

Fall 2012

Secure Media Streaming for Android

Jaie Patil
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects



Part of the [Computer Sciences Commons](#)

Recommended Citation

Patil, Jaie, "Secure Media Streaming for Android" (2012). *Master's Projects*. 272.

DOI: <https://doi.org/10.31979/etd.bnd2-9e6f>

https://scholarworks.sjsu.edu/etd_projects/272

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Secure Media Streaming for Android

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Jaie Patil

December 2012

© 2012

Jaie Patil

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Secure Media Streaming for Android

by

Jaie Patil

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2012

Dr. Mark Stamp Department of Computer Science

Dr. Chris Pollett Department of Computer Science

Ms. Nishtha Patel Infostretch Corporation

ABSTRACT

Secure Media Streaming for Android

by Jaie Patil

Digital media distribution systems must protect the confidentiality and integrity of content and ensure its authenticity, without introducing excessive overhead. Often, it is desirable to sacrifice some degree of cryptographic security for improved performance.

In this project, we have implemented a secure streaming server for the Android platform. We consider various encryption strategies for streaming video and analyze the tradeoff between security and efficiency.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to Dr. Mark Stamp for being my advisor and guiding me throughout my project. I would also like to thank my committee members Dr. Chris Pollett and Ms. Nishtha Patel for accepting to be on the committee and providing necessary input regarding the project. Also, I am indebted to the university and its faculty members without whom this project would not be possible.

I would also like to thank my friend Sarvesh Sharma for his valuable guidance throughout this project.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
1.1	Previous Work	2
1.1.1	Adaptive Streaming	2
1.1.2	Secure Streaming	2
1.2	Contributions	4
2	Background	6
2.1	Video Streaming	6
2.1.1	Streaming Principle	6
2.1.2	Video Streaming Types	7
2.2	Streaming Process	7
2.3	AES for Encryption	8
2.4	RSA for Authentication	9
3	Web Server and Database	11
3.1	Web Server	13
3.2	Admin Panel and its Features	14
3.2.1	Manage Users	15
3.2.2	Manage Videos	16
3.3	Database	16
4	Android Application	22
4.1	User Registration	22

4.1.1	User Login \Sign In	24
4.2	Video Streaming	24
5	Encryption and Decryption	27
5.1	.3gp video file format	28
5.2	A Complete Video Encryption method using AES	30
5.3	Our approach of encryption using random block method	31
5.3.1	Random Block Video Encryption (RBVE) Algorithm	32
5.4	Video Up-Loader	34
5.5	Video Decryption	34
6	Analysis of Encryption Schemes	35
7	Conclusions and Future Work	44

LIST OF FIGURES

1	Streaming Process	8
2	Project Infrastructure	12
3	Admin Panel Home Screen	15
4	Admin Panel Manage Contents	15
5	Manage Users	16
6	Manage Videos	17
7	User Schema	19
8	Video Schema	20
9	User Registration	23
10	Password Validation	23
11	User Login Screen	24
12	Media Streaming Welcome Screen	25
13	Video Streaming	25
14	Tabular Flow of User Login and Video Streaming	26
15	Video Encryptor and Uploader	28
16	General Details of the .3gp file	29
17	Video Details of the .3gp file	29
18	Audio Details of the .3gp file	30
19	Video Encryption Process	33
20	Video Upload Process	34
21	Comparison of Video Encryption Algorithms	38

22	Tabular view of comparison of Encryption Timings	39
23	Graphical representation of comparison of Encryption Timings using CVE and RBVE	39
24	Tabular view of comparison of Decryption Timings	40
25	Graphical representation of comparison of Decryption Timings using CVE and RBVE	40
26	Comparison of Encryption Timings	42
27	Comparison of Decryption Timings	43

CHAPTER 1

Introduction

For many decades, video has been an important medium for communication and entertainment. Streaming visual data to various users is becoming popular, thus protecting transmitted data from security threats has become one of the main concerns both for end users and data providers.

As the Internet and media are developing rapidly, the availability of media streaming applications is increasing. IPTV, video conferencing and distance education are some of the examples of these media streaming applications [17]. In media streaming, as soon as the file starts downloading, the end user can start viewing the file; however, this process leads to security flaws and needs to be managed carefully when valuable multimedia assets are hovering over the network [5].

Encryption and authentication, both play a vital role in the security of media streaming applications. These two factors meet three information security requirements of confidentiality, authenticity, and non-repudiation [17]. When media content is encrypted, only authorized users can access the protected content; this is called the confidentiality of the media [11]. Authenticity means both the sender and the integrity of the receiving media content is trustworthy. Non-repudiation ensures that the sender cannot deny the action of sending the media [11].

The Internet, combined with smart phones and its peripherals, gives rise to unlimited mobile possibilities [4]. One of these possibilities is explored and exploited by capturing video on a mobile device, in real-time, which is then viewed by the entire world. Currently, streaming video from a mobile phone is limited to commercial prod-

ucts [4]. Our paper states: how streaming can be achieved in an Android phone and suggests a design for a secured system architecture using multimedia video streaming. Our aim is to design an Android application that deals with HTTP based streaming and simultaneously achieves flexibility and security by using media-aware protection.

1.1 Previous Work

In the research field of multimedia streaming, the focus is individually given on the flexibility and security requirement, however, only a few research works jointly consider both [17].

1.1.1 Adaptive Streaming

Video adaptation is the most crucial aspect of media streaming. Several coding strategies address the issue of serving divergent clients with adaptive video quality [17]. Simulcast is majorly used for video adaptation. In this technique a single video is encoded into many individual streams, each having its own bitrate and quality, according to the clients [17]. A client can select one of them as per the bandwidth. However, this method is unsuitable for mid-network nodes, and simply switches between various video streams having different bit-rates [17]. Layered scalable coding is another technique where the entire video file is divided into a base layer and many other layers. Layers from the stream can be added and dropped to achieve video adaptation [17].

1.1.2 Secure Streaming

Secure RTP (SRTP) was introduced with the focus of security of RTP flows that provides the important features of cryptography, i.e., confidentiality, authentication

and protection for RTP traffic along with its associated control traffic [17]. SRTP achieves secure streaming by providing the basic requirement of security services, between a sender and a receiver, however, it does not provide the ability to securely adapt the protected media, due to which sender authentication is kept unresolved for consideration of computation cost and bandwidth overhead [17].

Initially, our efforts were towards researching the core section of the project, i.e. the algorithm to implement Secure Video Streaming. The first milestone was to perform video streaming on an Android platform. Hence, we performed our research and implementations for streaming videos using RTSP.

In addition, the application needed to stream videos only to authenticated users via an Android OS. We researched various authentication approaches and our main research was based on an authentication methodology using a Darwin Streaming Server (DSS). However from an Infrastructure point of view, we considered our study of Web Servers and Databases to be more useful.

In the implementation phase, an “Administrator” was given complete access as both a Member and a Manager of the application. After looking through various research papers, we decided to conduct the authentication part using RSA, where every user first registers and then logs into the application. We used AES as the encryption algorithm, both because it is the current standard and is considered highly secure, and because it is practically the only symmetric algorithm guaranteed to be available on all Android versions. Streaming video files along with Security on the Android Platform is done using HTTP. In order to improve the Response Time of encryption and decryption of videos we developed a new approach, i.e., “Random Block Video Encryption,” which helps in improving the total response time of the process.

1.2 Contributions

The contribution of this project is in the design and implementation of an open-source video streaming application on the Android Platform and deals with a HTTP based streaming. The major emphasis is on two things: authentication of the user accessing the application and encryption and decryption of videos based on performance.

Many of the current algorithms encrypt videos at compression time and decrypt at decompression time. However, these algorithms were not feasible for us because they require a customizable video player in order to encrypt videos at compression time and decrypt at decompression time. Also AES, the traditional encryption algorithm, is unsuitable in real time video applications due to complex computation and slow speed. Therefore, we designed the new approach where we did selective encryption using random block method, and thus achieved faster results by compromising some security. This project was integrated by combining the following components:

1. An Android application capable of serving secured and authenticated video content;
2. A web interface to upload content and manage users as needed for application authentication;
3. MySQL database for organization and ease of data between interfaces;
4. An Android based Java application (.apk) file for installation of application on Android platforms;
5. An end-to-end performance evaluation of the resulting system.

The organization of this project report is as follows: Section 2 briefly describes the relevant background regarding video streaming and other related topics such as streaming principle, types of video streaming, etc; Section 3 describes the web server and database details in addition to the use of an admin panel; Section 4 describes the Android application along with authentication details using RSA; Section 5 covers the implementation details of video encryption using AES and our approach using a random block methodology; Section 6 covers the detailed analysis of a variety of encryption strategies and their comparison with our approach; and finally, the paper concludes with our conclusions and suggestions for future work in Section 7.

CHAPTER 2

Background

We begin with the implementation of related concepts and attempt to cover relevant background information regarding our study and research on various concepts, including video streaming, its principle, and types of video streaming. We also briefly discuss the research made on how encryption of video and user authentication is accomplished using AES and RSA respectively.

2.1 Video Streaming

A Streaming mechanism is a technique for transferring data where as the file is sent to the end user and is processed as a steady and continuous stream [11]. Streaming video is a sequence of moving images, which are transferred in the compressed form and sent it over the Internet to viewers so that they can display it on the screen as they arrive [5]. If video data is received by an end user as it streams, then users do not have to wait to download a large set of files, before watching video or listening to the audio [11].

