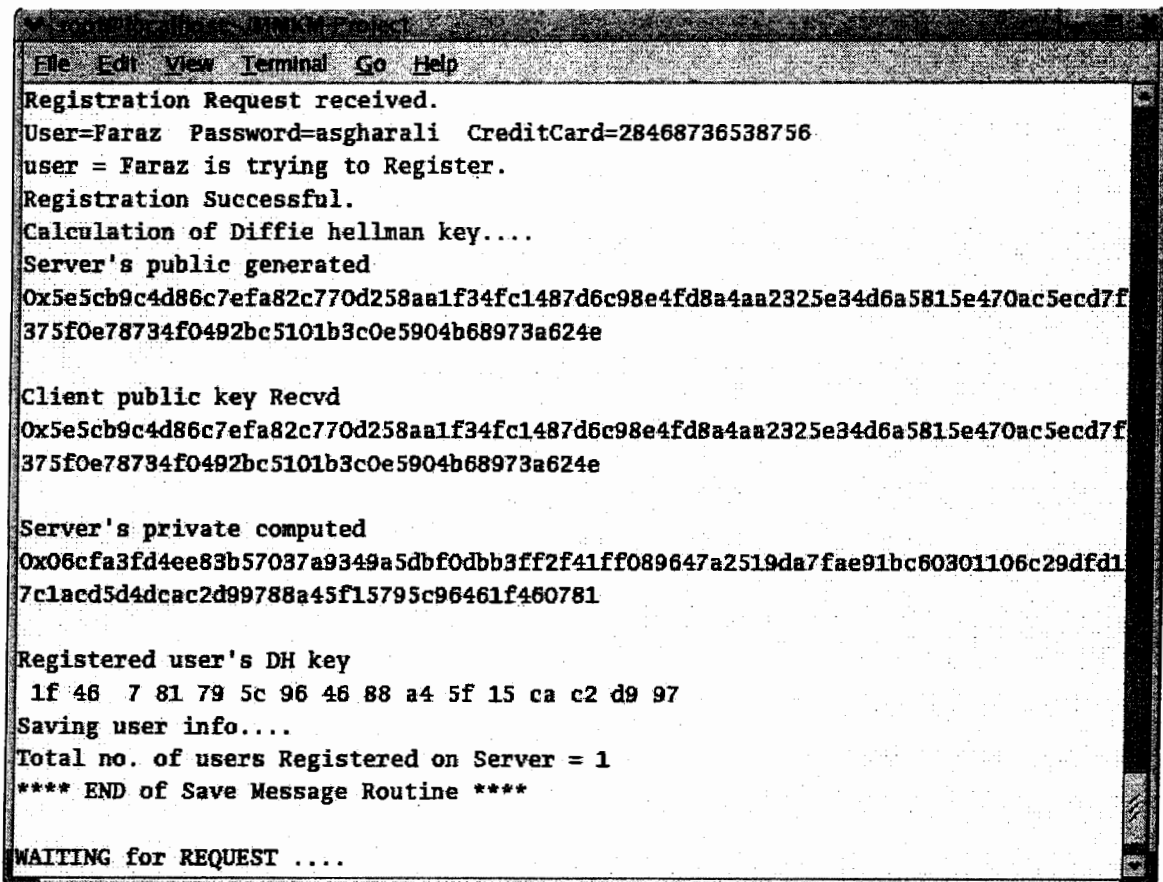
```
[root@localhost MMKM-Project]# ./rs
Registration Server is listening
Current Date:24
Current Time:20
Total no. of users Registered on Server = 5
WAITING for REQUEST ....

Login Request received.
user = fareha is trying to login.
comparing user=humera fareha
comparing pwd=asgharali ahmed
USER's DH key comparing user=muizasharmin fareha
comparing pwd=bwp*@ ahmed
USER's DH key comparing user=fareha fareha
comparing pwd=ahmed ahmed
USER's DH key f 80 fa 71 81 3b 81 f6 bf 50 3d b7 f9 c5 21 7
Login Successful.
following session key being sent....
12 d0 1 da ec bd 5b 40 bb bb 8b bb 7d 14 2a 8a

WAITING for REQUEST ....
```

Fig. D.1 Login Request processed by RSRVR

Registration is request received by RSRVR from client. RSRVR generate Diffie-Hellman key and save user information in srvr.dat file.



```
File Edit View Terminal Go Help
Registration Request received.
User=Faraz Password=asgharali CreditCard=28468736538756
user = Faraz is trying to Register.
Registration Successful.
Calculation of Diffie hellman key....
Server's public generated
Ox5e5cb9c4d86c7efa82c770d258aa1f34fc1487d6c98e4fd8a4aa2325e34d6a5815e470ac5ecd7f
375f0e78734f0492bc5101b3c0e5904b68973a624e

Client public key Recvd
Ox5e5cb9c4d86c7efa82c770d258aa1f34fc1487d6c98e4fd8a4aa2325e34d6a5815e470ac5ecd7f
375f0e78734f0492bc5101b3c0e5904b68973a624e

Server's private computed
Ox06cfa3fd4ee83b57037a9349a5dbf0dbb3ff2f41ff089647a2519da7fae91bc60301106c29dfd1
7c1acd5d4dcac2d99788a45f15795c96461f460781

Registered user's DH key
1f 46 7 81 79 5c 96 46 88 a4 5f 15 ca c2 d9 97
Saving user info....
Total no. of users Registered on Server = 1
**** END of Save Message Routine ****

WAITING for REQUEST ....
```

Fig. D.2 Registration Request processed by RSRVR

D.3 Client Software (CLNT)

CLNT software starts running with Login Window. Already registered user must enter his/her username and password for further processing.

