T.C.

USKUDAR UNIVERSITY

INSTITUTE OF SCIENCE

CYBER SECURITY PROGRAM

MASTER'S THESIS

DESIGN AND IMPLEMENTATION OF SECURE-STREAMING SYSTEM WIT DYNAMIC QUALTIY MODIFICATION: AUDIO STREAMING APPLICATION

Hadeel Maher El Helou

Thesis Supervisor
Prof.Dr. Ihab ELAFF

ISTANBUL-2021

T.C.

USKUDAR UNIVERSITY

INSTITUTE OF SCIENCE

CYBER SECURITY PROGRAM

MASTER’S THESIS

DESIGN AND IMPLEMENTATION OF SECURE-STREAMING SYSTEM WIT DYNAMIC QUALTIY MODIFICATION: AUDIO STREAMING APPLICATION

Hadeel Maher El Helou

Thesis Supervisor Prof.Dr. Ihab ELAFF

ISTANBUL-2021

ABSTRACT

DESIGN AND IMPLEMENTATION OF SECURE-STREAMING SYSTEM WIT DYNAMIC QUALTIY MODIFICATION: AUDIO STREAMING APPLICATION

Media streaming is now a part of our lives and in the current and modern period, the use of media has become in all electronic systems and smart applications and many people depend on media in many aspects of their lives, Therefore, it is very important that the media be broadcast safely and in high quality, so the thesis is chosen to achieve this goal and is designed a protection system which is design and implementation of a secure broadcast system the application of audio streaming, focus on regular media streams and take care of the quality and each time the quality is improved so that this system of media broadcasting is not safe. Therefore, in our system, we want adding some features for this system to be secure and has good quality and this is achieved by using encryption algorithm. So is used RSA algorithm to achieve the goal and to add some characteristics to the media streaming system and be safe and protected by controlling parameters to balance security and good quality with real time.

Keywords: RSA algorithm, Media Streaming, Encryption, Decryption.

ACKNOWLEDGMENTS

First and foremost, I want to thank my god who helped us to continue my studied and support me in difficult time.

I am grateful in my supervisor and I would like to thank my advisor doctor Ihab Elaff who helped me in doing my thesis. He gave me important and useful advice and helped me through difficult times. This constant support helped me complete the thesis successfully.

On the other hand, I would like to say thank you to all the doctors in the College of Computer Engineering and Cyber Security who provided us with all the valuable information and gave us enough time to understand in the field of cyber security.

I dedicate my success and special thanks to my family, my mother, my father, my husband and my children, as they were the constant support in the most difficult times, Thanks for their understanding, love and encouragement through the tough times encountered during the production of this thesis.

THANKS AGAIN TO ALL WHO HELPED ME

DECLARATION FORM

I have obtained all the information and documents in this study within the framework of academic rules, I have presented all visual, audio, and written information and results in accordance with scientific ethical rules, that I have not made any falsification in the data I use, that I have referred to the sources I have used in accordance with scientific norms, that my thesis is original except for the cases cited. I declare that Written and I produce it according to Uskudar University Institute of Science Thesis Writing Guide.

28/12/2021

Hadeel Maher El Helou

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

PKCS : Public Key Cryptography System.

PSS : Probabilistic Signature Scheme.

OAEP : Optimal Asymmetric Encryption Padding.

DES : Data Encryption Standard.

AES : Advanced Encryption Standard.

IP : The initial permutation.

OTP : The One-Time Pad.

PCM : Pulse Code Modulation Analog-to-Digital Conversion.

DAC : Digital-to-analog conversion.

ADC : Analog-to-digital conversion.

DBFS : Decibels relative to full scale.

RTP : Real-Time Protocol.

RTCP : Real-Time control protocol.

RTSP : Real Time Streaming Protocol.

VoIP : Voice over IP.

UDP : User Datagram Protocol.

TCP : Transport control protocol.

SSL : Secure Socket Layer.

1. INTRODUCTION AND BACKGROUND

At this time in our life, we use the networks and the internet so much to communicate with others and we do not imagine our life without communication so the digital life is very important for us. So when communicate with others, this communication should be secure and safe. On the other hand, digital life has many risks if we do not protect ourselves from these risks, study, and analyses these risks such as attacking, so we should use cryptography to securely communicate without any problems. With the development and progress in the means of communications and the total reliance on the digital world, it has become necessary to protect the media, whether audio or video, from any attack and external penetration, so it became for technical people to study sufficient knowledge in order to be able to protect digital devices from any penetration and defect. Therefore, they must The use of fast and secure encryption techniques to protect the audio from a defect, there is always a need for a safer and faster audio encryption technology, in the previous period the focus was on encrypting and protecting text data and very few algorithms concerned with audio media and some of them were effective in the issue of quality well and others gave results Unsatisfactory in terms of security and quality requirements. Therefore, encryption of audio data is considered one of the difficult techniques that need adequate study to provide, so during my thesis focused on how to provide protection for audio systems while maintaining the audio flow in a way that preserves the real time streaming.

1.1 Introduction to Cryptography

Cryptography is the study and implementation of certain techniques to secure the process of communicating with other people such as attackers. So cryptography is concerned with creating and analyzing some system that protects the public from reading and hearing private messages. By transforming data from its natural and intelligible form to any person into an indistinct form so, that it is impossible for those without definite secret knowledge to know its content. The various fields of knowledge of information security and digital security must achieve confidentiality, integrity of information and authentication, the three objectives are basic aspects of encryption, and the system is protected, and use encryption in a number of sciences such as computer science, network science, electrical sciences and communications. Cryptography is one of the important and necessary sciences that must come within any field, as its practical applications vary to include several fields. The concept of encryption also includes the encryption of data and multimedia in order to ensure that data and media are delivered securely and of high quality to the other party so that no one changes and sees the information that passes through the Internet, so encryption focuses on three elements that must be taken care of and the use of techniques necessary to ensure the safety Therefore, when sending a message to the other party, there is a message, and this first party uses a key and a certain formula to encrypt this message, and then an encrypted message is formed from us. The message is sent to the second party and it consists of incomprehensible symbols and numbers so that it remains safe from any modification or knowledge by one party A third and then the encrypted message arrives and it understands and decrypts it using a secret key or through his key according to the type of algorithm used in the client (Qadir, Varol,2019).

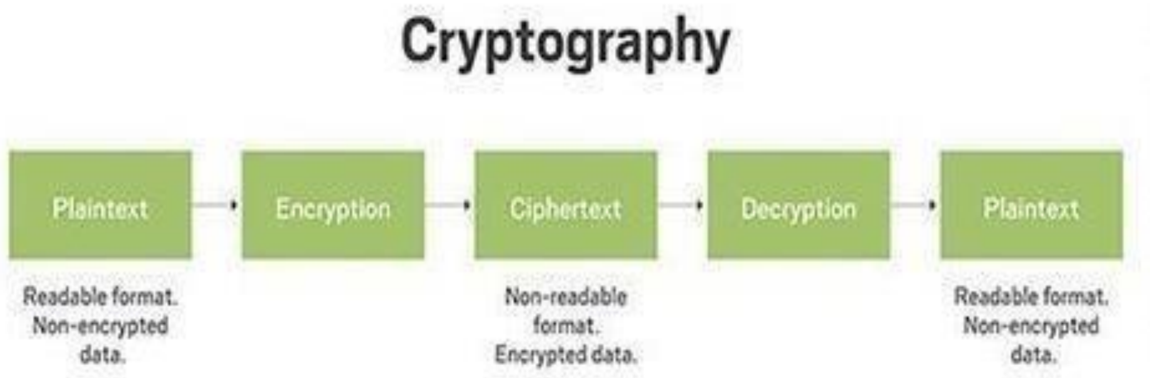


Figure 1: cryptography operation (Piper and Murphy,2009)

1.2 Cryptographic goals

The main goals of cryptographic, which are prevented the attackers to stealing the data (Schneier,2015).

- Confidentiality: A process that ensures the security of the data that passes through the network and reaches only the person authorized to receive the data, and here this is done by appropriate different encryption algorithms (Richards,2021).
- Integrity: A process by which we ensure that data is not modified deleted or added during its transmission over the network (Aumasson,2018).
- Authentication: A process that checks the identity of the real sender and then authenticates and allows him to pass the data (Gulzar, Nikki,2013).
- Non-repudiation: a process that ensures that the communication parties do not reject the transmission, so the sender must send the data and keys, and the recipient must receive the data and decode it using his private or subscriber key according to the type of algorithm used (Wang, Cong,2012).

1.3 Traditional cryptography

Types of traditional ciphers are ciphers switching ciphers, where rearranged characters are replaced and ambiguous ciphers are replaced, so that we replace these letters and numbers with other letters and numbers (Zager,2015).