2.1.1 Streaming Principle

In the process of streaming multimedia, media files are delivered by a provider, and are constantly received by, and normally rendered to the end-user's screen [31]. In streaming applications the transmission time of data packets is important since the packets should reach their destination in a timely manner. Even the smallest time delay in this process can cause network congestion, and eventually result in the loss of delayed packets [25]. This results in a loss of quality data, breakage of

the synchronization between client and server, and error propagation in the rendered video [25].

2.1.2 Video Streaming Types

1. Downloading

When a particular file is downloaded, it first gets saved on a local computer, and then can be accessed for viewing. The major disadvantage of this platform is having to wait for the entire file to download before the user can view it [29]. For smaller files this should not be a concern, however for larger files and long presentations it can be inconvenient [4].

2. Streaming

In this type of video streaming, the end user has the advantage of instantly watching file from the moment download starts. In effect, the file is sent to the user in the form of (more or less) constant streaming; and the end user can watch the video immediately as and when it arrives [28]. The obvious advantage of this method is that the waiting time is zero. Streaming media also has additional advantages such as being able to broadcast live events [4].

2.2 Streaming Process

Streaming takes place in following four steps:

1. The client will visit a Web page and will request a particular file;
2. The page then visits the Web server to find the requested file;
3. The Streaming server streams the file directly to the client and;

4. The Client software decodes and plays the requested file.

Please refer to Figure 1 for the basic streaming process:

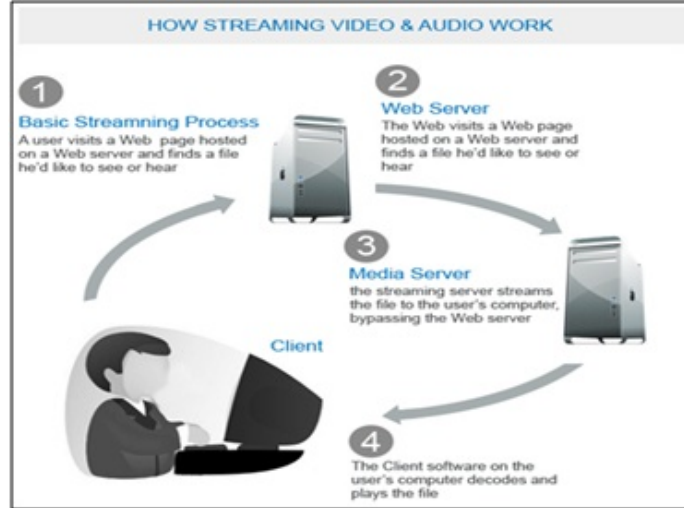


Figure 1: Streaming Process

2.3 AES for Encryption

Advanced Encryption Standard (AES) falls under the category of cryptographic mechanisms. It is mainly used to protect electronic data and to maintain its safety [16]. AES is an iterative, symmetric-block cipher that makes use of keys with the length of 128, 192 and 256 bits, and performs encryption and decryption on data blocks with the size of 128 bits, i.e. 16 bytes [16]. However, a symmetric-key cipher uses the same key to encrypt and decrypt data as compared to public-key ciphers, which make use of a pair of keys [16]. In the case of Iterative ciphers, encryption and decryption are performed in the form of a loop structure where input data undergoes a series of permutations and substitutions [16]. These ciphers use a loop structure that repeatedly performs permutations and substitutions of the input data [16].

Advanced Encryption Standard, is executed in four steps by performing a series

of an iteration on input data. AES performs ten iterations for an 128-bit length key, 12 iterations for an 192-bit key, and 14 iterations for a 256-bit key [30]. We discuss, in brief the following in order to understand the steps performed in iterations during an AES implementation:

1. SubBytes: at this step of the iteration, non-linear byte substitution for each byte of the block takes place on the input data [30].
2. ShiftRows: then, the bytes are cyclically shifted within the block [30].
3. MixColumns: at this step, group of four bytes is formed, forming four-term polynomials. It then multiplies the polynomials with a fixed polynomial $\text{mod}(x^4 + 1)$ [30].
4. AddRoundKey: finally, it adds the round key with the block of data [30].

In our implementation of “MediaStreaming” application, we have completed encryption and decryption of videos - 16-byte block of data using a 128-bit key.

2.4 RSA for Authentication

RSA is a widely used algorithm for encryption and authentication for many websites and in the world of the Internet [20]. It was developed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. RSA is also included as part of the Web Browsers from Microsoft and Netscape [20]. The encryption system produced by RSA algorithms is owned by RSA Security [20].

This algorithm involves a series of mathematical operations performed on two large prime numbers, resulting in a set of two numbers that constitutes the Public key and another set as the Private key [20]. The encryption and decryption take place

with the help of these two keys generated pertaining to a particular user [20]. The Public key is open to be read by everyone; however the Private key is only known to the owner [20]. In an RSA algorithm the private key is never sent across the internet [20], hence it is not susceptible to an attack by hackers.

The Public key is used to encrypt the text which is then decrypted using the Private key obtained from the same set of numbers that underwent mathematical operations [20]. Thus, when User A sends a message to User B, User A fetches User B's Public key from the central administrator and sends an encrypted message to User B using B's public key [20]. User B can decrypt this message using their own Private Key. This process ensures privacy; however, User B can be authenticated to the User A, by B using their own Private key to encrypt a digital certificate [20]. User A can decrypt the sent digital certificate using B's Public-key [20], assuring to User A that the message received is from User B, hence User B, is authenticated [20].

CHAPTER 3

Web Server and Database

Every architectural implementation, irrespective of any domain, needs a strong base beneath it and records to store inputs and outputs. Our application “MediaStreaming” also needed a platform to build the application and a Database to store the information relevant to the application.

An Infrastructure (Server) helps different applications spread across the globe at various geographical locations to work together as a single Enterprise [23]. It easily manages the large number of users and transactions within an enterprise. The Infrastructure behaves as a backbone of an Enterprise to work in a Unity [23]. It also helps an enterprise to be more productive [23]. thus, the Infrastructure helps developers build multi-tier applications [23] and acts as a platform by integrating diverse computers, networks using multiple operating systems and software packages [23].

Our Infrastructure widens over a diverse computer’s Window’s Operating System, an Android platform and various software packages. We shall discuss briefly the architecture design, before we discuss the details of the Infrastructure involved in our project.

1. **Presentation Layer:** The project demanded an user interface wherein users with their privileges can create individual accounts and upload content (videos) on a server. As the application was developed, keeping in mind worldwide users, we needed a WWW domain.

2. **Application Logic Layer:** The application layer represents elements that are specific to this application, and contain back-end processing for the UI, and bindings between the application and business logic layer. It includes the video streaming part of the application as well.
3. **Business Logic Layer:** The business logic layer contains logic that is specific to a business domain, i.e. functionalities such as encryption, decryption of videos needed to be performed.
4. **Data Access Layer:** A Database was needed to store in user information and video details, and also the content for the entire application.

Please find Figure 2 for the pictorial representation of all the layers mentioned above.

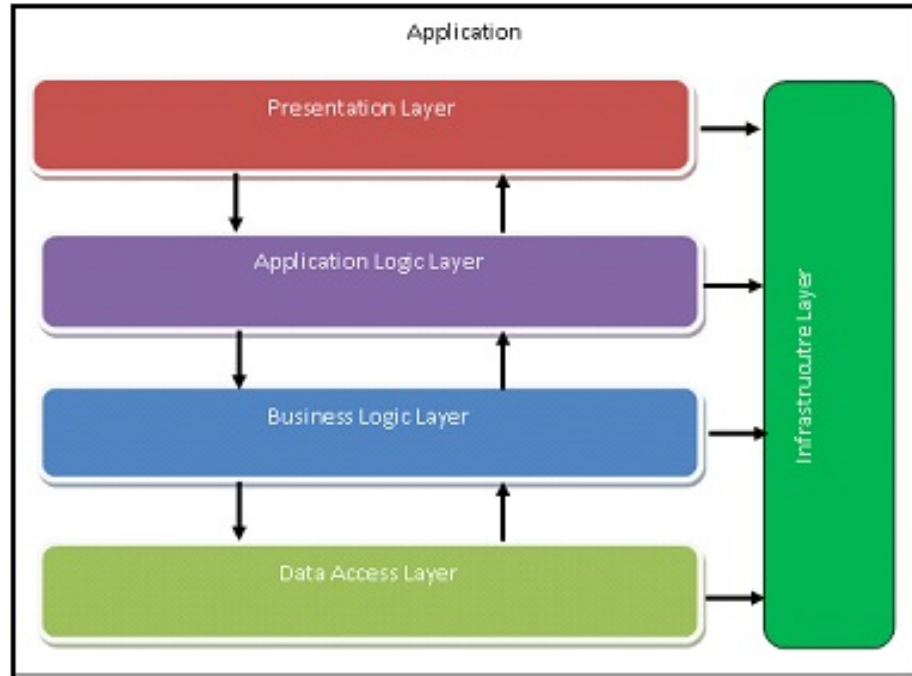


Figure 2: Project Infrastructure

3.1 Web Server

Roles and Significance of a Web Server: A Computer (hardware) or a Computer Application (software) that renders content \information to web users via the Internet can be referred to as a Web Server [32]. A Web Server is mostly used to host websites. It is also used for the purposes of data storage, running enterprise applications, etc. [32]. A Web Server delivers web pages in response to the requests sent by clients using the Hypertext Transfer Protocol (HTTP) [32]. These Web pages are simply HTML documents along with additional content such as images, style sheets, and scripts [32]. While our project's, you would see, while the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting Web forms, including uploading files. Hence, the architecture elements mentioned in Figure 2 demonstrates an Infrastructure Layer that performs the execution and implement each layer's functionality.