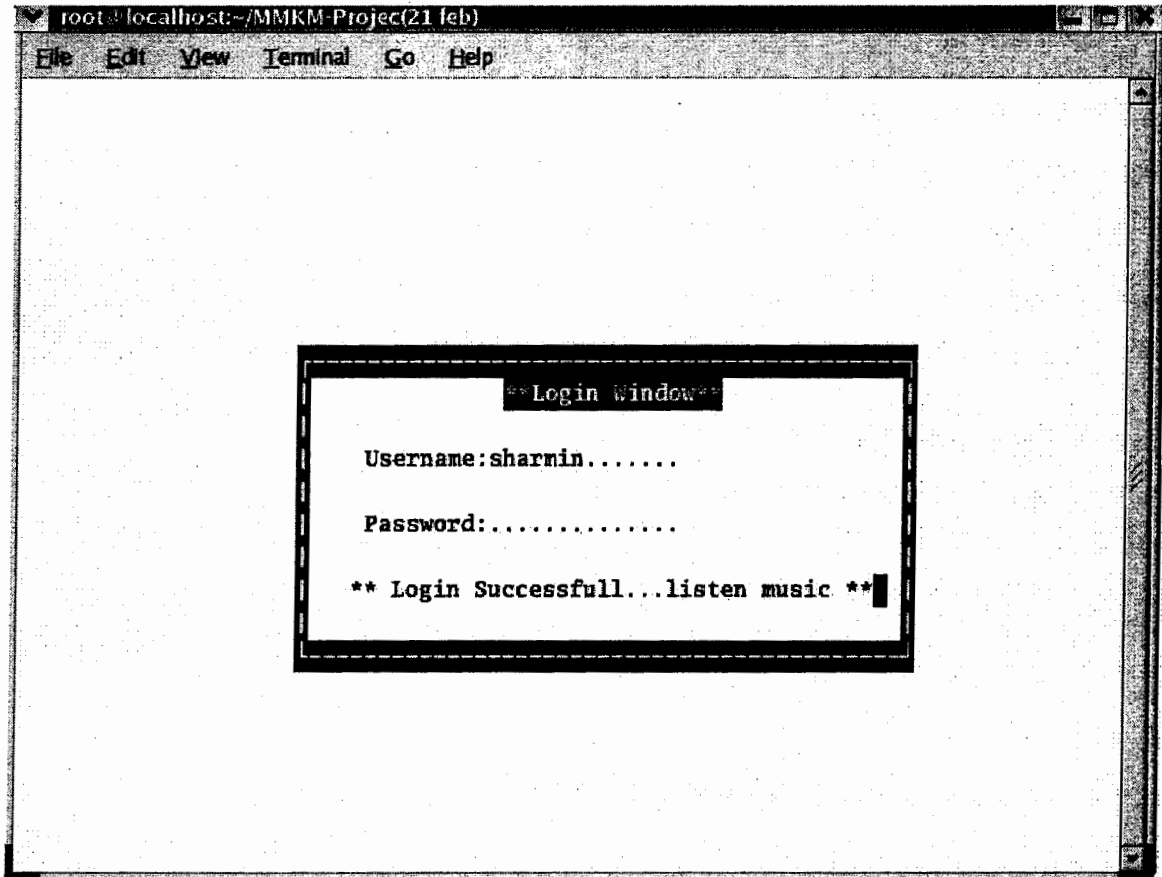
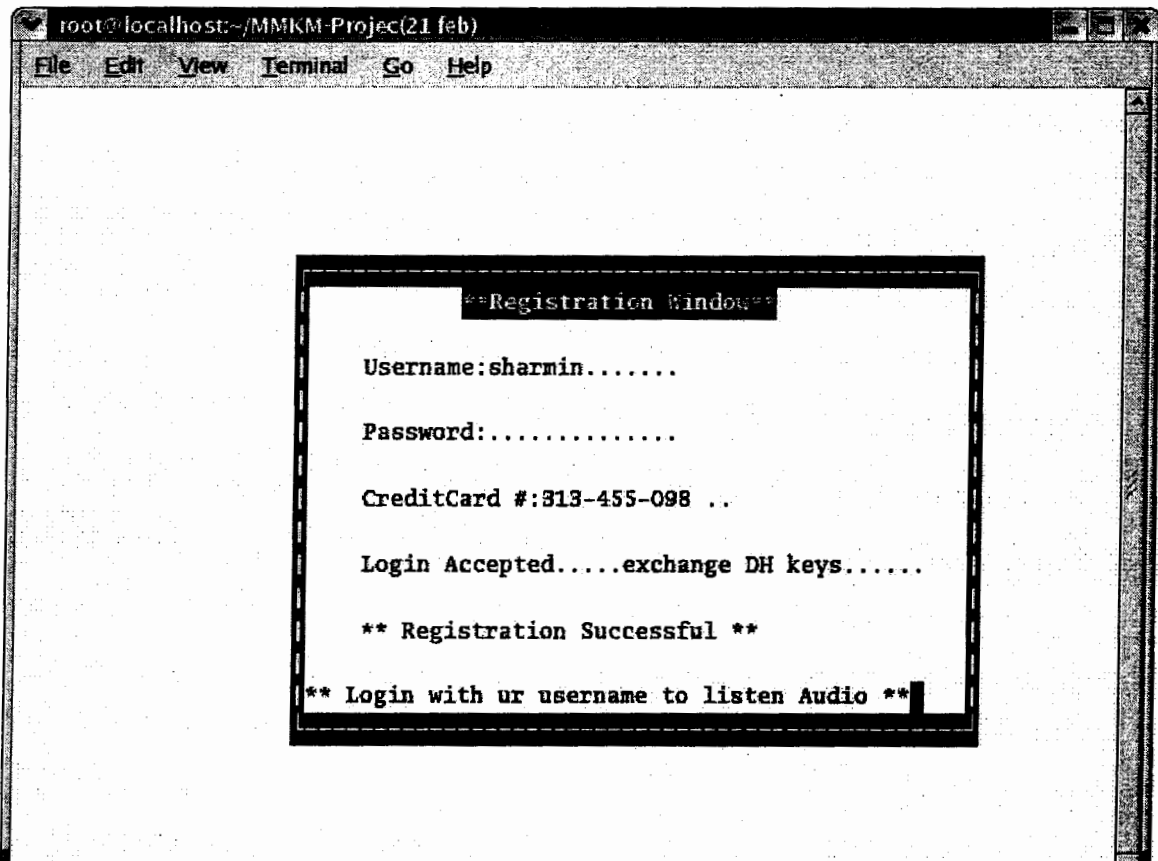