1.6 Modern cryptography

Modern encryption algorithms depend largely on mathematical operations and computer science, and mathematical algorithms have been adopted due to the difficulty of breaking, penetrating and decoding them, which makes it difficult to break these algorithms in practice, but it is possible to lose this algorithm in theory, but in practice, it is difficult to break. Therefore these formulas are called mathematically safe; In accordance with theoretical advances in science, for example, these solutions require constant adaptation, with improvements in integer-parsing algorithms, and faster computing techniques. Secure systems exist according to information theory, safety in terms of information theory, and safety in terms of information theory, and have previously been shown to be unbreakable even with unlimited computing power.

1.5 Components of cryptography

The encryption system consists of several basic elements, including:

- Plain text: The original data that the sender sends to the recipient is called the real text
- Cipher text: data that is incomprehensible that is converted to the original text into cipher text.
- Encryption: converting the original text into an encrypted text using a special algorithm and an encryption key, whether it is public or private.
- Decryption: It is called converting the encrypted text to its original form using a key to decrypt.
- The Keys: numbers and letters used for encryption and decryption.

1.6 Types of cryptosystems

Cryptographic systems are divided into two systems, either symmetric or asymmetric.

1.6.1 Symmetric ciphers

In symmetric encryption algorithms, a single shared secret key is used to encrypt and decrypt the message (Oguntunde, Arekete, Odim, Olakanmi, ,2017) Examples of such algorithms are Blowfish, DES, AES, etc. Each algorithm uses a different method to encrypt and decrypt data; In addition, this type is used for decoding and decrypting data of a fixed size. In addition, the key used has a fixed size. Letters and numbers are used as plain, unencrypted original text. Then, after using the private key, an encrypted message is produced that reaches the other party for the other party to decrypt using the same key that was encrypted, and the symmetric encryption consists of:

- Original text: The original text that the sender wants to send to the recipient.
- Encryption Algorithm: The algorithm used and chosen by the sender to use its mechanisms to perform the encryption process, and uses a shared secret key to output an encrypted text.
- Secret key: It consists of letters, symbols, numbers, a certain length, and certain values.
- Encryption: The text that is output after using a certain algorithm and with the help of the shared key the cipher text is generated.
- Decryption algorithm: The process used to find out the content of the original text using the shared secret key.

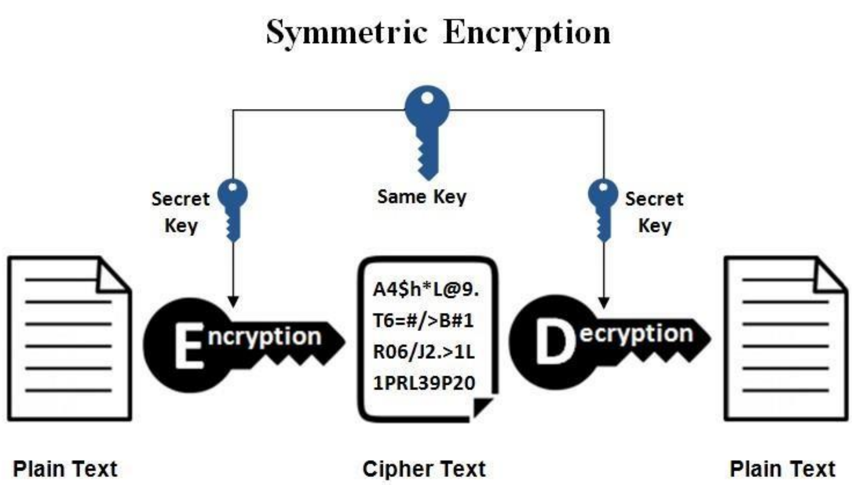


Figure 2: Symmetric encryption (Rivera,2020)

There are many advantages to symmetry coding and it has some disadvantages Advantages:

- Symmetric encryption is characterized by high data transmission and that in hardware applications, a large number of bytes per second can be encrypted, as well as software applications.
- The keys are relatively short.
- Effectively used in digital signature.

- Symmetric cipher can be used as a block to give a strong result and the algorithm used of this type becomes strong and difficult to decipher.
- This type was used in many digital computer inventions. Disadvantages:
- The key must remain secret from both the sender and receiver at all times and the transmitter.
- The key must be changed frequently to ensure data protection.
- The keys used are relatively short, and since it is used in the digital signature, relatively long keys must be used.
- You need high management of the shared key.

1.6.2 Public key ciphers or Asymmetric ciphers

Asymmetric key encryption is one of the techniques that uses two types of keys that are used to encrypt the encryption. Both keys are called the public key and the private key, where the sender uses the public key to encrypt the message and the receiver uses his own key to decipher the encrypted message. The recipient sends the public key through a private public key distribution channel. Asymmetric encryption uses mathematical formulas for encryption and decryption, but symmetric encryption uses a symbol substitution scheme. Asymmetric encryption is used in web services, digital signature, and authentication.

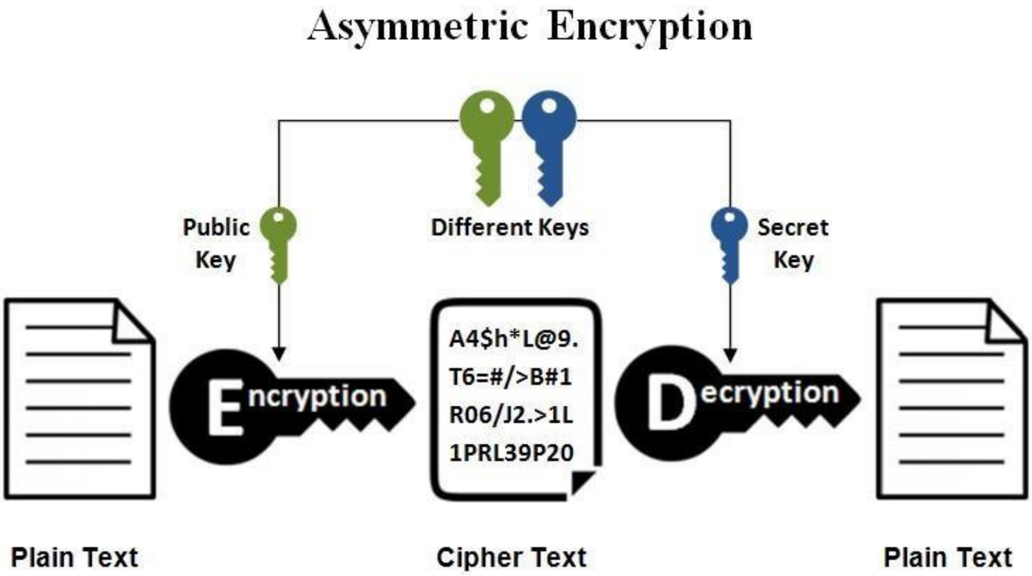


Figure 3: Asymmetric Encryption (Agrawal and Mishra, 2012)

1.7 Substitution techniques

1.7.1 Caesar Cipher

The ancient example of ciphers is the ciphers of Julius Caesar. This method uses a private key and symmetric cipher, where we offset each character in the original text by a fixed number of characters. Where we assign a number to each character and use an arithmetic equation for encryption and decryption For example, for encoding, we have to consider all the characters in the cipher text and the decrypted character where x is consider of the character in the cipher text, have to change the defined character with -n where n is the key and use mod 26 this number in advance the size of the English alphabet is 26 also we want to do the encrypted letter is within the range [0,size_alphabet-1] so this is why to use mod 26,the decryption side is opposite of decryption should shift the given letter with -n (where n is the key):x-n mod 26(Bokhari, Shallal,2016).

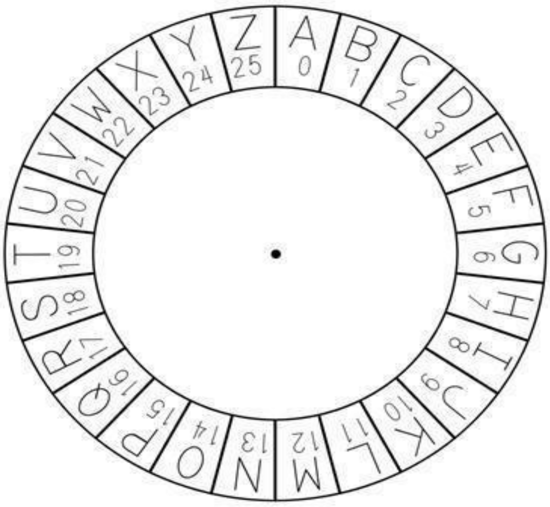


Figure 4: Caesar Cipher (brilliant.org,2017).

1.7.2 Substitution Ciphers Mon alphabetic

Cipher:

Single-alphabetic cipher ,in this cipher, the text characters are selected so that we can encrypt the characters according to one secret common alphabetic key, such as Caesar's cipher, and each character is moved based on the digital key (Jain, Chhibber, Kandi,2018).

Multi-alphabetic cipher:

Multi-alphabetic cipher ,this form of cipher uses the substitution method. The most famous example is the Vigenère cipher. It is considered a simple and easy cipher.