Infrastructure Layer

The Infrastructure was intended to be such that the application was to be made accessible by WWW. Hence, we needed to perform Hosting and have a domain for the application.

Domains are available and provided by many Web dealers. As this was not a major part of our project, we bought the domain “www.securemediastreaming.com” from one of the dealers and hosted this domain on a Windows platform. Below are the details for the Server:

General Details Primary Domain Name: securemediastreaming.com Platform: Windows

Programming Languages Supported:

1. PHP v5
2. ASP
3. ASP .NET v3.5

Database:

1. MySQL
2. MsSQL

Infrastructure:

1. Dual Quad Core Xeon Processors
2. 24GB Ram
3. Highly redundant architecture

3.2 Admin Panel and its Features

The application intended to provide user privileges to all the users in order to upload their choice of video files. Admin panel also contains an Administrative side, for monitoring all content being uploaded and for accessing user information and other managerial features. In our application, the Administrator is a manager who is responsible for the required monitoring of the application by handling registered users, their details, and information about the uploaded videos on the server. An Admin can edit and delete a user and video contents. An Admin panel's home screen is shown in Figure 3.

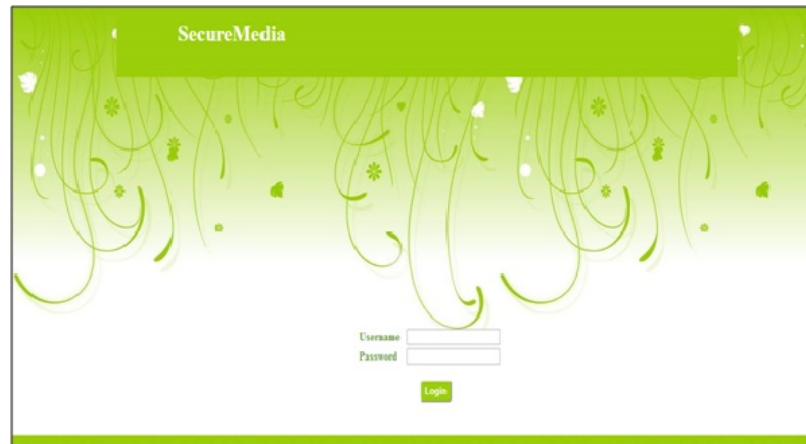


Figure 3: Admin Panel Home Screen

Figure 4 shows the Manage Contents screen, after successfully logging in using admin credentials.

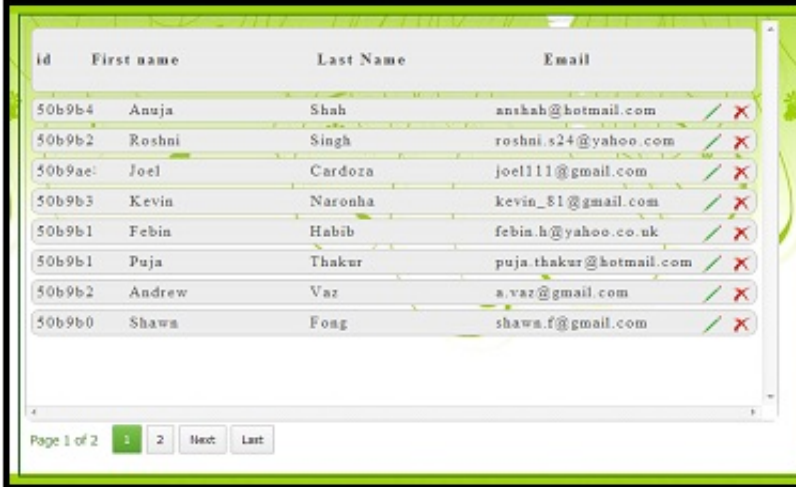


Figure 4: Admin Panel Manage Contents

3.2.1 Manage Users

“Manage Users” is one of the sections present in the UI application that only an Administrator can access. When the Administrator successfully logs into the UI application, the “Manage Users” button helps the Admin user navigate to the User Information section as shown in 5. This section of the application allows the Admin User to edit or delete User Accounts related to the application. Figure 5 shows the

Manage Users feature of admin panel.



id	First name	Last Name	Email		
50b9b4	Anuja	Shah	anshab@hotmail.com	✓	✗
50b9b2	Roshni	Singh	roshni.s24@yahoo.com	✓	✗
50b9ae	Joel	Cardoza	joel111@gmail.com	✓	✗
50b9b3	Kevin	Naronha	kevin_81@gmail.com	✓	✗
50b9b1	Febin	Habib	febin.h@yahoo.co.uk	✓	✗
50b9b1	Puja	Thakur	puja.thakur@hotmail.com	✓	✗
50b9b2	Andrew	Vaz	a.vaz@gmail.com	✓	✗
50b9b0	Shawn	Fong	shawn.f@gmail.com	✓	✗

Page 1 of 2 1 2 Next Last

Figure 5: Manage Users

3.2.2 Manage Videos

“Manage Videos” is another section present in the UI application, that only Administrator can access. When the Administrator successfully logs into the UI application, the “Manage Videos” button helps the Admin user navigate to the video information section as shown in 6. This section of the application allows the Admin user to delete video files present on the server and its relevant information from the Database. Figure 6 shows the Manage Videos feature of an admin panel.

3.3 Database

The database is widely used for data storage and for dynamicity of website and applications. It is a major part of the application where users interact with the system to fetch and upload content.

Designing the Database is a very crucial stage involved in the development [26] of an application and is a requirement before actual coding begins in order to produce

Seed	Title	Format	Path
479225	voltage	3gp	enc_voltage.3gp
145058	Famous	3gp	enc_video2.3gp

Figure 6: Manage Videos

a high-performance application [26]. The database needs to be designed efficiently, so that it contains only relevant and required information [26]. A well-designed database keeps queries simple for faster and efficient results [26]. Hence, it is always recommended that a good amount of time is spent on designing the database before coding in IDE [26].

The database for our application was designed in MySQL and we stored it on a Windows OS web server. When the user registers through an Android application to access the videos, they have to fill up required information such as: First Name, Last Name, Address, Email Address and Password. After successfully registering, this data is stored within the database on the server side. The password is converted to an encrypted password and stored along with Salt to increase and ensure the security of the system.

The application has a Data Management Layer where all the information related to the authentic user's details is stored as a security parameter.

A MySQL Database controls the complete storage management of data in the form of schemas, pertaining to users and videos.

A “Media Streaming” application consists of two schemas in the database: “Users” and “Videos” for storing the respective information in tables.

Users Schema: This table is used to store user-related information gathered during registration of the application and also authentication details required for users to access the Android application. Listed below are the columns defined within the schema:

1. Unique_id: Primary key of the schema
2. Firstname: First Name of the User
3. Lastname: Last Name of the User
4. Email: Email address of the User
5. Encrypted_password: Password of the user stored in encrypted form
6. Salt: Salt information to improve the strength of password encryption
7. Created_at: Account creation date
8. Updated_at: Account last modified date
9. Pkpublic: Public key assigned to the user during registration which would be used for user’s authentication

See Figure 7 as it represents the User Schema.

address	email	encrypted_password	salt	created_at	updated_at	pkpublic
San Jose, CA	anshah@hotmail.com	k9ZjyZKxRS/Y5FjeIX52P1VozZGYxZD3ODU0MDE0Y2Q0MT...	3d1d27854014cd414a5210b27028a9	2012-12-01 02:38:27		NULL -----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
Noida, India	roshni.s24@yahoo.com	xdRBncRBWn1EpLHN4mgSUTHYIBIMGVZThjyA0NjFzGVmMz...	e0ebe8c60461bde37ec31b7a86fb8fb	2012-12-01 02:33:45		NULL -----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
Florida	joel111@gmail.com	Q2uFUW2mGD4QXURU7MevTzl+esxMTEzMWRIZTImMmU5ZWEzMj...	11131dee92e9ea328afb84ce700c742	2012-12-01 02:14:44		NULL -----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
Main Street, Milpitas, CA	kevin_81@gmail.com	K0G+RJ6qzL1NjbZNU40elpYbwZYmQ4NmFjhzMzZWYwNjJjZT...	6bd86ac733ef062ce4a965240ec04e31	2012-12-01 02:35:29		NULL -----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
London, UK	febin_h@yahoo.co.uk	kSiD4JYaSdSrSwz4G5ijYDfFqpmIjRhNjVmZDdZVWY4MZY5OT...	f54a65d7be83f99525134bf1c78aa0	2012-12-01 02:28:27	0000-00-00 00:00:00	-----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
Bandra (West), Mumbai, India	puja.thakur@hotmail.com	HS4GBFAUUI/mnNdxjslGJTQp1LQ1MWRhNjJhNGQ3YzFhZmVmNz...	51da62a4d7c1afe7718ccc3907551f2	2012-12-01 02:27:07		NULL -----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
Texas	a.vaz@gmail.com	33dkmz+4esNKwQpFdVXD8sgQopkZDQ0YTdMGM3ODiYYWYz...	d44a7e0c789ba72c33278b65a8d8235	2012-12-01 02:32:28		NULL -----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
Los Angeles	shawnt1@gmail.com	7td5zty9RCn6TaMBaGYOnY9aZkZGhwYzI4MTI4ZTJkOTFmYm...	d4b0c28128e2d91fb99232113123fc2	2012-12-01 02:24:28		NULL -----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
Mission Road, Santa Clara, CA	robert.de@yahoo.com	3dVvulzE8UkkaLx1xNOUXVZnEylzE3ZGNNTihZTY1nM5NW...	2717dcb59ae657395a9446a4bc0505c	2012-12-01 02:25:43	0000-00-00 00:00:00	-----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...
Germany	msif_21@hotmail.com	VdLtv+Oiox84UjyQJlreSOAREwNTNhzGJjMg2hTcZYWFJMD...	053adbc286576aac05442cdcdbf14112	2012-12-01 02:22:56	0000-00-00 00:00:00	-----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQ...