Fig. D.4 Login Request processed by CLNT

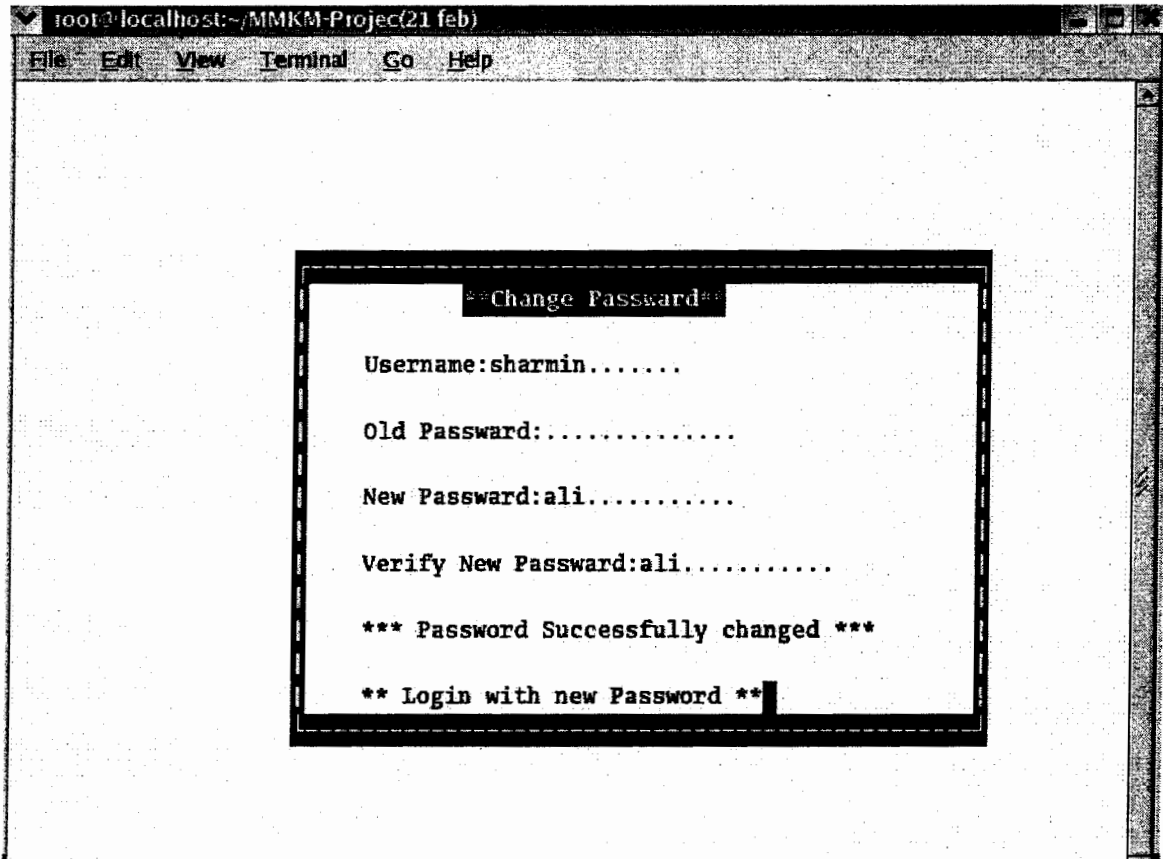
CLNT software registered new users by getting their username, password and credit card number. On success response from RSRVR CLNT generates Diffie-Hellman key and save all information in clnt.dat file.