1.7.3 Vigenère cipher

This encryption is similar to the Caesar system, but it uses more than one key instead of one key, which is asymmetric encryption, and the encryption method is similar to the Caesar method and is a form of multi-character substitution.

1.7.4 Play Fair Blades

In this encoding, we use an array5 *5 of its values, the original text is inserted into the array, and the rest of the alphabet is added to the rest of the array. The repeating character is not added, but is added once, and we need 25 character instead of 26 (Khan,2015).

1.7.5 One-Time Pad

The One Time Pad (OTP) is a powerful encryption algorithm that cannot be broken, and it is a symmetric encryption system that needs one secret shared key, and the key is attached to the text, and each character is encrypted using the method of combining the character with the bit in the shared key and adding it in a standard way (Singh, Athithan and Pillai, 2021). The encryption method is strong and it is difficult to break and decrypt if the key contains several things, including:

- The key is random
- The length of the key must be the length of the original text
- Not to be used twice, either in whole or in part
- The key must be kept secret and secure by the parties

1.7.6 DES Algorithm

DES Algorithm This method is considered a symmetric encryption method that uses one shared secret key and the plain text is 64-bit and the original text is converted into ciphertext using a shared key length of 48 bits(Kaur, Sodhi,2016)This algorithm is based on the Festal block cipher, as it contains 16 rounds and is used in each round a different key. Another type of this type appeared, Triple DES, in which encryption is done with a key, which is the first key, and decryption is done using the second key, and it is decrypted again with the third key, and there are two variable keys as well.

NIST replaced this algorithm because the length of its keys is 56 bits, which is small because the encryption strength depends on the size of the key, so it has become very rare to use this algorithm from serious updates to the encryption (Ratnadewi, Adhie, Hutamal , Ahmar,2018). This is for DES, but it still uses Triple DES, but its encryption is considered old (Saikumar, 2017) Now I will explain what DES is and how it works in the beginning:

- Plain text must be 64 bits long and converted to 64 bits cipher text by first permutation
- We switch the first on the original text.
- By permutation the first two halves are created, a part named Left Plain and the other Right Plain Text.
- The two particles go through 16 rounds
- The right and left parts are joined and the final stirring is done.
- A 64-bit cipher text is generated. There are five steps for encryption process.

The algorithm process breaks down into the following steps:

- The process begins with the 64-bit plain text block being handed over to an initial permutation (IP) function.
- The initial permutation (IP) is then performed on the plain text.
- Next, the initial permutation (IP) creates two halves of the permuted block, referred to as Left Plain Text (LPT) and Right Plain Text (RPT).
- Each LPT and RPT goes through 16 rounds of the encryption process.
- Finally, the LPT and RPT are joined, and a Final Permutation (FP) is performed on the newly combined block.
- The result of this process produces the desired 64-bit cipher text.

The encryption process step (step four, above) is further broken down into five stages:

- Shift Key
- Expansion permutation
- S Box
- P Box
- XOR and flip

For decryption, same encryption steps but in reverse.

1.7.7 AES Algorithm

AES is this type of encryption developed by Rijndael Block Cryptography. This type of encryption has different key sizes, using 128, 192, and 256 bits (Abdullah,2017).

It is newer than DES encryption and is characterized as symmetric encryption based on only one key. This means that encryption and decryption use the same key. This encryption is one of the new widely used and popular operating systems and it is faster and faster than DES as it was used instead of DES, which is considered weak due to its small size, key, and AES encryption features:

It is a symmetric key and has different key sizes such as:

- It contain of 128,192,256 bits
- It is faster and more powerful compared to DES and Triple DES.
- It uses different languages such as Java and C.

1.7.8 Operation of AES

AES is a cracked cipher and the substitution switch network consists of a set of overlapping processes, one input is replaced by specific substitutions and the other is bit shuffling and also depends on the byte. The original text handles 16 bytes for 10 rounds and uses12 rounds for 192-bit keys and 14 rounds for 256-bit keys.

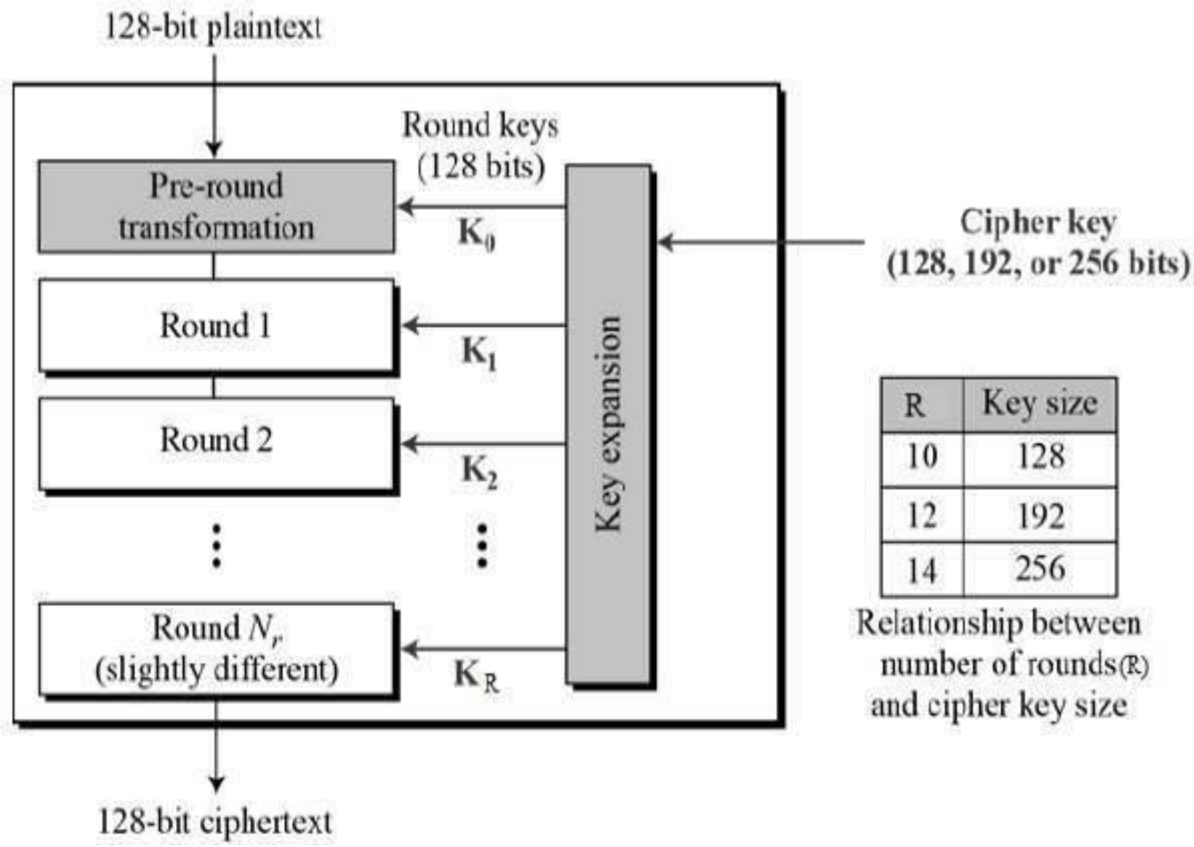


Figure 5: AES Operation (JADHAV,2019).

1.7.9 Encryption

The encryption process in AES consists of four stages:

- **Byte substitution**

Here 16 bytes are entered, the S-box table is used, and the array consists of four columns and four rows.

- **Row shifting:** in this stage, the four rows are shifted to the left

The transformation is performed as follows: the first row is the same, we have not moved any element, the second row is moved one byte to the left, the third row is left, and the fourth row is moved three times to the left

 Mix the columns Add a round key

The 16-byte array is now 128 bits and XOR is converted to 128 bits of the circular key. If this is the last round, the output will be the cipher text. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we start another similar round.

1.7.10 Decryption

The decoding process is similar to the decryption process, but it is reversed and consists of four stages:

- The first stage is to use and add a round key
- The second stage is a combination of columns
- The third stage is Shift rows
- The fourth stage is where the byte substitution It is better to encrypt and decrypt separately.

1.7.11 RSA Algorithm

RSA (Rivest-Shamir-Adleman) is an acronym, This algorithm is an encryption system that uses asymmetric encryption and is widely used to transfer data (Nisha, Farik,2017). This encryption system is one of the safe, easy and strong systems. This type of key uses two types of keys, a private key and a public key, and it is characterized by strength and confidentiality due to its dependence on Two keys, so there are many modern systems that use it, and mathematical equations are used, and it is characterized by the difficulty of analysis. This is why it is considered one of the safest ways to transfer media and data. This algorithm consists of four stages:

- Key generation: In this stage, the sender generates a public key and a private key by using two different prime numbers, preferably large, to increase security $n = p * q$, since n is characterized by its length and number of bits, and this is what distinguishes this type of encryption that it can control its length n to increase protection and the difficulty of penetration. Then we know the public key through a mathematical equation and the private key as well

- Key distribution: In this stage, each of the sender sends the message using the public key e of the receiver, and the receiver decodes the message using his private key, and the private key remains private and is not distributed at all.