Figure 7: User Schema

Videos Schema: This table is used to store video related Information for videos uploaded by an Administrator on the Web server. Listed are the columns defined within the schema:

1. Title: Title \Name of a video
2. Path: Complete location along with the filename
3. Aeskey: The secret key used for the symmetric algorithm
4. Aesiv: The initialization vector (IV) required for the symmetric algorithm
5. Format: Video format
6. Seed: Seed value that is used during the AES encryption and decryption of videos.

Figure 8 represents the Video Schema.

title	path	aeskey	aesiv	format	seed
voltage	enc_voltage.3gp	CZ3lVNlrpWx/Kg4sFfY3Fw==	1MkMg31D5QIRdqSwrTGcYQ==	3gp	479225317
Famous	enc_video2.3gp	MUcDFopwi1zqR0i3LMkPXA==	QnoalTdiD8han6MaEY+Aog==	3gp	1450586551
Song	enc_Karaoke.3gp	mOq8mwa0jW8myX3q/ZeGiw==	fK2x1TgBY8QlnD9+FPptlQ==	3gp	759374859

Figure 8: Video Schema

Encrypt Password using Salt: Usually most of modern Web applications need to encrypt a user's passwords. As soon as an end user signs in using a password, from that moment onwards, their password is encrypted and stored in order to maintain security.

Data stores and communication can be compromised. Therefore, we should take into consideration that our user's passwords are delicate personal data. Passwords are keys to their personal privacy and include sensitive data, therefore we do not have the right to hinder their privacy or to know their passwords.

In the case where passwords are simply encrypted, they are easily accessible (such as the traditional UNIX password file). Without a Salt, the attacker can take a list of encrypted passwords and run a pre-encrypted dictionary against each quickly and effortlessly.

In a cryptographic system, Salt plays an important role. It consists of random bits and creates one of the inputs to a one-way function that generally uses a cryptographic hash function. A password is treated as the other input. Along side of the Salt, the output of an one-way function is stored, rather than storing the password, and is used for the user authentication.

Hence, the Salt just needs to be random. It can be freely known as it doesnot help an attacker gain password information. Our system will store the plain text Salt within our database in a column next to the hashed password.

The main purpose of a Salt is to identify if two users are sharing the same password. If the random Salt is not present for every individual password, then the hash value of each password will be the same; this becomes obvious if the password for the first user is cracked then the second user must be sharing a similar password.

Public Key: A cryptographic system uses two keys, a public key known to everyone and a private or secret key known only to the recipient of the message. When User A wants to send a secure message to User B, A uses B's public key to encrypt the message. User B then uses his private key to decrypt it.

The crucial factor about a Public-key cryptosystem is the relation between the Public and Private keys as encryption could be possible only with the public key, whereas for decryption one needs to use its corresponding private key. Furthermore, it is virtually impossible to discover Private key even if the Public key is known.

CHAPTER 4

Android Application

As introduced in the earlier phases of this paper, video application “MediaStreaming” is developed and dedicated for the Android platform; and this application takes care of the security of media being transmitted onto the Android platform. This application is developed using the Android Java language, and it has two parts:

1. User Registration
2. Video Streaming

When the users launch a “MediaStreaming” application on their device, they are presented with a Sign In screen. If the user is not registered, there is an option to register before Login. After successful registration, the user now has an access to various videos, which are uploaded to the server. The User can select a video and then play, pause, rewind, and forward it. The user may logout once they have watched the desired video.

4.1 User Registration

User Registration is completed using the Android Java code. The steps for registration are explained as follows: 1. The user first installs an Android application and gains a servers RSA public key. 2. The user creates RSA keys while registering within the application by providing the information as per Figure 9:

Figure 9 represents the User Registration part of the Android application.

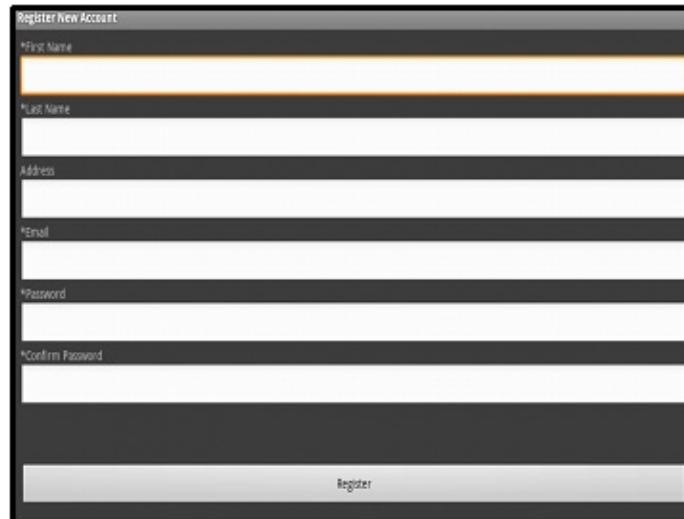
A screenshot of a web form titled "Register New Account". The form contains several input fields: "First Name" (with an asterisk), "Last Name" (with an asterisk), "Address", "Email" (with an asterisk), "Password" (with an asterisk), and "Confirm Password" (with an asterisk). Each field is represented by a white rectangular box with a thin black border. At the bottom of the form is a wide, light gray button labeled "Register".

Figure 9: User Registration

During the registration, if the values for the user's password and confirm password do not match, then user receives an error message stating, both passwords should match. Figure 10 represents this function.

A screenshot of the same "Register New Account" form, but now it contains data. The "First Name" field is filled with "Florida". The "Email" field is filled with "jasmine.f@gmail.com". The "Password" field is filled with seven dots. The "Confirm Password" field is also filled with seven dots. Below the "Confirm Password" field, there is a red error message that reads "Both the passwords should match". The "Register" button is still at the bottom.

Figure 10: Password Validation

The registration part of the application is well supported with all the field validations, as per Web standards, and the requirements of the application.

Only after successfully registering their required information can the user send their RSA Public key to the server.

4.1.1 User Login \Sign In

The Login screen of the Android application is shown in Figure 11.



Figure 11: User Login Screen

Once at the the Login Screen, the user needs to enter their credentials, which were created during the initial registration process. Once the user logs in using these credentials, they are redirected to the welcome screen of our media streaming application.

4.2 Video Streaming

When the user registers through the Android application, they have access to videos that are being uploaded by the Admin on the server.

The user is taken to the welcome screen when they are logged in. Videos can be uploaded by the Admin and the name of the videos can be changed. This will be reflected as videos names and seen by the users when they come across this screen. Figure 12 represents the “Welcome Screen” in the Media Streaming application that displays the list of available videos for the user to stream.

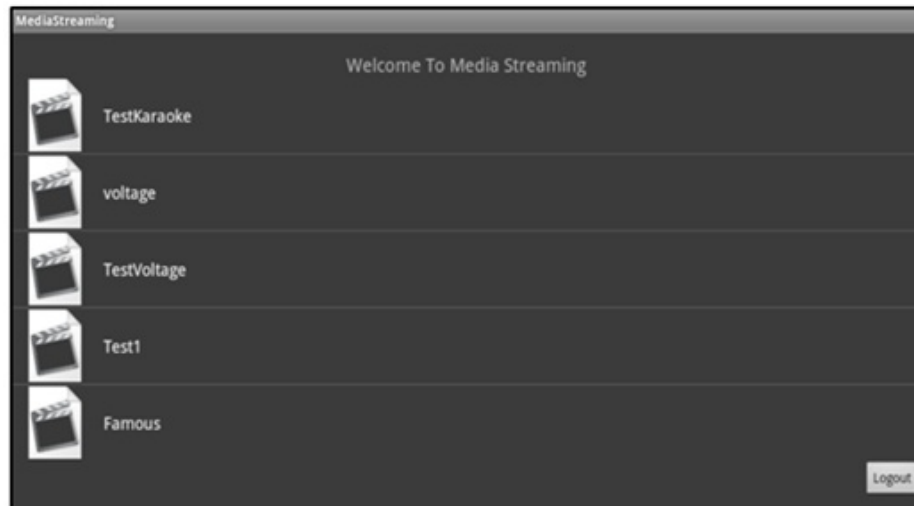


Figure 12: Media Streaming Welcome Screen

The user can select the video of their choice from available video options and then play it. Once the user starts watching the video, they can take advantage of the features such as, *Pause*, *Rewind*, and *Forward*. Figure 13 represents the actual streaming of the selected video where the user can pause, rewind and forward the video.



Figure 13: Video Streaming

Figure 14 represents the sequence of flow and the back end functionalities taking place during the entire process after the user logs into the Android application until the video is streamed.

User of Android Application	Web Server
The user sends the login request to the server	
	<ul style="list-style-type: none"> • The server creates a random number • Encrypt with corresponding user public key.
<ul style="list-style-type: none"> • The user receives an encrypted random Number • Decrypt with his/her private key. • Send a random number to the server. 	
	<p>The server checks this with a previous random number,</p> <ul style="list-style-type: none"> • If it matched, then the user is authenticated else an error message will be generated • If authenticated then the server creates an AES key for encryption and decryption. • Sends this key to the user
<ul style="list-style-type: none"> • The user receives an AES key. • User requests for video streaming. 	
	The server encrypts video with AES key and sends to the user.
User decrypts this video with AES key and plays it.	