A terminal window titled 'root@localhost:~/MMKM-Projec(21 feb)' with a menu bar containing 'File', 'Edit', 'View', 'Terminal', 'Go', and 'Help'. The terminal displays a registration process. A dashed box highlights the following text:

```
**Registration Window**  
  
Username:sharmin.....  
Password:.....  
CreditCard #:313-455-098 ...  
Login Accepted.....exchange DH keys.....  
  
** Registration Successful **  
  
** Login with ur username to listen Audio **
```

Fig. D.5 Registration Request processed by CLNT

CLNT software gives chance to user to change his/her password. Following output appears on screen on success. This mechanism updates both `svr.dat` and `clnt.dat` by RSRVR and CLNT respectively.



```
root@localhost:~/MMKM-Projec(21 feb)
File Edit View Terminal Go Help

**Change Password**

Username:sharmin.....
Old Password:.....
New Password:ali.....
Verify New Password:ali.....

*** Password Successfully changed ***

** Login with new Password **
```

Fig. D.6 Change Password Request processed by CLNT

Final output screen of CLNT displays audio player properties, SRTP and SRTCP packets statistics like SR, RR, BYE packets.

```

root@localhost:~/MMKM-Projec(21 feb)
File Edit View Terminal Go Help

** Song List **
0-track1.wav
1-track2.wav
2-track3.wav
3-track4.wav
4-track5.wav
Enter Song Serial #: 1
You have chosen 1
Choice Queue # 4

** Player **
** Initialize dsp
initial bytes=44
depth = 16
chanel = 2
Sample Rate = 44100
bsize = 16384
Receiving Audio data.
loop=163

** SRTP Header **
Version=2
Padding=0
No sources=4
Payload type=3
Start Seq=41360
ts=513858
Source=414091151
Mac= 24 b2 35 df

** SRTCP RR Report **
Count=6=2
Payload type=201
Last Seq=41350
Jitter=149747
Last SR=5
RR Index=6
RR Mac= 31 c8 c1 8b
BYE Packet 203 Sent

** SRTCP SR Report **
Payload type=200
NTP ts=1140663074
RTP ts=508089
Count=6
Octet Count=120
SR Index=6
SR Mac= b3 bd 23 a1

```

Fig. D.7 SRTP & SRTCP Packets Statistics on CLNT

D.4 Installation Process

The whole application is available in the form of **MMKMproject.tgz**. Follow the following procedure on both server and client, turn by turn:

- i. `tar xzvf MMKMproject-1.2.tgz`

It will create a directory with the name project on your hard disk.

- ii. Move to the directory.
- iii. Type `make` and hit `<enter>`. It will compile the programs. It will compile three programs, registration server (RSRVR), Audio Server (ASRVR), client (CLNT) and few others.
- iv. Copy these binaries to `/usr/local/bin`

D.5 Running Process

- i. Make sure Sound card is installed on the server as well as on client, the utility is provided on Linux to check soundD.
- ii. For Registration Server, run `./rs`. It will start waiting for connection from the client. As soon as Client sends registration, login, change password request. Reg. server responds according to request.
- iii. On the Audio server, run `./as`. It will start waiting for connection from the client. As it will get audio streams request it starts playing audio streams to connected client and also will start sending and receiving SRTP and SRTCP reports and will show it on server screen as well as on client screen in multicast environment.
- iv. On the client machine, run `./c` for normal login. This run command is for already registered users. It will make initial connection with server and also verify that the server is running. The audio streams list will be displayed on client side to get a desired choice for audio file.
- v. On the client, Type audio file choice number and hit `<enter>` and then sit back and listen the audio files according to queue maintained on Audio server.

D.6 Commands Available During Runtime

Following events are defined:

a. Starting initial application – see above “Running Process”.

b. On The Client

For Registration

Type `./c -R` on command prompt and hit <enter> It will register a new client on reg. server by getting username, password and credit card information from client.

For Login

Type `./c` on command prompt and hit <enter> It will ask for username and Password which has been created on the time of registration.

For Changing Password

Type `./c -P` on command prompt and hit <enter> it will change the previous password of registered user.

Press `ctrl+c` It will quit the application.

On The Registration Server

Type `./rs` on command prompt and hit <enter> It will run the reg. server and it will be in waiting state to entertain user request of registration, login and changing password.

Press `ctrl+c` It will quit the application.

On The Audio Server

Type `./as` on command prompt and hit <enter> It will run the Audio server and it will be waiting to send secure audio streams to client.

Press `ctrl+c` It will quit the application.