- Encryption: At this stage, after the sender takes the public key, he encrypts the message and uses a mathematical equation to calculate the cipher text.so sender used the public key of receiver PU = (e, n).

$$c = m^e \ (mod\ n)$$

- Decryption: After the receiver receives the message, it decrypts it using his private key and using the following mathematical equation. So, for decrypt the cipher text receiver uses their private key PR= (d,n).

$$m = c^d \ (mod\ n)$$

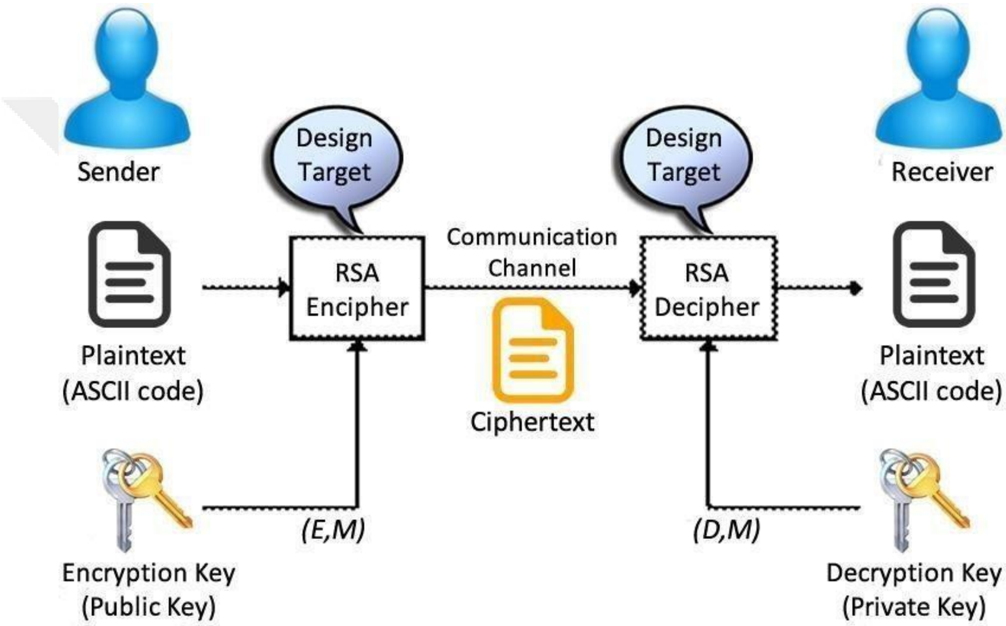


Figure 6: RSA Operation (Bodur, Kara,2015)

Example for RSA process:

In the first step: generating the keys, the public key and the private key We choose two first dissimilar numbers: P = 53 and Q = 59.

Next, we calculate: n = P * Q = 3127.

Now we want to compute the public key e must be e, an integer, not a factor of n.

1 < e < (n), and the greatest common divisor must satisfy one for each of the public key and

Φ (n)

GCD (e, Φ (n)) = 1, and then we calculate Φ (n): so Φ (n) = (P-1) (x-1) then, Φ (n) = 3016.

The second step is to generate the private key:

Calculating the private key: d * e mod Φ (n) = 1 then d = e ^ (- 1) mod Φ (n) d = (k * Φ (n) + 1) / e for some integer k for k = 2, the value of d is 2011.

Using our public key (n = 3127 and e = 3) and the private key (d = 2011)

The last step is now encryption and decryption, we want to send this number after convert it to ASCCI number: 89

$$\boldsymbol{c = m^{e}\ (mod\ n)}$$

So the result is encrypted data is 1394, now we want will decrypt 1394.

$$\boldsymbol{m = c^{d}\ (mod\ n)}$$

After used this equation obtained the plain text equal 89.

1.7.12 Potential RSA Attacks

Despite the strength of this type of encryption, it encounters problems when

Initially, when using encryption for small values of the public key, the cipher text can be easily decrypted, so it is recommended to use an appropriate public key.

When the communication between more than one person and all the sender uses the same public key to encrypt the message and send it to all recipients, it is easy to decrypt and know the original message, so it is recommended to change the public key and use a public key assigned to each recipient

Since this type of encryption does not have a random component, the original text can be hacked by encrypting the original text and seeing if it is equal to the cipher text. This is done by knowing the general character and appearance of the message and guessing some of the words at the beginning and end of the message. Therefore, it is recommended to use padding to ensure that the hacker does not overwrite the text. The message can be easily decoded (Lone and Uddin,2016).

By knowing the private key, it is possible to know the first two numbers used in the beginning, so the first two numbers must be chosen accurately. Therefore, if you have the private key, it will be easy to find out the public key.

1.7.13 RSA Protection

The security and protection of the RSA algorithm is one of the important matters that depends on the understanding of this algorithm. One of the most important elements used to implement the security element is the size of the key, as the larger the key size, the more difficult it is to hack, meaning the greater the number of bits, so the keys used must be long and somewhat complex because it is difficult to penetrate and achieve element of safety.

1.7.14 Present and future importance of RSA

It is very important to know that RSA is secure and protected despite some of its few hacks, so it must be used correctly and the exact calculations selected for it. Therefore, care must be taken to use appropriate prime numbers, and the length of the public key is appropriate and somewhat large so that it is difficult to penetrate, and the padding is used in the message, whether in the beginning Or the latest applications that do not take into account these things. These things are considered somewhat unsafe and easy to penetrate. It is recommended to use a key with a length of not less than 1024 bits, and there are other strong keys of 2048 bits in length and others (Coron ,Joye, Naccach and Joye,2004).

Now I will talk about why RSA was chosen and why I chose it for sure because it contains many features and functions that distinguish it from others, including:

- Confidentiality: because this type of encryption is characterized by having high secrecy, as the content of the message moves only between the sender and the receiver only.
- Integrity: This feature is present in this type so that it does not specify any modification, addition or deletion of the data during the transfer process.
- Authentication: This feature is very important because in RSA the sender guarantees the sent information because only the receiver can decrypt it using his private key and the sender encrypts the message using the public key of the receiver also to ensure and check the recipient's identity by sending him the public key in order to encrypt the message and send it to the receiver.
- Non-repudiation: The sender cannot encrypt the message with his private key, but by using the public key of the receiver, because the private key is not interchangeable and private.
- This type of encoder has a variety of key sizes.
- RSA encryption is used to digitally sign messages, and it was recently used in digital currencies with digital signature for messages and the use of the public key of the

receiver and decryption using the recipient's private key through a certain value of the receiver and the receiver compares this value and checks this value.

1.7.15 Hash function

Hash functions are important and secure algorithms that are used in the security of information and applications. As it is a mathematical function that converts a specific numerical value into a numerical fixed size, and the output of the function is always fixed, as the return values are called hash values(Geetha ,Gupta,2020).

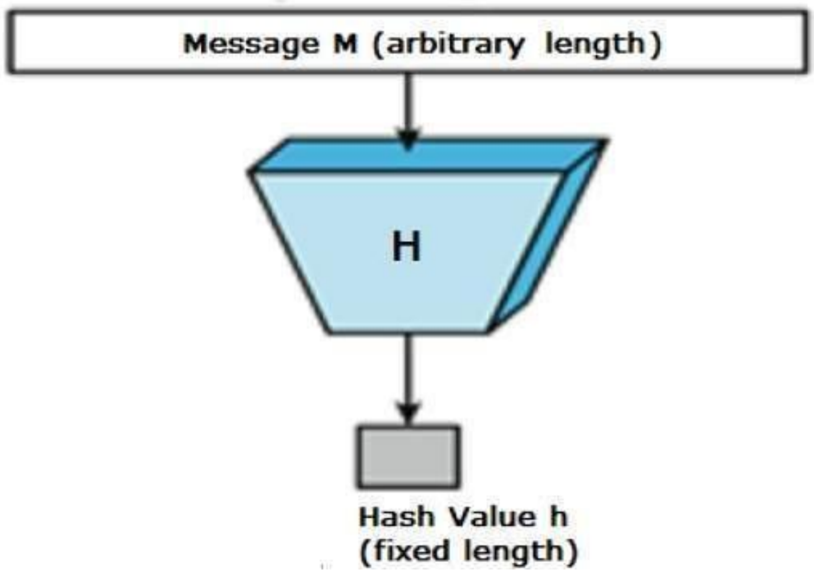


Figure 7: Hash function operation (Sobti1, Geetha, 2012)

Among the features and characteristics of this function are:

- The output message has a fixed size
- The function segments the message to give a fixed size.
- The hash of the message is much smaller than the input data.

A hash function that outputs a certain number of bits is considered a hash function and the output message length ranges constant 512 bits. Examples of Hash functions: MD5, SH512.

1.7.16 Features and characteristics of the hash function

It is important for this function to have an effective encoding that it possesses a number of functions and properties, including:

- Pre-image resistance: It is not possible to invert the hash function, meaning if the input function is h, the hash value will be z, this feature protects from any attacker who is trying to know the input.