Figure 14: Tabular Flow of User Login and Video Streaming

CHAPTER 5

Encryption and Decryption

Video encryption is an extremely useful method for stopping unwanted interception and viewing of any transmitted video. The main goal of cryptography is keeping data secure from unauthorized attackers. The reverse of data encryption is data decryption, which recuperate the original data. Security, time efficiency, format compliance, and compression friendliness are some of the important features of a video encryption algorithm. Security is a basic requirement, which means that the cost of breaking the encryption algorithm is equal to the cost of buying a video authorization. The time efficiency means, encryption and decryption should not take much time as the heavy delay is not acceptable in real time.

In order to encrypt a video before it gets uploaded to the Web server for streaming, we used the complete video encryption methodology using the AES algorithm. However, later we discovered that the time it takes to encrypt a video is comparatively high, and it could cause an issue when we encrypt a video of larger size. Looking at the time efficiency of the algorithm, we decided to design an algorithm that does not need to encrypt an entire video but only random blocks of it. We then used a random block encryption method, which is explained in Section 5.3, to make the video encryption more time efficient. The file format we used during the implementation of this project is .3gp.



Figure 15: Video Encryptor and Uploader

5.1 .3gp video file format

The “.3GP” file extension is a well known mobile video file format and it delivers multimedia over third generation mobile networks [33]. Generally, mobile phones that are equipped with a recording facility use this file extension as a standard file format to record multimedia clips. One of the crucial factor of using .3GP technology is that it downloads or transfers video and audio clips faster [33].

The latest mobile phones use the .3GP file format for creating, playing, and transferring media files over third generation wireless networks. Typically, this fast paced, .3GP file format contains audio and video streams [33]. MPEG-4 Part 10 or MPEG-4 Part 2 usually contain video streams that are structurally based on MPEG-4, whereas audio streams can be stored in various formats, which includes AAC or AMR [33]. Third generation file format refers to big-endian formats; where the first bytes contain the most significant data and are stored in the earlier section of the file [33].

Figure 16 , Figure 17, and Figure 18 represents the “General”, “Video” and “Audio” details of the .3gp file format respectively.

General	
<i>CompleteName :</i>	C:\Users\jaie\Videos\sample 3gp videos\Manchester_United.3gp
<i>Format :</i>	MPEG-4
<i>Format_Profile :</i>	3GPP Media Release 4
<i>CodecID :</i>	3gp4
<i>FileSize/String :</i>	7.60 MiB
<i>Duration/String :</i>	5mn 33s
<i>OverallBitRate/String :</i>	191 Kbps

Figure 16: General Details of the .3gp file

Video	
<i>ID/String :</i>	1
<i>Format :</i>	MPEG-4 Visual
<i>Format_Profile :</i>	Simple@L1
<i>Format_Settings_BVOP/String :</i>	No
<i>Format_Settings_QPel/String :</i>	No
<i>Format_Settings_GMC/String :</i>	No warppoints
<i>Format_Settings_Matrix/String :</i>	Default (H.263)
<i>CodecID :</i>	20
<i>Duration/String :</i>	5mn 33s
<i>BitRate_Mode/String :</i>	Constant
<i>BitRate/String :</i>	128 Kbps
<i>Width/String :</i>	320 pixels
<i>Height/String :</i>	240 pixels
<i>DisplayAspectRatio/String :</i>	4:3
<i>FrameRate_Mode/String :</i>	Constant
<i>FrameRate/String :</i>	20.000 fps
<i>ColorSpace :</i>	YUV
<i>ChromaSubsampling :</i>	4:2:0
<i>BitDepth/String :</i>	8 bits
<i>ScanType/String :</i>	Progressive
<i>Compression_Mode/String :</i>	Lossy
<i>Bits-(Pixel*Frame) :</i>	0.083
<i>StreamSize/String :</i>	5.10 MiB (67%)
<i>Encoded_Library/String :</i>	Lavc52.20.0

Figure 17: Video Details of the .3gp file

Audio	
<i>ID/String :</i>	2
<i>Format :</i>	AAC
<i>Format/Info :</i>	Advanced Audio Codec
<i>Format_Profile :</i>	LC
<i>CodecID :</i>	40
<i>Duration/String :</i>	5mn 33s
<i>BitRate_Mode/String :</i>	Constant
<i>BitRate/String :</i>	59.0 Kbps
<i>BitRate_Nominal/String :</i>	32.0 Kbps
<i>Channel(s)/String :</i>	2 channels
<i>ChannelPositions :</i>	Front: L R
<i>SamplingRate/String :</i>	44.1 KHz
<i>Compression_Mode/String :</i>	Lossy
<i>StreamSize/String :</i>	2.34 MiB (31%)

Figure 18: Audio Details of the .3gp file

5.2 A Complete Video Encryption method using AES

In this type of encryption, the entire content of the video is first compressed and then encrypted using a standard traditional algorithm, AES. We use the AES as the encryption algorithm, both because it is the current standard and is considered highly secure, and because it is the only symmetric algorithm guaranteed to be available on all Android versions. However, this technique is unsuitable in real-time video applications due to heavy computation and slow speed.

The idea of this algorithm is to treat the .3gp bit stream as text data and does not use any of the special structure. This algorithm provides the security to the entire 3gp stream since every byte is encrypted. Using this algorithm is not an applicable solution for larger videos, since it is very slow, especially when we use a triple AES. Because of the encryption operation, delays increase and the overload will be unacceptable for real-time video applications.

The algorithm starts with the creation of an AES object, and then the generation

of a KEY and an IV for that object. Once the object and key creation is completed, we read the video file as a stream with its content, which is followed by passing that data onto an encryption function. This function creates a “CryptoStream” object from an AES key and an AES IV which then eventually reads the byte data, encrypts this data, and writes encrypted data into the “FileStream” Object. Object forms an AES key and an AES IV. This CryptoStream object reads byte data, encrypts this data, and writes encrypted data into a FileStream Object. The basic steps of a “Complete Video Encryption” are as follows:

1. Create AesCryptoServiceProvider object and generate an AES key.
2. Create file stream object and read video data.
3. Pass this data to an encryption function which creates the CryptoStream Object to encrypt the data and FileStream Object to write the encrypted data into a file.

5.3 Our approach of encryption using random block method

The algorithm used for our purposes randomly encrypts the bytes within video frames. Since our algorithm is not encrypting every single byte of the video data, it reduces the computational complexity. This section discusses the random block encryption, which only encrypts a subset of the data. The aim of a random block encryption is to reduce the amount of data that is encrypted while preserving a sufficient level of security.

In general, the size of a video file is larger than other files such as images, text files, etc. We wanted a more efficient and effective method for video encryption. However, we faced two major issues such as: time and streaming. Looking at these factors we

decided to design an algorithm that is more time efficient with large video files and, at the same time, has effective streaming. Currently accomplished algorithms exist, that can encrypt videos at compression time and decrypt at decompression time. However, these algorithms are not feasible for us because they require a customizable video player in order to encrypt videos at compression time and decrypt at decompression time.

With this in minds, we designed our new algorithm, “Random Block Video Encryption,” which is efficient in terms of timing and streaming. Our algorithm first divided the video file data into blocks of 16 bytes. We then generated a series of random numbers from a seed. The size of this series depends upon the encryption ratio. We encrypted blocks using AES, which are being generated in a random series. The security of this algorithm depends upon two factors: an AES key and a Seed used for Random Series. We increased the security of our algorithm by using a series of random numbers. If we encrypt a video by using a predefined method, then anyone can identify which blocks are encrypted and which are not. With random block encryption method, as the sequence of encryption is dependent upon a seed, an attacker cannot find the encrypted and unencrypted blocks. So “Random Block Video Encryption,” is Secure, Fast, and efficient for streaming.

5.3.1 Random Block Video Encryption (RBVE) Algorithm

- Read a video file: Here we first read the video file in the format “3gp”.
- Define the block size used for encryption and decryption: We use 16 bytes as a block since we are using AES; it works on 16 and multiples of 16 bytes, and if do not use 16, then it does padding.

- Get total bytes of video: We are calculating the total bytes of a video by dividing the file length by our defined block size i.e. 16.
- Define the encryption ratio: In this algorithm, we used the encryption ratio as 0.1.
- Calculate the encrypted blocks: Encrypted blocks are calculated by multiplying the total blocks with the encryption ratio.
- Create a series of random numbers from seed: The size of the series is equal to encrypted blocks.
- Sort that series in ascending order: Once we obtain the series of random numbers, by first sorting in ascending order before proceeding to the next step.
- Encryption of blocks: After finishing all the above steps, we have now encrypted the blocks that are in the series of random numbers using AES.