APPENDIX E
RESEARCH PAPER



EuroJournals, Inc.
AMS Publishing, Inc (Austria)
P.O. Box 1212
Vienna

Tel: 0033 1222 32231
Fax: 0033 1222 32232
Web: www.eurojournals.com
E-mail: editor@eurojournals.com

Ref: 0038
Date: 30.06.06

Qaisar Javaid
Department of Computer Science Faculty of Applied Sciences
International Islamic University Islamabad, Pakistan

Dear Dr Javaid,

This is to inform you that based on the feedback from two referees, your article titled: MIKEY and SRTP Integration for Multicast Streaming (by Mamoona Asghar, Moizza Sharmin, Shiraz Baig, Qaisar Javaid and Khalid Rashid) has been accepted for publication in the next issue of European Journal of European Journal of Scientific Research. Please do not hesitate to contact me should you require further assistance.

Adrian Marcus Steinbreg



Adrian Marcus Steinbreg, PhD
Manager,
EuroJournals, Inc.

MIKEY and SRTP Integration for Multicast Streaming

Mamoona Asghar
mamoona16@gmail.com

Moizza Sharmin
muiza_khan@yahoo.com

Shiraz Baig
shiraz_baig@yahoo.com

Qaisar Javaid
qiccie@yahoo.com

Prof. Dr. Khalid Rashid
drkhalid@iiu.edu.pk

*Department of Computer Science, Faculty of Applied Sciences,
International Islamic University, Islamabad.*

Abstract

This paper presents the design and implementation of key management architecture for secure multimedia audio streaming to multicast receivers. It describes the methods and techniques which are used in making a multicast multimedia streaming application (MMKM). It also highlights the security issues involved, implementation of Multimedia Internet Keying (MIKEY), integration with Secure Real time Transport Protocol (SRTP) and then using it for transmitting real time multimedia data in Multicast environments. The main emphasis has been on security rather than on data transmission and compression. In case of multimedia, only audio data is chosen.

Keywords: SRTP, MIKEY, AES, Diffie-Hellman, HMAC, Multicasting, Cryptography, Security, Confidentiality, Authentication, Multimedia.

1. Introduction

There has been work to define a security protocol for the protection of real-time applications running over RTP [1]. However, a security protocol needs a key management solution to exchange keys and related security parameters. The focus is on how to set up key management for secure multimedia sessions such that requirements in a heterogeneous environment are fulfilled. MIKEY describes a key management solution that addresses multimedia scenarios for unicast and Multicast environment. [2]

SRTP provides a framework for encryption and message authentication of RTP and RTCP streams. It defines a set of cryptographic transforms with appropriate key management for unicast and multicast RTP applications. [3]

Although SRTP and MIKEY have been discussed in detail in their respective RFCs and other related documents. Some contributions found related to

working of both Protocols to make secure multimedia transmission in Unicast environment but there is little on their integration and then implementation for secure Multicast transmission.

Multicast is inherently a receiver-based concept. The sender is not consulted about the addition of a multicast receiver. Receivers can join and/or leave a particular multicast session, whether or not it is currently active, at will. [4] However, this aspect has been addressed in Multicast Multimedia Key Management (MMKM) application and we have made arrangements to make up for this shortcoming by maintaining a list of receivers.

We believe that Diffie-Hellman (DH) key agreement method is the most secure method among the three methods of distributing keys namely pre-shared, public key & Diffie-Hellman. This needs a separate discussion, as to why it is the most secure method. However, one point may be highlighted that two parties agree upon a shared secret key in such a way that the key is unavailable to eavesdroppers. [5] We have used this technique in MIKEY, making data transmission safer.

DH is basically a peer to peer technique [2], but MMKM application implements it in one to many scenarios. This is one of the unique features of this study.

We have developed a workable application which not only evades many security threats (discussed in section 2), but also integrates SRTP and MIKEY, successfully manages multicast transmission of multimedia and gives a method of implementation for future enhancement and application development. There are number of options and choices, laid down in their RFCs. MMKM application and this paper make those decisions and lays down a clear path of implementing transmission of multicast multimedia.

2. Threat Model

The threat model that is perceived and implemented in our application is discussed below:

2.1 Passive Attacks

Traffic generation Key (TGK) has been generated through Diffie-Hellman technique. It never travels on the network, so there is no possibility of its interception. Then Traffic Encryption Key (TEK) also called Master key is generated. Advance Encryption Standard (AES) [6] is used to encrypt this key and transport it across the receiver, using DH key. If an un-authorized interceptor captures the key he/she cannot use it. 128 bit AES encryption is used, and we rely on the strength of un-breakability of this encryption scheme in a finite time.

The data is encrypted with an Encryption Key of 128 bit, which has been generated from Master Key in a secure cryptographic manner. Both Master and Encryption keys are modified after every 24 hours. Thus we assume that we are safe against interception of TGK or TEK and data eavesdropping threat.

2.2 Active Attacks

We believe that Diffie-Hellman key is not compromised, as it has never traveled on the network, that's why it prevents impersonation.