- The second characteristic is the second resistance before the image

Through our knowledge of the inputs and the special segmentation, it is difficult and not possible to know other inputs with the same segmentation.

This means that if the hash function h for input x produces a hash value of h (x), it is difficult to find another input value of y that is h (y) = h (x)

- Collision resistance: This feature means that it is not possible to find two different inputs of any length that have the same hash. This means that this function is free from any collision.

Since the hash function outputs of the message a fixed size, it is possible to cancel collisions and it is difficult to find collisions in this function

This feature makes it impossible for the attacker to hack to know the value of the input Since the hash function compresses a function with a fixed hash length, it is impossible for the hash function not to collide. This collision-free property only confirms that these collisions are hard to find.

This property makes it very difficult for an attacker to find two input values with the same hash.

1.8 Thesis Objectives

Regular media streaming care only about adjust quality for real-time response; however, data exchange is not secured. In many cases, media streaming required to be secured between the sender and the receiver to provide users with more privacy. Cyphering the media will add another component to the system which will affect its performance so media will not have a real-time response. The main objective of this research is to add security component to the streamed media with dynamic balance between security and quality that keeps the real-time response of the system where audio streaming is taking as the case study. Thesis chapters are arranged as follow:

Chapter 1 provides an overview of cryptography Chapter 2

provides detailed media streaming.

Chapter 3 provides details about system design

Chapter 4 shows systems results

Chapter 5 contains conclusion and future work

2. MEDIA STREAMING

2.1 Introduction to Streaming Media

With the tremendous technological development and dependence on digital systems in our time, the dependence on the use of multimedia has become through the use of applications and the use of websites on the Internet.

First, the main reason is due to the higher demand for networked storage and bandwidth compared to the old, low and tiring storage methods.

Second, the facilities and improvements that exist on the Internet, such as high speed, high quality, and protocols used for quality control and control.

Third, due to the high demands of high-quality media and the use of applications that combine two technologies, namely television and the web, which are live broadcasting and streaming video and audio on demand, this requires high storage space, high quality and compressed files that will be saved on the servers. There are servers on networks designed to stream live audio data, such as TV and video programs such as Converse, and there are servers that support ondemand media streaming, such as YouTube, Zoom, Messenger and other software and applications used to transmit voice and media through them. At the request of the recipient, the server sends him the appropriate file, through the network communication layer TCP and UDP, before sending it to the network. In addition, a one-way transmission takes place in one direction over the network and uses special software to voice through smart devices and tablets. Computers are equipped and prepared to stream this audio using digital media hubs as well, and the audio is either wired or wirelessly streamed through speakers, as well as network audio stores so that when you want to play at any time, the audio file is there and so is the server and they work together Some to output and improve the output sound when listening to it if it is of low quality, by providing the server with feedback on quality and time and the client stores the audio in buffers for a few seconds before playing the audio through the speakers, so there is no audio delay (Kretschmer,2015). VoIP is one of the most widespread protocols on the network, which is intended for streaming voice over the network, and performs two-way transmission while maintaining real-time streaming. It also uses a wide bandwidth. An example of this is a proper audio stream, Windows Media.

The most important streaming media protocols are:

- Real Time Protocol (RTP): Defines the correct format of the data packet for transmitting audio and video over the network. It processes the real time transmission so that the client starts receiving the audio in a few seconds.

- Also, this protocol is based on RTP:
- Define and set the type of payload
- Package serial number
- Timestamp: The first bit of the sampling instance in RTP packets.
- RTP packets encapsulated in UDP segments
- This protocol is located in the transport layer and is an extension of UDP functions
- RTCP (Real Time Control Protocol): The function of this protocol is to give feedback which is quality of service reports and information about the recipient for the audio files in the current session
- RTCP also collects information about lost packets, vibration, lack of quality and delay in single and dual networks.
- Real Time Streaming Protocol (RTSP): This is a protocol used for user interaction such as allowing fast forward functions to be turned on and off, and also telling the RTSP used to request setup to the server how to transfer media and which port to use.

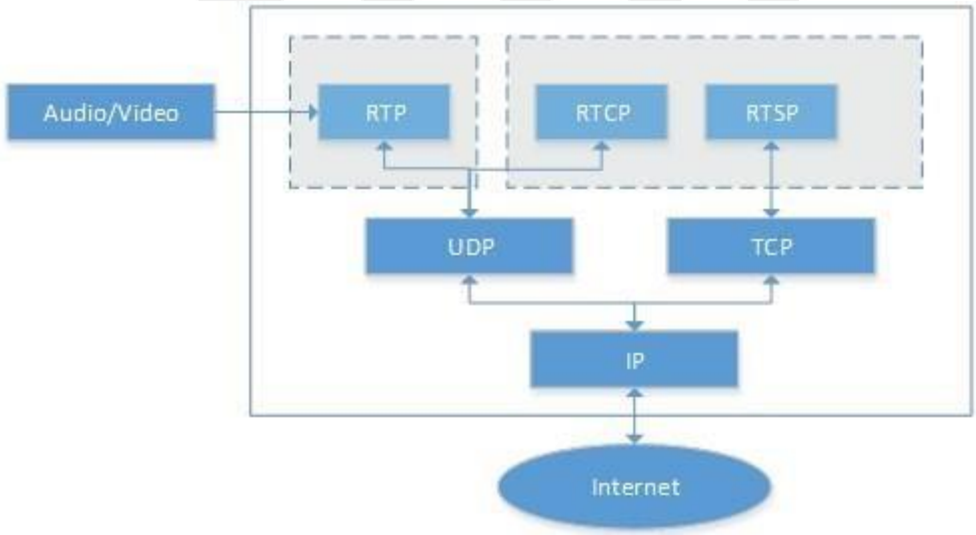


Figure 8: Media Streaming Protocols (Durresi, Jain, 2005).

Many people use the Internet to listen to a variety of audio media, whether educational or entertainment, and watch videos. Because audio or video playback is not present in web clients, it has become necessary to have an application to play media and it is called a media player, decompress the audio and video, as the audio is compressed to provide enough storage space and a sufficient frequency range. Once the media player is running, the compression is decompressed for the media during playback.

Users request through a web client to stream audio and video. However, since audio/video playback is not directly integrated into today's web clients, a separate assistant app is required for audio/video playback. The auxiliary application is often called a media player, the most popular of which is currently Real Networks, and the most common of them is Real Players, so it is necessary to know the functions of media player:

- Decompress: Audio and video are compressed to save disk storage space and network bandwidth. The media player must decompress audio and video during playback.
- Elimination of the sound vibration that appears due to the delay of the beams, which leads to the occurrence of vibrations in the beams. The problem can be resolved by using audio and video buffering before playback.
- Correcting and modifying errors that result when packets are lost, and the loss of packets is large, by rebuilding the lost packets and sending them again, and this must be done by the server to return the lost packets due to congestion on the network.
- It is an interactive interface for the user because of the available icons for interaction, such as pause, play, volume control, and others.

2.2 Streaming Media Player Requirements

One of the common types of media streaming is either recorded or live, in addition to media such as audio with images. It is a requirement for media playback that the user has a player program to be able to open the media such as Windows Media Player

An important requirement to deliver a large audio stream is that a bandwidth is required for the audio to be of high quality and technology in transmission as needed

- Server vs. Server less Delivery: With this requirement, media files are received through a special streaming server and the audio can be uploaded and delivered unlike a dedicated streaming server.
- Depending on the current web server activity and this depends on the congestion on the network. If there is congestion, the server activity is not good. It relies on sending media over the World Wide Web in real time

- Unicast: This use requires a wide bandwidth to create a two-way system. The server sends the audio file to the user upon his request and allows the user to use multiple functions such as pause and play.

- Broadcasting: This type is sent from the server to a group of users, whether they need a file or not, and it is completely sent to the network users.
- Multicast: sends data to customers who want the audio file, and here it does not need a wide bandwidth.
- Broadcasting over the Internet: The network is used to broadcast, whether audio or video, live or recorded.
- Embed: This requirement places the media source in a file and the file cannot be modified or edited to place the source within a document also The source cannot be edited.

.

2. 3 Audio sampling

Digital audio is one of the systems that are used to store and process audio information, and there are some characteristics of the sound wave such as amplitude and frequency, and when using a computer or communication device, the audio is converted from a continuous wave to digital, through programs dedicated to sound, as these programs allow us to control The sound is greatly enhanced by recording time, amplitude, and frequency, monitoring the digital wave movement over the network or on a computer to enhance the sound and converting the transmitted information into understandable information and processing the sound wave (STUART,2007), The second user converts the signal into a continuous signal into a discrete time signal. This approximation of the discontinuous signals largely ensures that the sound arrives as in a continuous wave, that is, the original signals of the sound are known through a series of samples until they reach the maximum wavelength, the Nyquist limit.