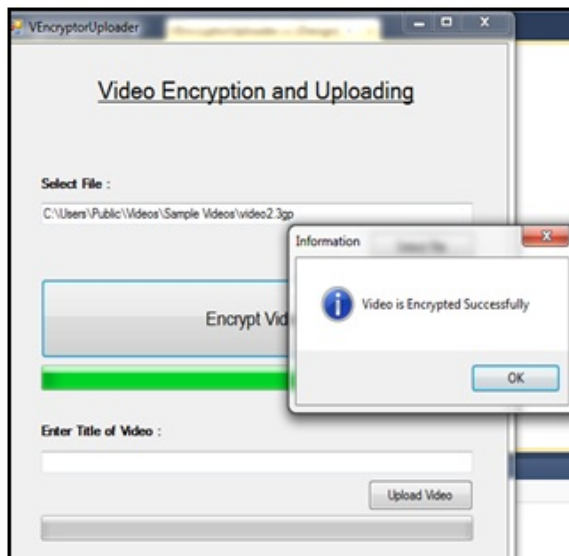


Figure 19: Video Encryption Process

5.4 Video Up-Loader

Once the video is encrypted we need to upload this encrypted video onto the Web-server so that authentic users can access and play video. We then need to upload video information onto the Web-server such as, video format type, AES key, and IV (Initialization Vector) used for encryption.

We used an FTP method to upload the encrypted video onto the Web-server. The encrypted video is uploaded to the Web-server, which then writes the video information, seed of random series, AES key and AES IV within the database of the Web-server. To write the data into a database, we have written the PHP code.



Figure 20: Video Upload Process

5.5 Video Decryption

For a video decryption, we first created a series of random numbers from the Seed where the size of the series depends upon the encryption ratio. We then fetched the data from defined positions of our series, followed by its decryption using AES. Finally, we wrote the decrypted data, which is then used for playing the video.

CHAPTER 6

Analysis of Encryption Schemes

Data encryption is a suitable method to protect data. Until now, various encryption methodologies have been proposed, that are mostly used for encryption of text and binary data [33]. Since video data are often used in real time applications and are large enough to handle, it becomes difficult to use them directly in video encryption. The wide use of multimedia files in various applications brings serious attention to security and privacy issues [33]. This section provides the evaluation and comparison of the established encryption algorithms based on performance parameters. We also discussed the graphical comparison of both the video encryption methods explained in Section 5. The complete video encryption method takes more time in encrypting and decrypting the video as compared to the random block encryption method. This comparison becomes more evident when the video size gets larger.

Encryption ratio (ER): This parameter gives the ratio between the size of an encrypted part, and the whole data size. To achieve a less complex computation, one should make sure that an encryption ratio should be minimized [33].

Speed (S): Generally in all the media and real-time video applications, speed is the crucial factor, as along with security, it becomes necessary for an algorithm to be time efficient to meet the real time requirements [33].

Compression Friendliness (CF): An encryption algorithm should not affect the data compression efficiency. Some encryption algorithms introduce more data that are necessary for decryption [33]. One should take care that the size of encrypted data will not be increased only after which, it will be considered as a compression

friendly algorithm.[33].

Cryptographic Security (CS): Cryptographic security is the most vital feature of cryptography. Checking the security, means verifying if the encryption algorithm would be safe with various cryptography and different plaintext-cipher text attacks [33]. Multimedia applications should consider this feature, as they contain the sensitive data which has valuable information [33].

Security, time efficiency, format compliance, and compression friendliness are the crucial factors of any video encryption algorithm, as multimedia files are widely used in real life applications [33]. Among them, security is the basic requirement, because the cost of breaking the encryption algorithm is equal to the cost of buying a videos authorization. Time efficiency means the total response time of encryption and decryption should be as low as possible as the heavy delay may affect the performance of applications [33]. Apart from that, the video should not be compressed after the encryption and decryption. The format compliant means that the data format of an encoded bit stream is not changed after an encryption process, to support such direct operations as browsing, playing, cutting, copying, and so on [33].

Pure Permutation: This algorithm implements an encryption by scrambling the bytes within a frame of MPEG stream with the help of permutation[33]. It is an extremely useful algorithm, when the hardware decodes the video, however, the software helps in decryption [33]. This algorithm is prone to a plaintext attack, and hence should be carefully used; since by comparing the ciphertext with the known frames, the attacker could easily recognize the secret permutation list. Once this list is figured out, it becomes easy to decrypt all other frames [33].

Fully Layered Encryption: This method uses, the traditional algorithms such

as AES or DES, to encrypt every single byte in the entire video stream. The aim of this algorithm is to treat the MPEG bit-stream as text data and does not use any of the special structure [33]. As every byte is encrypted, it provides the security to the entire multimedia stream, and until now, no algorithm has been introduced to break DES or AES. It is definitely not an applicable solution for large video files, because of its slow speed. Additionally increased encryption operation delay and overload will be unacceptable for real-time video application [33].

Our first method of video encryption, i.e., “Complete Video Encryption,” is based on the above established technique where we encrypt every byte of data and thus it takes a more amount of time to both encrypt and decrypt the videos.

Perceptual Encryption: This type of encryption is generally used in, pay-per-view video, and video on demand situation. This feature requires that quality of audio and visual data is only partially degraded by encryption [33]. Due to this perceptibility it becomes possible for potential users to listen and view low quality versions of multimedia products before buying them [33]. Factor p continuously controls the visual quality degradation, which usually represents a percentage corresponding to the encryption strength [33].

Random Block Encryption: There are already accomplished algorithms that are based on Selective encryption techniques, and encrypt videos at compression time and decrypt at decompression time. However, these algorithms are not feasible for us since they require a customizable video player for encrypting videos at compression time and decrypting at decompression time.

We increase the security of our algorithm using a series of random numbers. If we encrypt the video using a predefined method, then anyone can identify which blocks

are encrypted and, which are not. With this method, the sequence of encryption depends upon the Seed, therefore an attacker cannot find the encrypted and unencrypted blocks; this feature makes our algorithm fast and efficient by maintaining its security.

Figure 21 represents the comparison of above mentioned algorithms with respect to various parameters such as ER, S, CF, and CS. However, often it is desirable to sacrifice some degree of cryptographic security for improved performance.

Algorithm	Encryption Ratio (ER)	Speed (S)	Compression Friendliness (CF)	Cryptographic Security (CS)
Pure Permutation Encryption	High, Could be variable	Fast	Satisfied / Non- Satisfied	Satisfied / Variable
Fully Layered Encryption	100%	Slow	Satisfied	Satisfied
Perceptual Encryption	Variable	Variable	Non-Satisfied	Satisfied
Random Block Encryption	< 15%	Fast	Satisfied	Variable, Not as secure as Fully Layered Encryption

Figure 21: Comparison of Video Encryption Algorithms

It is difficult for a single algorithm to satisfy all performance parameters [33]. Encryption algorithms should be selected based on the requirements of an application in use [33]. It is challenging for researchers to design an encryption algorithm that maintains tradeoff among all parameters like speed, encryption ratio, compression friendliness, format compliance and cryptographic security [33].

Figure 22 and Figure 24 represents the encryption and decryption time comparison using both the methods which we described in this paper.

Figure 23 shows the graphical representation of above table, where “X” axis stands for the size of video and “Y” axis stands for their corresponding encryption

Video Size in MB	Avg Encryption time in seconds using CVE	Avg Encryption time in seconds using RBVE
0.98 MB	2.23	0.25
1.57 MB	5.66	0.55
2.26 MB	9.79	0.98
2.84 MB	14.85	1.5
3.52 MB	30.63	3.63
4.68 MB	48.9	4.85
4.99 MB	59.41	5.83
5.96 MB	78.61	7.89
6.89 MB	100.37	10.23
7.11 MB	114.92	11.46

Figure 22: Tabular view of comparison of Encryption Timings

time taken by “CVE” and “RBVE”.

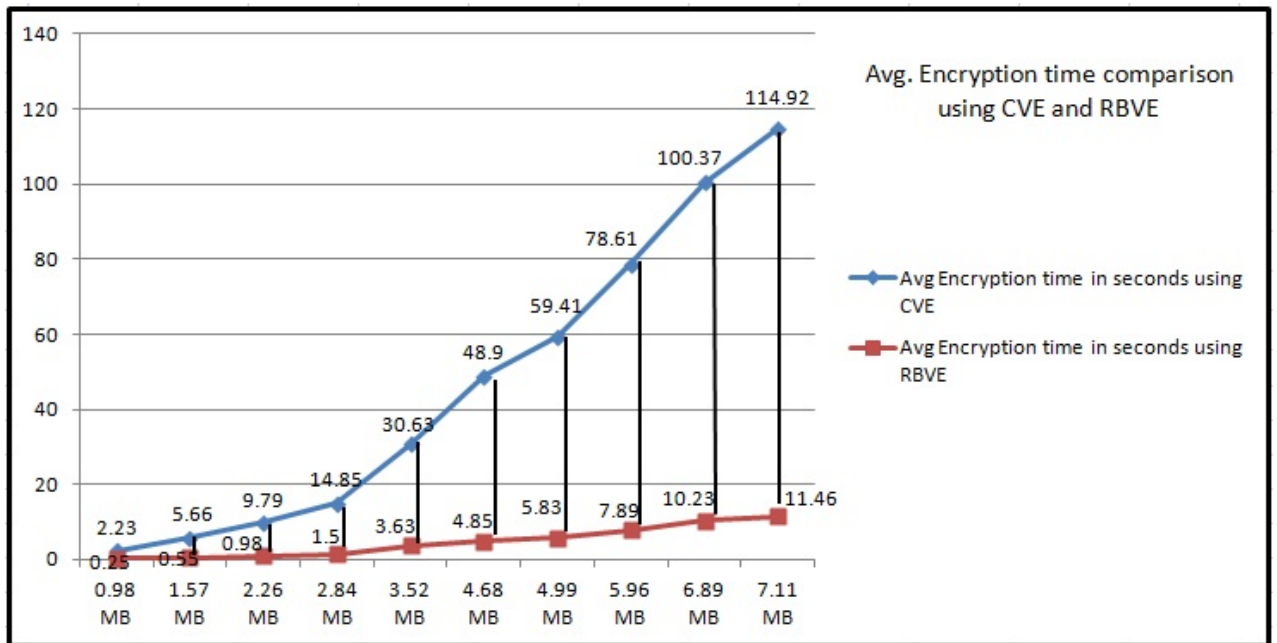


Figure 23: Graphical representation of comparison of Encryption Timings using CVE and RBVE

Figure 25 shows the graphical representation of above table, where “X” axis

Video Size in MB	Avg Decryption time in seconds using CVE	Avg Decryption time in seconds using RBVE
0.98 MB	2.24	0.24
1.57 MB	4.80	0.49
2.26 MB	10.42	1.09
2.84 MB	15.66	1.94
3.52 MB	28.95	3.04
4.68 MB	46.40	4.45
4.99 MB	62.37	6.08
5.96 MB	80.05	8.24
6.89 MB	108.51	10.19
7.11 MB	119.28	12.22

Figure 24: Tabular view of comparison of Decryption Timings

stands for the size of video and “Y” axis stands for their corresponding decryption time taken by “CVE” and “RBVE”.