To avoid man in the middle attack, an authentication key is generated for the process of authentication. If hacker captures encryption key, then he must capture authentication key to launch this attack. We believe that this is extremely difficult and this attack cannot be launched. Authentication also detects the security threats related to unauthorized replay, deletion, insertion and manipulation of messages. A mechanism has been incorporated in the application to drop adulterated packets.

The denial of Receipt attack prevention is tackled in a way that the receiver keeps sending a message after a fixed interval (10 seconds) and it verifies that he/she is receiving the data.

2.3 Theft of Data

The user has been given a password and login for its authentication. If user gives his/her login and password to his/her friend, it is possible that data

theft takes place. This is basically a billing issue. We expect that a person will not willingly give his /her login and password to an irresponsible person if the data is very confidential. But for the purposes of billing a friend might like to oblige the other friend. We have not kept a self destruct mechanism in implementation. There is no check, if same username logs in on two machines. This check can better work with billing system. However, in this case, the billing agency does not entail a loss, as the party is still paying for the time that is being used by his/her friend.

3. Proposed Architecture

MMKM Architecture is designed to resolve the basic and advance problems of Multicast scenarios. The basic features of multicasting are joining and/or leaving the group by users while the advance features are efficiency in key delivery, error robustness, load distribution, extra network administration, and join/leave notification. In multicast environment the data need to shift from one member to another to travel in network that make the data vulnerable because the attacker can easily attack at several points in the network at the same time. We shall rely on conventional means to cater for this threat.

Proposed MMKM architecture consists of two servers and multiple clients. The application has following three modules;

- Registration Server (RSRVR)
- Audio Server (ASRVR)
- Client (CLNT)

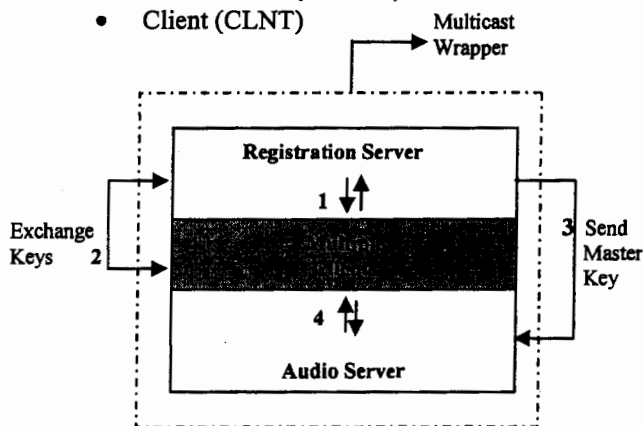


Fig. 1 MMKM Architecture
Communication flow of MMKM components

When any of multicast receiver wishes to listen real time file after joining the group, first the client communicates with Audio Server, then audio server communicates with Registration server for receiving Master keys to establish session among clients and audio server. Moreover the proposed solution introduces a counter to limit the number of audio requests from clients in a multicast group. The MMKM architecture provides complete authentication to registered members which automatically provides security and better Quality of Services to users.

It also maintains a list of join and/or leave notification of all registered users according to their joining and/or leaving time.

4. Implementation

The implementation scenario has three phases. First MIKEY is used for key management, key distribution and identification of party through digital signatures. Second phase encompass of the encryption and authentication of SRTP packets. Final phase is the transmission and playing of real time audio data. A number of practical issues like request of audio file from the recipients, password authentication and transmission of multimedia file with its characteristics i.e. bit depth, bitrate, channels are tackled during implementation. Multithreaded application is employed for this purpose. Two servers are used, one for key management and the other for handling of audio transmission.

4.1 Key Terminology and Role of MIKEY viz SRTP

It is considered prudent to clarify certain terminology of the keys, because SRTP and MIKEY uses different terms for certain keys which perform same functions. MIKEY used the terms of TEK and TGK. It distributes either a TEK or TGK. If it distributes TGK, then it is used to generate TEK. If it distributes TEK, security protocol (SRTP) directly uses it. The SRTP calls the TEK of MIKEY as its Master Key. The Master key is then used to generate six Session keys. These keys are Encryption Key, Authentication Key and Salting Key on the sender side and same three keys on the receiver side,

making a total of six keys for SRTP packets, while six same keys are also generated for SRTCP packets.

MIKEY is being used to generate a TGK. This TGK is used as TEK and is the Master Key for the SRTP. The Master Key is transported to the receiver side in a cryptographically secure manner. In this way, the same Master Key is available with sender and receiver to generate further keys.

4.2 Multimedia Internet Keying (MIKEY)

A Registration server (RSRVR) is constantly listening to requests of a new user who wishes to register, by sending a registration request. The request follows the pattern of MIKEY, in which digital certificate and other relevant information is provided. A user name and login is accepted and then a Diffie-Hellman key agreement takes place. A 72 byte (288 bit) string is generated, a random number is chosen and the algorithm is applied. After completion of one round of transmission, a TGK is generated. Then server uses timer as a seed, and generates a 16 byte (128 bit) random number, which is treated as TEK or Master Key. This key is encrypted with DH and transmitted to the receiver. Thus TGK and TEK have been generated and distributed. The TEK is now available for generating the six keys used by SRTP.