For pulse code modulation, digital audio (PCM) and digital signals are used to reproduce the sound. It performs analog-to-digital conversion (ADC) and digital-to-analog conversion (DAC), as well as storage and transmission. The system generally referred to as the digital system is actually a separate analog of time and separate from the previous electrical counterpart. While modern systems can be very precise in their methods, the primary task of digital systems is to ensure that signals are sent and received without loss of quality and time.

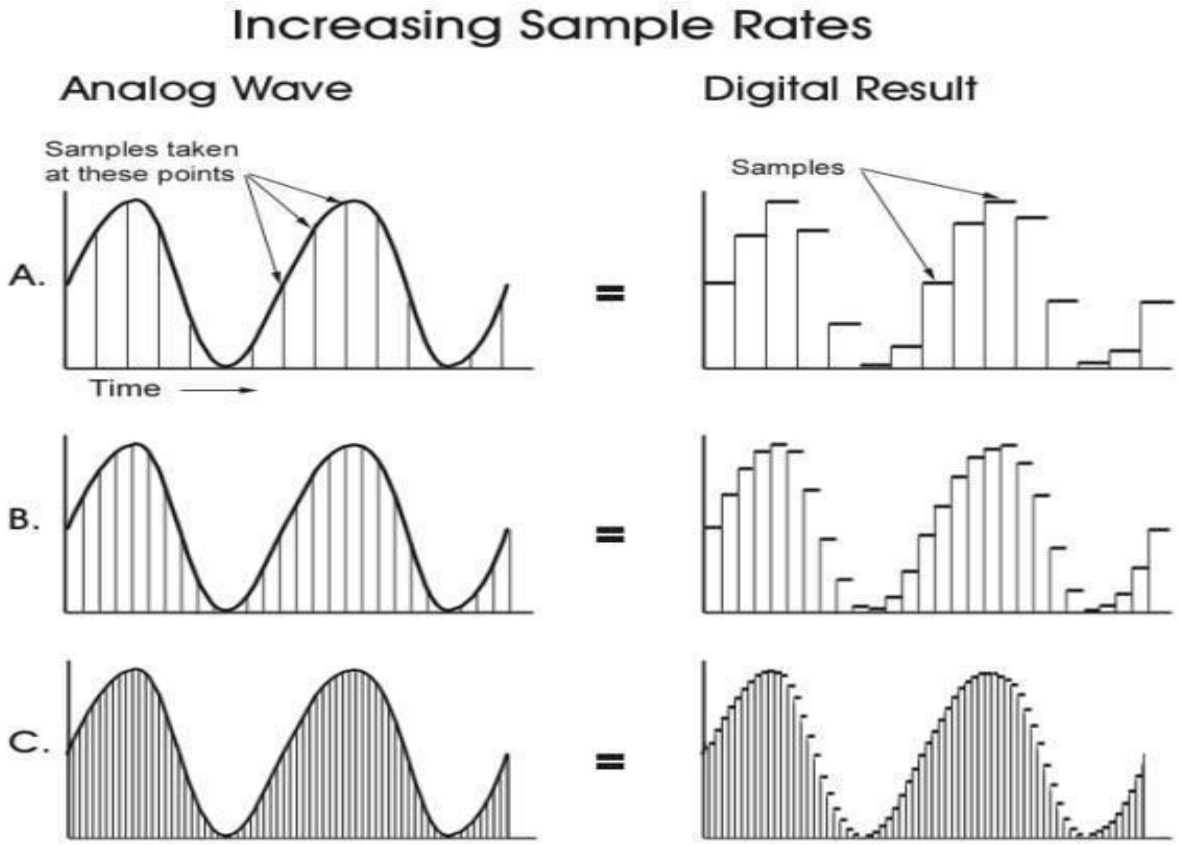


Figure 9: Audio sampling (Brown,2021) 2.4 Sample rate

Sample rate can be defined as the number of points transmitted per second, measured in Hz or kilohertz, or how many points carried per second we need to build a digital wave Examples of sample rate 44100 samples per second as either 44100 Hz or 44.1 kHz (Katabi,2003).

Bandwidth is the highest amount of data that can be sent over the network. This data is considered continuous and analytical, and through it, the difference between the highest and lowest frequency can be known. The maximum frequency is half the sample rate, known as Nyquist, which is the upper limit of the frequency that is at the middle of the sample. The bandwidth can be applied and used on the frequency content For streaming audio signals or on audio programs, it supports different frequencies from 80 kHz to 192 kHz.

These measurements and numbers in kilohertz are called the sound sample rate, and since it is possible to modify the sound samples according to the quality and the allowed bandwidth, there are low rates for audio files such as 22050Hz, 48kHz,192kHz, the well-known and most widely used, and others. Determines the wavelength, which is the wavelength, so the wave must be measured twice to determine the wavelength with high accuracy Audio sample rate, the system takes these measurements at a speed called the audio sample rate, and is measured in kilohertz. The audio sample rate determines the range of frequencies captured in digital audio. On most digital audio workstations, you will find an adjustable sample rate in your audio preference.

This controls the audio sample rate in your project. The options you would see in an average DAW between 22050 kHz sample rate were reasonably common for lowbitrate MP3s, 32000Hz, 44.1 kHz, 48 kHz, and might seem a bit random. To measure the frequency of this sine wave, we need to be able to detect and define a single cycle. The full cycle of any wave contains a positive and a negative phase. To find out the length of this cycle, the wavelength, which leads us to the frequency of the wave, we need to detect these two phases. Therefore, we need to measure the wave at least twice for each full cycle to accurately capture its frequency. There are also low-pass filters that block and remove frequencies above the Nyquist that connects to the transducer that high frequencies may cause distortion.

2.5 Audio bit depth

Analog sound can be defined as a continuous wave that shows a very large number of values. In the digital wave, we need sufficient values in each sample to reach the shape of the analog sample and read the wave without distortion. Therefore, the depth of the sound bit says by specifying the number of the seeking section for one sample that can be stored and using programs Audio We specify the bit depth, so it is 8 bits, 16 bits, 24 bits and 32 bits, and the values can be calculated by 2^n(n-bit). Examples of values used are bit depth. For example, when want to send.

• 8 bits:2^8,   256 values

• 16 bits:2^{16} ,65,536 values

• 24 bits: 2^{24},16,777,216 values

• 32-bit: 2^{32}, 4,294,967,296 values

The higher sound depth we use, the more accurate and quality the sound becomes, and the closer we get to the correct analog wave

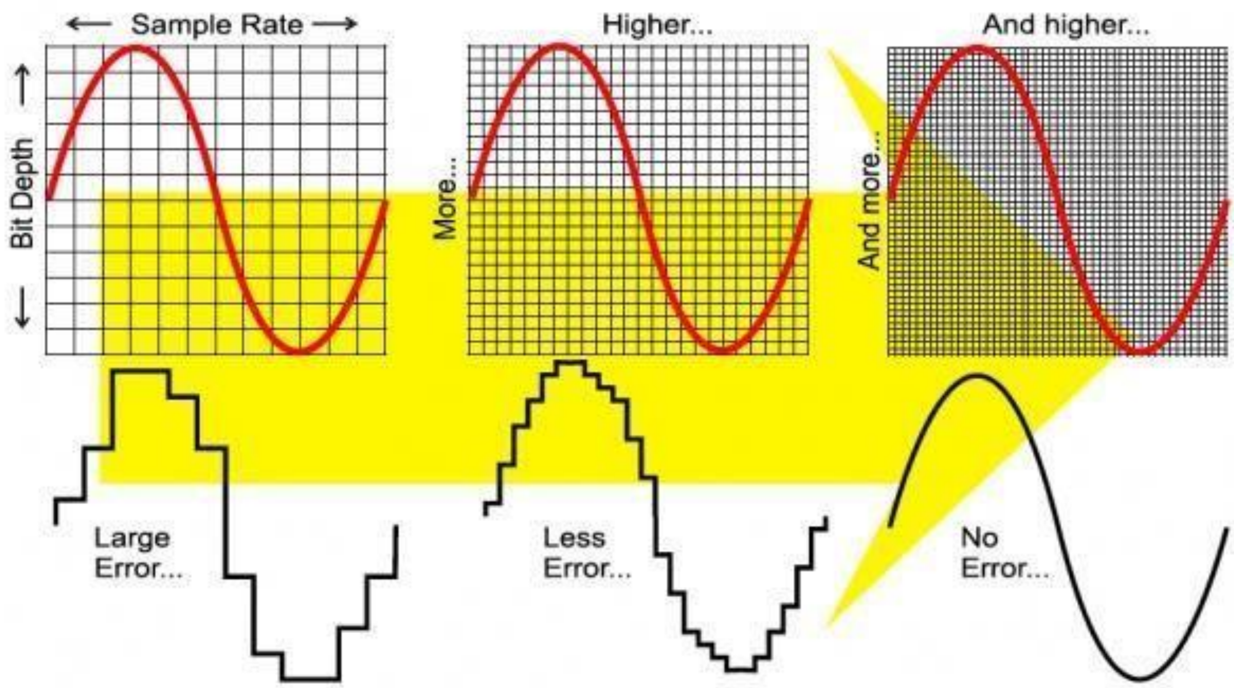


Figure 10: Relation Sample rate and bit depth (Brown,2021).