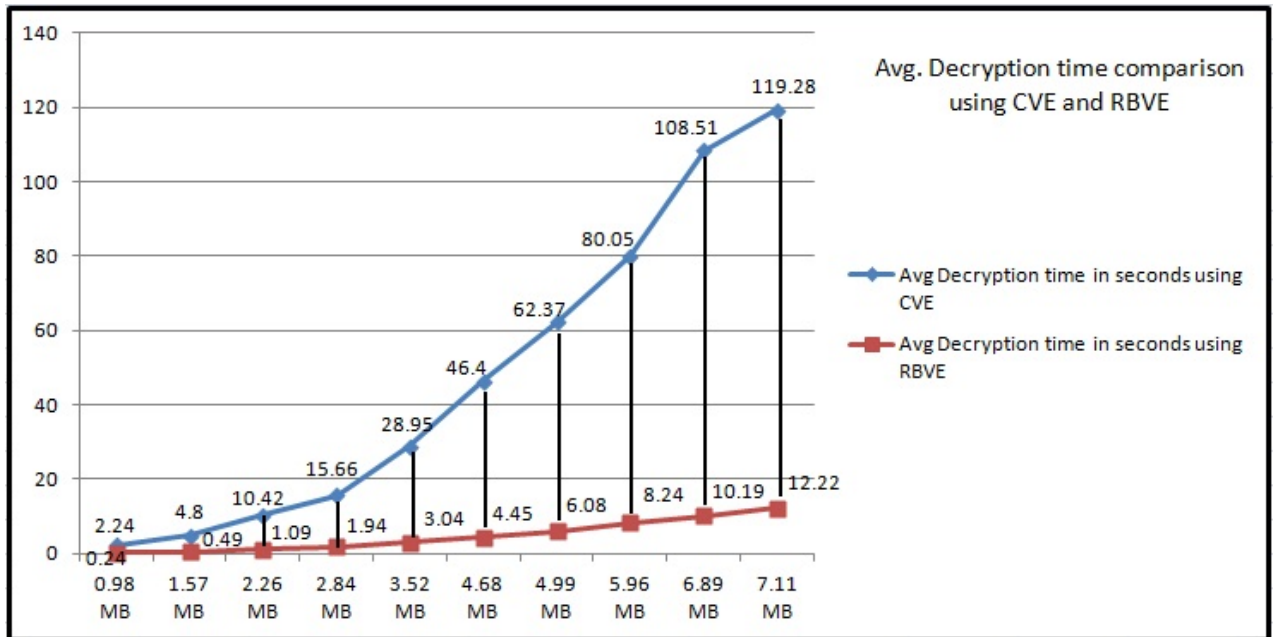


Figure 25: Graphical representation of comparison of Decryption Timings using CVE and RBVE

Comparison between the above-mentioned implementations, i.e. Complete Video

Encryption and Random Block Video Encryption can be easily depicted and interpreted in Figure 26 and Figure 27. The below graphs give the Performance analysis, at each iteration during encryption and decryption of a video, and help showcase the Time Difference between execution of both the algorithms. The analysis involves 10 iterations to display the Performance in the form of a Graph. The comparison is done based on Time taken in seconds at each iteration along the Y-axis and Iteration count along X-axis, during the process of encryption and decryption.

The application developed, graphically displays encryption and decryption measurements in terms of time consumed along with numerical figures to minutely locate time values involved in the completion of each iteration. The whole analysis is managed by a user interface that helps user browse a video file, on which encryption and decryption are to be performed using the two algorithms, and performs actions as per the mentioned instructions and events.

Once the user browses through a video file and clicks on “Encryption and Decryption,” an event is fired to start the encryption and decryption of the selected video file using both the algorithms. “Show Performance” helps to view a time analysis of both algorithms in a graphical manner, measured in Time along the Y-axis and iterations along X-axis. “Show Encryption Time” and “Show Decryption Time” events display the same, i.e. time difference for encryption and decryption numerically.