MIKEY deals with three types of scenarios, peer to peer, one to many, and many to many. On the other hand, MIKEY has three techniques that it uses, namely Pre-shared, Public Key, and Diffie Hellman. The DH technique is mostly used in peer to peer scenario. But we have used it for one to many scenarios. This is how we do it. Each receiver exchanged DH key parameters with the RSRVR at the time of registration. Both parties, then, arrive at a final key, which is symmetric in nature. A database is maintained at the RSRVR. The key of each user is stored in the database along with its username. The server generates a TEK which is same for every user and transfer to the receiver.. At the time of login, the DH key and TEK both are used. This TEK is then used by the receiver to generate his/her three keys, which are subsequently used by SRTP and SRTCP.

4.3 Secure Real time Transfer Protocol (SRTP)

The TEK is now available to SRTP. The SRTP uses various algorithms to derive the keys. Key derivation reduces the burden on the key establishment. Six different keys for both SRTP and SRTCP are needed per crypto context. All these are derived from a single Master key by using Keyed-Hashing for Message Authentication Code (HMAC) Method. [7] The Pseudo Random Function (PRF) used is described in the following table.

<u>Parameter</u>	<u>Technique Uses</u>
a. TGK Generation	Diffie Helman
b. TEK Generation	HMAC-SH1
a. Derivation of Encryption Key	HMAC-SH1
b. Derivation of Authentication Key	HMAC-SH1
c. Derivation of Salting Keys	HMAC-SH1
d. Encryption of Data	AES_CM
e. Master key length	128 bits
f. Encryption (Session key length)	128 bits
g. Authentication (Session key length)	160 bits
h. Master salt key	112 bits
i. Session salt key length	112 bits
j. Key derivation rate	0
k. Key Life Time	24 hours
l. MKI indicator	0

4.4 Data Encryption

We have used a sound file for the streaming media. The server would open the file and advertise 44 bytes of the wave file characteristics. Then it would go on transmitting the packets, in a multicast manner. These packets would be released at fixed intervals. All the clients who are listening will receive the audio file characteristics and audio file packets. SRTP packets will be encrypted and transported. These packets will be decrypted and played on each client site. SRTCP packets would also be transmitted to generate specific reports. All these transmissions are being handled in different threads, and on different socket.

4.5 Other Services

Furthermore, there is a need that when a client leaves, i.e. sends a BYE packet, it must be recorded

at the ASRVR. Normally, in multicast environment, we cannot keep a record, of who is coming and going. [4] We were required to keep a record of that. Each client, after login, sends "I am alive" message after every 10 seconds. If for six consecutive turns, this message is not received, it is assumed that the client has died. Of course, if the client leaves in the normal manner, it will send a BYE packet and the server would come to know that a particular client has left.

Another facility of Audio files queue has been provided. In this case, a client, after login, gets a list of songs available with the server. The client can choose any file of his choice and send that choice to the server. The request will join the queue and the file will be played on its own turn.

5. Results

The function `gettimeofday()` is used to determine the critical timings of the transmission of every 50th packet in seconds and microseconds (This time is included in the Time Stamp of the SRTP header). The main purpose is to calculate the timing difference between non-encrypted packets and the encrypted packets. Two scenarios are examined, which are as follows

5.1 Scenario I

In this case full encryption of data packets and key generation is carried out. When MMKM application transmits encrypted audio packets, there is 5 seconds delay to transmit every 50th packet i.e. the values returned by function `gettimeofday` varies from 1132844404 to 1132844409 in seconds. The last two digits of the seconds and all the digits of the microseconds have been taken to show the results.

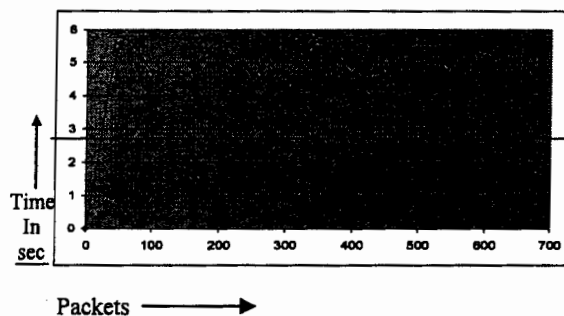


Fig. 2 Encrypted Audio Packets

5.2 Scenario II

In this case no key generation is done and the non-encrypted data packets are dispatched. When MMKM application sends audio packets without any key generation and without encryption, the time varies in order of 4 to 5 seconds and vice versa. However, the quality does not suffer.