8-bit: 256 amps. Value

• 16 bits: 65,536 amps. Value

• 24 bits: 16,777,217 amps. Value

• 32 bits: 4,284,967,296 amps. Value

It also uses a higher bit depth of audio along with an audio sample rate that increases the points to build a digital wave very close to an analog wave

But the analog wave does not always agree with the potential peak if we increase the number of bits so that the last bit in the data is rounded to either 0 or 1 and there is a little randomness in the digital sound and it is called white ground noise because it is mechanical noise created by background noise to prepare the real sound One of the reasons that causes noise is the relationship between sample rate, bit depth, and sound. It causes correlated noise or ground noise at a certain frequency. In fact, the noise floor is high, so this noise can be resolved by random distribution to ensure that it occurs in the frequency process. A random approximation can be used as the digital system can it distorts if the amplitude of the router is high and the maximum value created by the dual system and the audio bit depth is set exceeds the number of amplitude values between ground noise and 0 dBFS.

2.6 Previous Research in Audio Streaming

The Research talked about encryption and decryption of audio signal based on RSA algorithm, this research was authored by (Yousif,2018) and was used RSA, The results obtained demonstrated that the residual intelligibility in the encrypted audio signal is low while the quality of the recovered audio signal is maintaining good with a satisfying level which confirm the suitability, reliability, high security and effectiveness of the introduced scheme to be applied in practical applications like audio data encryption and decryption .

Also, There is another search talking about a survey on different encryption techniques for image ,video, audio and docs ,this research was authored by (Poduval, Rai, Khan, Sane,2021) the method was used AES,DES,RSA AND BLOWFISH, The results show that RSA uses a lot time encryption and decryption compared to others. Blowfish spend the least amount of time, The results show that AES requires the highest number of bits to be encrypted and DES requires the least number of bits to be encrypted correct.

In addition, there is search talking about Performance Analysis of RSA and ElGamal for Audio Security, this research was authored Tin zar nwei, Su wai phyo , he was used Algammal and RSA and the results encryption and decryption time of RSA algorithm are significantly faster than ElGamal algorithm and about Real-Time Encryption/Decryption of Audio Signal , this research was authored (NWE, WAIPHYO,2014) and the method was used RSA and the results , the current paper applied and evaluated two different encryption/decryption algorithms to a real-time audio cryptography.

also Real-Time Encryption and Decryption of Audio Signal research was authored (Khalil,2016) , the method was used RSA, the results the current paper applied and evaluated two different encryption/decryption algorithms to a real-time audio cryptography.

and Study on Current Scenario of Audio Encryption research , this research was authored (Gandhi, Gosai, 2015) ,the method was used DES, 3DES, AES, Blowfish, RC4 ,the results Design and implementation of a new algorithm for audio encryption Based on above parameters.

The research talked about encrypt audio file using speech audio file As a key , was authored (Hassan, Al-Mukhtar and Ali,2020) , the method was used AES, the results for all tested audio files shows that the proposed algorithm for audio files is secured because of its, uniform histograms, large keys space, low correlation, and low PSNR, showed that the proposed algorithm for encryption of audio file is a very good choice in same time is a very good security to audio transmission.

Also the research about literature survey on recent audio encryption techniques , this research was authored (Srividya,Sudha,2016) the method was used DES, 3DES, RC2, RC4, RC6, Blowfish the results Audio file is considered as a stream and encryption is dependent on both data and key. Brute force attack is impossible on encrypted audio file, which are typically large. It is also resistive to statistical attacks. It is not suitable for low quality audio files as they will be prone to statistical attacks.

2.7 Problem Definition

Regular media steaming system consist of three stages, first stage is sampling of media, second stage is transfer the media through the network and the third stage is play media, but these system are not secure because there no any king encryption and decryption in communications sides. Therefore, for the system to be more secure, two blocks has been added: the first block consider of encryption of data after sampling and the second is for decryption of data before playing. For the proposed, system two properties are in concern namely: streaming quality and security of that streaming. The security component will add extra effort on both the server and the client for encryption and decryption respectively. For keeping the system keep streaming data in real-time with the good efficiency of the encryption, this is necessary to conserve and achieve a balance between the security and the quality.

3. SYSTEM DESIGN AND IMPLEMENTATION

3.1 System Overview

In this section of this chapter, it has been explained how the idea of the thesis can be implemented to achieve and add a new feature to the streaming audio media systems, which is a new security system so that the system becomes a high quality secured system.

In a regular media streaming system consists of three stages, the first stage is sampling the media, the second stage is the transmission of media over the network and the third stage is media playback, but these systems are not secure because it does not have any kind of secure encryption and decryption ,therefore, in order for the system to become more secure, we must add two blocks, the first block add encryption and add it after data sampling and add the decryption block after the media is transmitted over the network so that the system is safe and secure during the media flow. Regular streaming system cares about improving the quality only like YouTube response, the broadcast quality is according to server condition, network condition and client condition, so there are three factors that affect streaming quality so YouTube cares about quality. For our system, two things and two goals, we care and are working to improve the broadcast quality as well as improve the security system. See Figure 3.1.

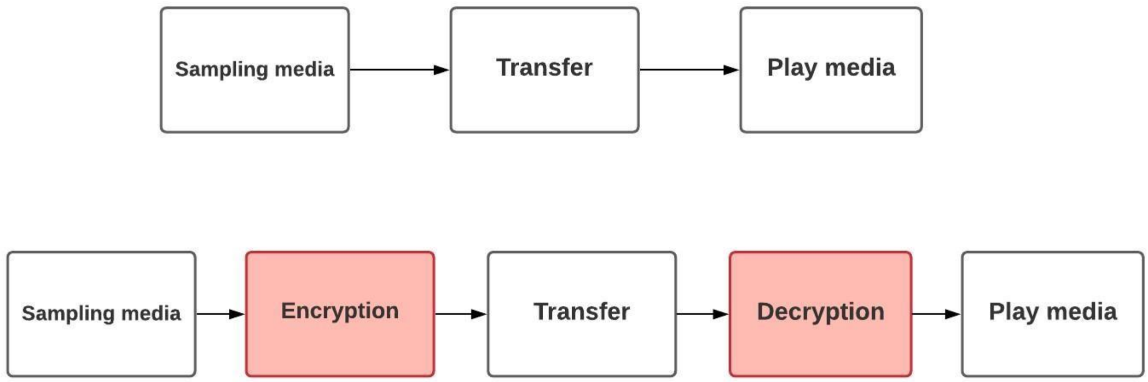


Figure 11: Regular Streaming System and Secure-Streaming System streaming system

In the second section, the RSA process is explained using the block diagram between sender and receiver, after which UDP (User Diagram Protocol) is explained and how it is more suitable than TCP in this because it has a number of attributes that make it attractive for broadcast and real time, he explained Also format the Secured-Streaming package and show the fields on the sender side and the receiver side. Moreover, in the chapter a flowchart was used to show and explain the details of the process between the sender and receiver sides, then the tables were shown and these tables were produced after programming the RSA code, determining the packet size, key size, calculating truncation, the time produced for the encoder time, the decryption time and the transfer time were used. Some equations to produce this time curves were produced and from the curve, the critical point was determined to achieve the optimum balance between safety, real time and quality.