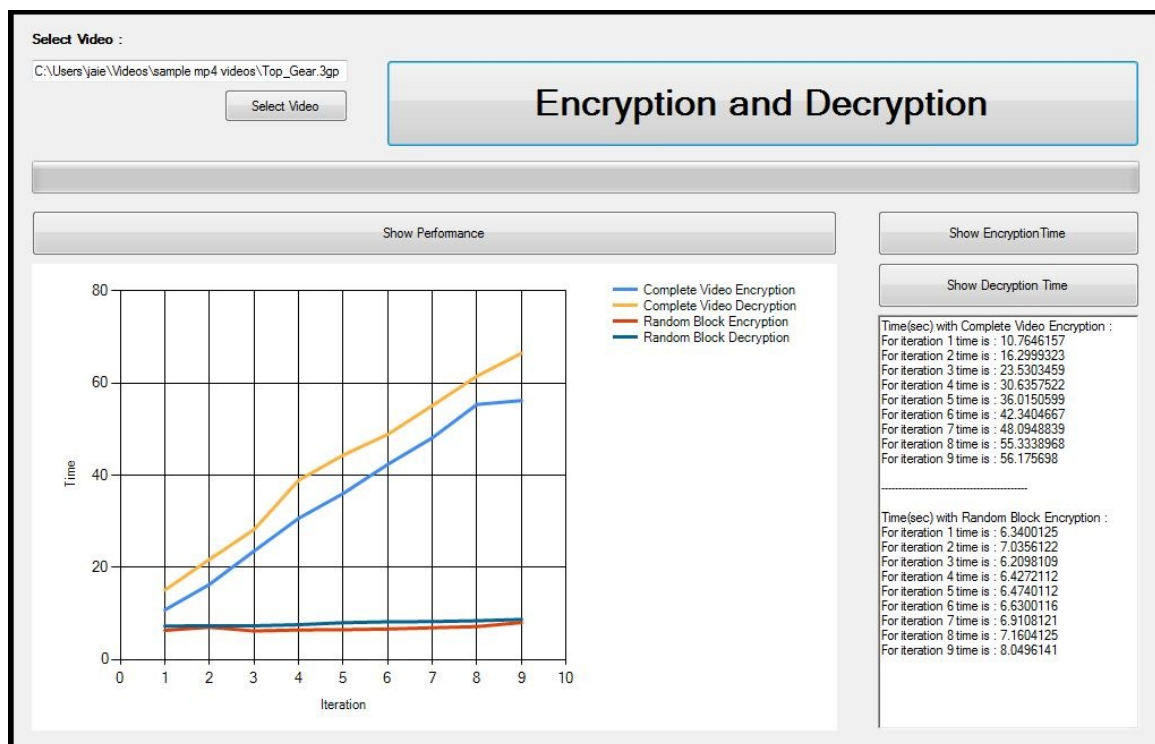


Figure 26: Comparison of Encryption Timings

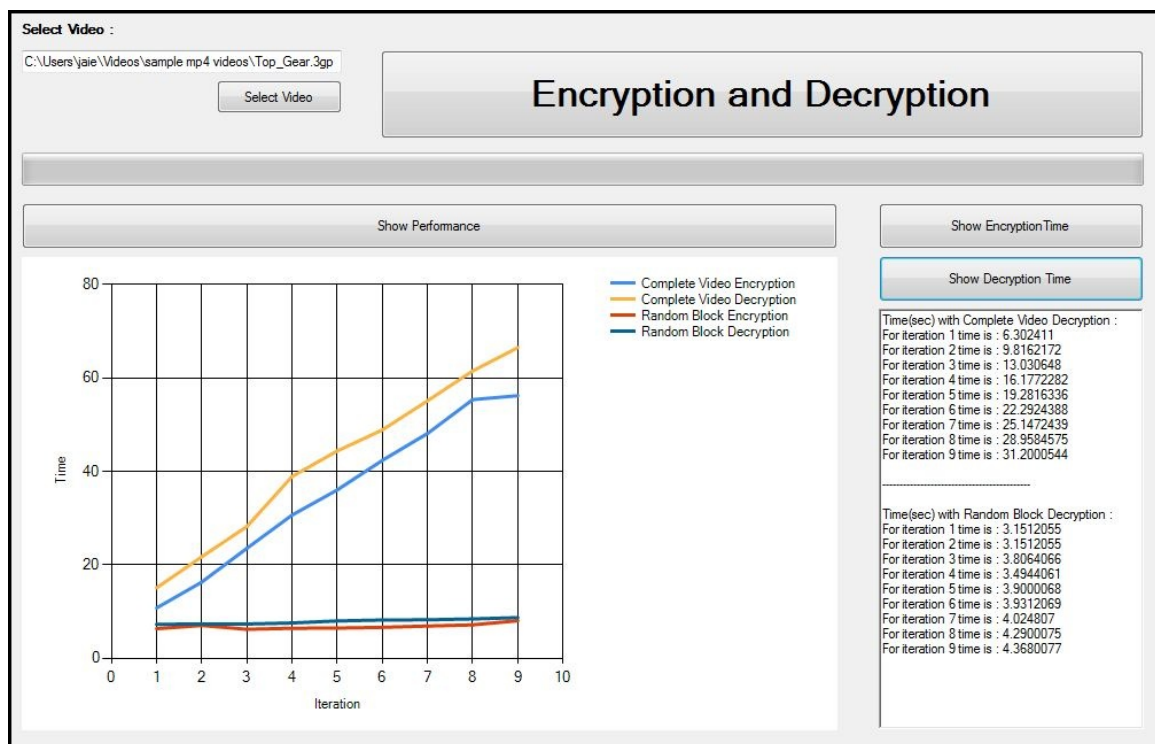


Figure 27: Comparison of Decryption Timings

CHAPTER 7

Conclusions and Future Work

An Android-based application “MediaStreaming” was created for the World Wide Web users to stream their choice of videos, securely. The application is supported through user authentication before accessing the videos available on the Web store. The video streaming design using security uses minimal processing with little overhead while maintaining security.

The authentication of each user is made strong by storing sensitive credentials for each user by using Salt in the database. The algorithm involved in authentication, is an RSA algorithm.

Encryption and decryption of videos are done via “Random Block Video Encryption” algorithm, which is based on AES. With an RBVE method, the sequence of encryption depends upon the Seed, therefore the attacker cannot find the encrypted and unencrypted blocks, creating a more secure, fast, and efficient video streaming.

The project has been constructed for easy integration and modification to take full advantage of future technologies. Few factors like bandwidth and video quality have not been taken into consideration during the development and performance testing of our application. As video streaming is managed via HTTP, the speed and efficiency also depend upon the network bandwidth. The application has been strictly created to work and execute on the Android Platform. Efforts should be made that an efficient application can be used across platforms such as Mac, Google, etc.

As of today, the application is capable of successfully encrypting and streaming

“.3pg” file formats. However, we have kept the application flexible enough to extend its functionalities to other file formats.

A perceptual hash algorithm extracts perceptual features from the multimedia content and achieves certain robustness. Though it allows, some amount of content-preserving processing, it is only sensitive to perceptually significant content modification. Perceptual hash should be studied as this approach implies possible breakthrough for multimedia authentication [17]. Many interesting issues related to scalable video coding, and its corresponding secure streaming mechanism need to be resolved so that more secure and adaptive media streaming can be achieved [17]. Also, during the response time, there are many other factors one needs to consider namely Network, video size, and so on.

The “Random Block Video Encryption” can be modified further using other encryption methods such as Hash key-based video encryption scheme for H.264\AVC, MPEG encryption algorithms, etc. A comparison and performance need to be tested for the best efficient algorithm implementation.

The project paper does not address the LIVE content streaming of videos. The scope could further be extended to the process of fetching LIVE video content securely by implementing an RBVE algorithm and performing user authentication, as this is an emerging factor within the field of video streaming.

LIST OF REFERENCES

- [1] Abomhara. M, Zakaria. O, & Khalifa. O, *An Overview of Video Encryption Techniques* International Journal of Computer Theory and Engineering, Vol. 2, No. 1 February, 2010, 1793-8201.
- [2] Agi. I, & Gong. L, *An emprical study of MPEG video transmissions*, in Proceedings of The Internet Society Symposium on Network and Distributed System Security, (San Diego, CA), pp. 137-144, February 1996.
- [3] Ali. W, & Nigam, A. *Java RTP Implementation*, Retrieved from http://www.cs.columbia.edu/~hgs/teaching/projects/java_rtp/report.html/
- [4] Apostolopoulos. D, & Wee. P, (2002). *Video Streaming: Concepts, Algorithms, and Systems*, Mobile and Media Systems Laboratory. 1-35.
- [5] Asghar. M, & Sadaf. S, (2010). *SVS - A Secure Scheme for Video Streaming Using SRTP, AES and DH*, pp 177-188.
- [6] Bhargava. B, & Shi. C, *An Efficient MPEG Video Encryption Algorithm*, IEEE Proceedings of the 17th Symposium on Reliable Distributed Systems, 1998, Pages 381-386.
- [7] Cheung. S, Ammar. M, & Li. X *On the use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution*, IEEE INFOCOM, March 1996.
- [8] Chiariglione. L, MPEG Technology Group, <http://www.chiariglione.org/mpeg>, (Accessed on March 2, 2009).
- [9] Girod. B, Chakareski. J, Kalman. M, Liang. Y, Setton. E, & Zhang. R, *Advances in Network-Adaptive Video Streaming*, 2002 Tyrrhenian Inter. Workshop on Digital Communications, September 2002.
- [10] Griwotz. C, *Video protection by partial content corruption*, Proceedings of Multimedia and Security Workshop at the 6th ACM International Multimedia Conference, (Bristol, England), pp. 37-39, 1998.
- [11] Holankar. D, (2011). *Secure Streaming Media and Digital Rights Management*. International Technical Paper, 1-4.
- [12] Huo. L, Fu. Q, Zou. Y & Gao. W, *Network adapted selective frame-dropping algorithm for streaming media*, IEEE Trans. Consumer Electron. 53 (2) (2007) 417-423.

- [13] Joshi, A, (2009). *How to setup Darwin streaming server on windows*, Retrieved from <http://generally.wordpress.com/2007/darwin-streaming-server-on-windows/>
- [14] Lian. S, *Multimedia Content Encryption: Techniques and Applications*. CRC, 2008.
- [15] Maples. T, & Spanos. G, *Performance study of selective encryption scheme for the security of networked real-time video*, in Proceedings of the 4th International Conference on Computer and Communications, Las Vegas, NV, 1995.
- [16] McCaffrey. J, *AES: Keeping Your Data Secure with Advance Encryption Standard*, <http://msdn.microsoft.com/en-us/magazine/cc164055.aspx>
- [17] Mou. L, & Huo. L, (2009). A secure media streaming mechanism combining encryption, authentication, and transcoding. *Signal Processing: Image Communication*, 825-833.
- [18] Paul. D, *How To create Streaming Video* <http://www.mediacollege.com/video/streaming/overview.html/>
- [19] Reibman. A, Jafarkhani. H, Wang. Y, Orchard. M, & Puri. R *Multiple-description video coding using motion-compensated temporal prediction*, *IEEE Trans. Circuits Syst. Video Technol.* 12 (2002) 193-204.
- [20] Rouse. M, *RSA Algorithm*, September 2005. <http://searchsecurity.techtarget.com/definition/RSA>
- [21] Schulzrinne. H, *RTP: A Transport Protocol for Real-Time Applications*, [http:// www.ietf.org/rfc/rfc3550.txt](http://www.ietf.org/rfc/rfc3550.txt), July2003/
- [22] Shi. C, & Bhargava. B, *A Fast MPEG Video Encryption Algorithm*, Proceedings of the 6th International Multimedia Conference, Bristol, UK, September 12-16, 1998.
- [23] Sirsalewala. M, *Application Infrastructure makes enterprises more productive*, (December, 2012) <http://www.networkmagazineindia.com/200212/inperson1.shtml>
- [24] Spanos. G, & Maples. B *Security for Real-Time MPEG Compressed Video in Distributed Multimedia Applications*, in Conference on Computers and Communications, 1996, pp. 72-78.
- [25] TEA, a tiny encryption algorithm, <http://www.ftp.cl.cam.ac.uk/ftp/papers/djw-rmn/djw-rmn-tea.html/>

- [26] *The Importance of Database Design in Web Designing*,
<http://www.eliteinfoworld.com/website-design-service/importance-of-database>
- [27] The Secure Real Time Transport Protocol, IETF draft,
<http://www.globecom.net/ietf/draft/draft-ietf-avt-srtp-00.html/>
- [28] Venkatramani. C, Westerink. P, Verscheure. O, & Frossard. P, *Securing Media for Adaptive Streaming*, in: ACM, Conference on Multimedia, November 2003.
- [29] Vetro. A, Christopoulos. C, & Sun. H, *Video transcoding architectures and techniques: an overview*, IEEE Signal Process. Mag. 20 (2) (2003) 18-29.
- [30] Vocal Technologies Ltd. *Advanced Encryption Standard(AES)*,
<http://www.vocal.com/cryptography/advanced-encryption-standard-aes/>
- [31] Wang. Y, Ostermann. J, & Y. Zhang, Video Processing and Communications, Prentice-Hall, New Jersey, 2002, pp. 368-393.
- [32] Web Server, http://en.wikipedia.org/wiki/Web_server
- [33] *What is .3gp file format*. Retrieved from,
<http://www.winxdvd.com/resource/3gp.htm>
- [34] Winkler. K,(2010). Just development. Retrieved from
<http://justdevelopment.blogspot.com/video-streaming-with-android-phone.html/>
- [35] Wu. D, Hou. Y, Zhu. W, Zhang. Y, & Peha. J, *Streaming Video over the Internet: Approaches and Directions*, IEEE Transactions on Circuits and Systems for Video Technology, March 2001.