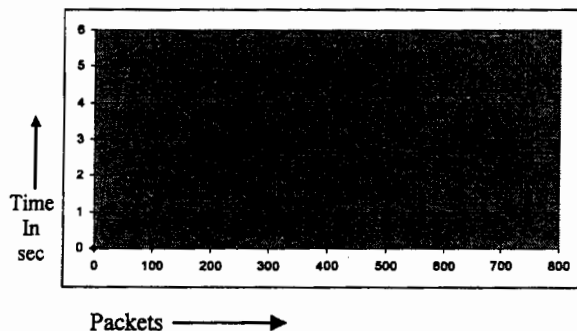


Fig. 3 Encryption without Audio Packets

Generally, it is considered best to use buffering technique in the multimedia applications. What we have done, we fill 50 buffers each of 16384 bytes and then start playing it. It find that after some time (less than a minute) all the buffers would become empty and playing would have to wait to fill in the buffers again. Thus the audio streams does not play continuously, it stops at intervals. This is an undesirable situation. It happens in certain audio streams while other audio file plays fine. Later we found, that the audio streams, which have been sampled at 48000 per second, at the record time, played fine during playtime. While, the audio streams that are sampled at 44100 per second, at record time, stops at intervals during playtime. This problem is briefly discussed below.

Generally Bytes per second (BPS) being played can be calculated by the formula:

$$\text{Sample Rate} * \text{Depth} * \text{channels} / 8$$

In case of 48000 sample rate the BPS is

$$48000 * 16 * 2 / 8 = 192000 \text{ BPS}$$

While with 44100 sample rate the BPS is

$$44100 * 16 * 2 / 8 = 176400 \text{ BPS}$$

Thus in every second we will be losing

$$192000 - 176400 = 15600 \text{ BPS}$$

Because the bytes would be playing much faster i.e. at 48000 sample rate, while the bytes are being dispatched by server at a slower rate i.e. 44100 sample rate. The fifty buffers, that we have used, contained $50 * 16384 = 819200$ bytes. In every second, we would be losing 15600 bytes. So, we would lose all the bytes in $819200/15600 = 52.5$ seconds. All 50 buffers would become empty in this time and the playing machine will have to wait to receive further data. There is no remedy to this problem except that we should use a better sound card.

To overcome buffering problem we use an artificial local client, which is running with ASRVR on same machine. The client is working on the principle that it sends request for every audio packet to play and also uses to monitor the transmission and playing of audio session. There is no buffering process involved. This technique ensures that any other client (on the Internet or on LAN) can not play the data faster than the local client. This system addresses the problem of playing too fast by the client. This solution will work beautifully on LAN or WLAN where there are no network delays and no problems of bandwidth. This solution has the limitation on the Internet, because it will not buffer the data. If there are network delays, the quality of sound will suffer.

6. Conclusion

The emphasis of this paper is to develop an environment that performs secure multimedia streaming to multicast receivers, which prevent security threats and also provides better quality of services.

MIKEY and SRTP RFCs [2, 3] are both related to each other in the security context, but their roles are not clearly defined in the RFCs, which causes implementation ambiguities.

MIKEY uses the words TGK and TEK for keys but does not clearly define other session keys. On the other hand, SRTP discusses two keys, the Master Key and the Master Salt Key. MIKEY does not discuss Salt Keys. Yet, it derives all the keys, but it is not clear that the derivation of keys is only for

initial security context parameters exchange or subsequently, it will also be used for the actual data transmission. The confusion arises in deciding how do MIKEY and SRTP integrate with each other.

The RFC of SRTP says that we should derive six keys from our master key. We have three choices, i) we can generate Master key by using random number, pre-shared key or Diffie-Hellman. The RFC does provide a choice but is silent on this aspect. According to RFCs we could end up with 12 keys, six for the MIKEY Phase and six for the SRTP (data transmission) phase. It produces too much overhead. We suggest that one key either pre-shared key or generated by Diffie-Hellman or Public key algorithms should be use and employ it for both phases. It will reduce the overhead of calculation of keys and simplify implementation. This concept has been used in our application.

Another point is about Salt Keys. SRTP RFC leaves it to the choice of the user whether to generate the Salt Keys or not. Similarly, if the Salt keys are to be used, the RFC does not clarify how many bits are to be changed at what frequency. In summary, the integration between SRTP and MIKEY should be more clearly defined.

7. Future Enhancements

MMKM modules (RSRVR, ASRVR, and CLNT) are flexible enough to accommodate and incorporate multiple functions in it depending upon day to day needs and future aspects. Some possible enhancements can be the use of MP3 Audio Format instead of .wav file. It can be better to implement on mp3 format because minimal disk space utilization is preferred now a days.

Billing System may be maintained to enhance security by using credit card number. Implementation of billing system can also eliminate the problem of "stealing" client software, even from a friend.

8. References

- [1] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, Request for Comments: 3550, July 2003
- [2] Arkko et. al., MIKEY: Multimedia Internet KEYing., Request for Comments: 3830, August 2004.
- [3] M. Baugher et. al.,SRTP: The Secure Real-time Transport Protocol, Request for Comments: 3711, March 2004
- [4] Fern Levitt, Internet Multicast Security, Overview of Issues and Technologies, Doc No. NU-R111, Rel. C, Feb. 8, 1999
- [5] E. Rescorla, Diffie-Hellman Key Agreement Method, Request for Comments: 2631, June 1999
- [6] NIST, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 26, 2001, <http://www.nist.gov/aes/>.
- [7] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", Request for Comments: 2104, February 1997.