3.2 RSA Detailed Operation

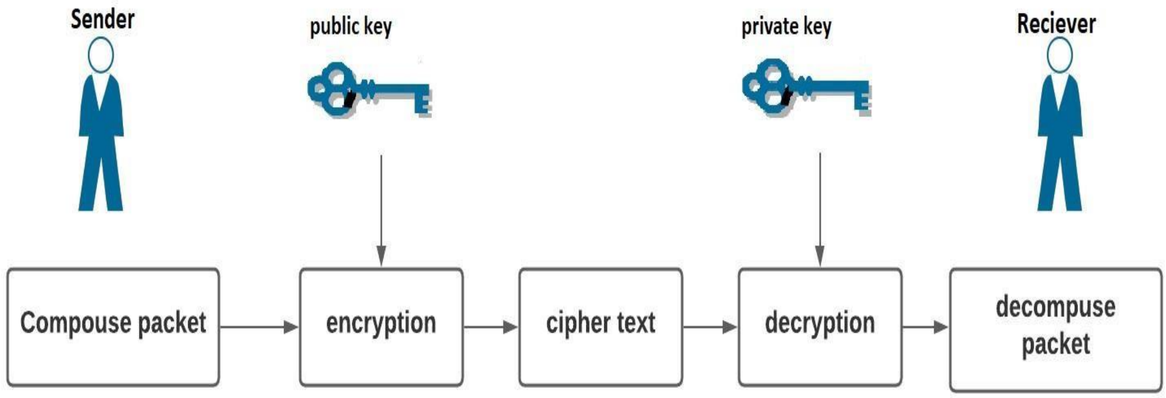


Figure 12: RSA operation

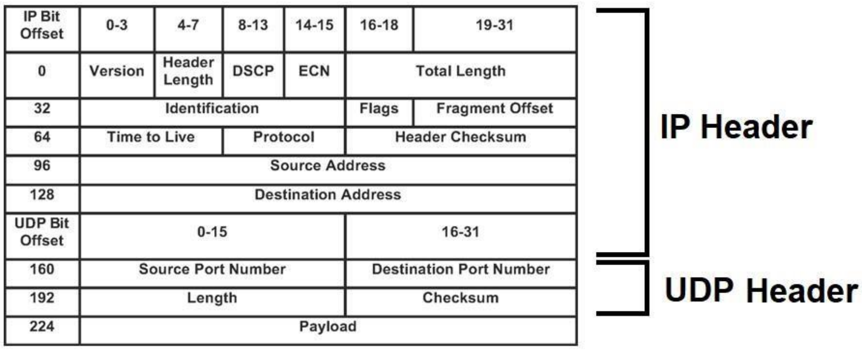
The main idea is when you want to communicate with another person, the sender sends the data using the public key of the receiver, then you encrypt the data and send it to the receiver, the receiver will decrypt and decrypt the data using his private key. In the system, the RSA algorithm is used in the code and depends on the key size, which results from two prime's numbers p and q. Therefore, from the key size we can tell how many bits are used when sending a message from source to destination so if n is big. We can achieve high security from any drop and it will be difficult for him to know the public key to decrypt the message. In addition, the relationship between n (key size) and total time was established to see when the critical point was achieved with the highest level of security and in real time. The system also depends on the packet size. When we want to send audio from source to destination on the Internet, we must define network packet, which is a small amount of audio that is sent over UDP / IP transmission networks. Packet size is about one byte, two bytes for Ethernet. As the packet is the unit of data directed between the source and destination on the network or the Internet, so when the value of the packet size is 300 bytes, this means that the length of the message is 2400 bits. Appropriate so as not to affect the major variables. UDP User Datagram Protocol is responsible for transferring packets between layers. The UDP user datagram protocol is located in the transport layer at the fourth layer above the third layer, which is the network layer at the network layers. It does not guarantee whether the data was transmitted correctly and whether it reached the other party or whether it arrived correctly or not. This protocol is also used to stream media such as Voice, Video, Voice over IP (VoIP) and UDP video conferencing is not connected and does not use handshake to start a session like TCP, it is not necessary that the delivery and time of arrival, the order of arrival of datagrams and the guarantee of the network, so this protocol is important for streaming It facilitates communication and addresses unlike the problem of delay and transmission is done at any time, TCP which needs a three-way handshake to ensure that all parties are active and connected, so it sends and needs a response that the message has reached the other party and from another answer that the data arrived correctly without any defect, and this It needs time, so this protocol is considered slow, which means that it needs to retransmit and error detection, all this takes a long time and needs a lot of time, however, many applications do not require a service, and reliable data flow may use a protocol, So it is a convenient UDP protocol for us The system prioritizes time and continuous streaming without delay In addition, it uses UDP where fast real-time communication is critical, b Biting frames or packets lost in In-process is acceptable and this is very important for my research to achieve real-time but TCP is optimized for accurate delivery rather than just-intime delivery and can cause relatively long delays while waiting for out-of-order messages or retransmitting lost messages. Therefore, it is not particularly suitable for real-time applications such as VoIP. For such applications, as we talked about in Chapter 2 about the most common protocols used for real-time transmission, protocols such as Real-Time Transfer Protocol (RTP) that operate and greatly assist the User Datagram Protocol (UDP) in transmission are usually recommended.

3.3 UDP/IP Packet format

This protocol is located in the transport layer and consists of a UDP datagram from a datagram header and a data section, and this data came to it from the RTP protocol. It consists of a UDP datagram header of four fields, each of which is 2 bytes (16 bits) and the data felid is the payload data, see Figure 3.3 showing the correct image of the components of this protocol

- Source port number: 16-bit information is used to identify the source port of a packet
- Destination port: This 16-bit information is used to specify an application level service on the destination
- Length: This field specifies the length in bytes of the UDP header and UDP data. The length of the header data, which is 8 bytes, is considered the length of the header. The field size sets a theoretical limit of 65,535 bytes for a UDP datagram.
- Checksum: This field checks for errors, whether packets are delayed or lost during transmission, and errors in the address of both the sender and receiver.

Then the data section or payload field representing all the data is moved from the transport layer to the Internet (IP) layer.



IP Bit Offset	0-3	4-7	8-13	14-15	16-18	19-31
0	Version	Header Length	DSCP	ECN	Total Length	
32	Identification				Flags	Fragment Offset
64	Time to Live		Protocol		Header Checksum	
96	Source Address					
128	Destination Address					
UDP Bit Offset	0-15				16-31	
160	Source Port Number				Destination Port Number	
192	Length				Checksum	
224	Payload					

Figure 13: UDP/IP Packet format (Irwin and Slay,2011).

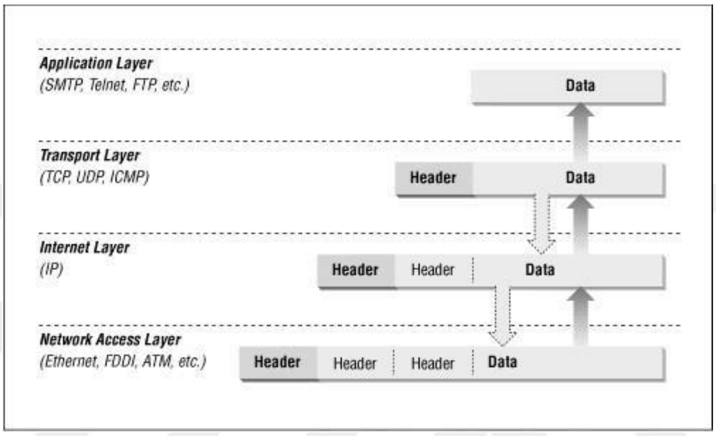


Figure 14: Data encapsulation (Wikipedia, 2013).

3.4 Secured-Streaming System Flowchart

In this section, the working method of the system used to protect the audio streaming is explained which is described in figure 3.5. There will be two parties in the protection system, a sender and a receiver. At first, one of the communication parties checks if it receives packets from the other party or not. If the answer is no, it will determine the timestamp Then it does samples for the data, and then it will determine the identification number of the sender and receiver, and then it encrypts the data and sends the encrypted data to the other party. However, if one of the parties is now receiving data and the result of the check is yes, he will decrypt the data and then authentication for the receiver identification number if it is the same as the identification number in the packet that was sent. If the answer is yes, he will authentication again if the sender's identification number is it is the same identification number as the package sent by the sender. If the answer is yes, it will determine the type of package that it will send to the other party shown in the table number, is the packet it will send a Synchronization Request from receiver to sender with Initial Public Key1, Initial Sampling Rate1 or packet type Synchronization Acknowledge from sender to receiver will be sent or the type of packet is a Secure Streaming packet handler, which we focus on in our system, which will be sent from the sender to the receiver and includes the encrypted text, which the sender will send to the

receiver, and at this type it will be completed through the explanation in Figure 3.8 SecureStreaming Packet Handler, which explains its explanation in the next section whose number is
3.7, If an encrypted text is sent to the other party ,in the fourth type, if the packet type is Update Parameters Packet from the receiver to the sender, and it will be a modification of the transmitted data, and the public key is used to send to the other party as for the last type of package, Stop Server Command, Stop Client Command, if either there is no data, from the sender to the receiver or vice versa, here we will finish the process of sending and receiving.
In figure 3.5 Secured-Streaming packet format shows the items that the sender chooses when he wants to send them to the other party. Specifies the sender identification number, the recipient identification number, the timestamp by which it is known when the first sample was sent, immediately sampling the first byte, which will be sent to the recipient to determine its time based on a sender and determining the timestamp is important to know if there is a delay in transmission based on the feed Returning from the receiver and who determines the type of packet sent and the last part in which all the data is carried to be sent to the other party and placed in the data field.

16-bit	**16-bit**	**32-bit**	**8-bit**	**16-bit**	**8-bit**	**12 bytes – 64K bytes**
Sender ID	Receiver ID	Time stamp (msec)	Packet type	Length	Res.	Data

Figure 15: Secured-Streaming packet format

Table 1: Packet type

	Packet type	**direction**	**data**
0	Synchronization Request	from Receiver to Sender	Initial Public Key1 , Sampling Initial Rate1
1	Synchronization Acknowledge	from Sender to Receiver	Initial Public Key2, Initial Sampling Rate2
2	Stop Server Command	from Receiver to Sender	null

3	Stop Client Command	from Sender to Receiver	null
4	Secure Streaming packet handler	from Sender to Receiver	Cipher text
5	Update Parameters Packet	from Receiver to Sender	Public Key , Sampling rate

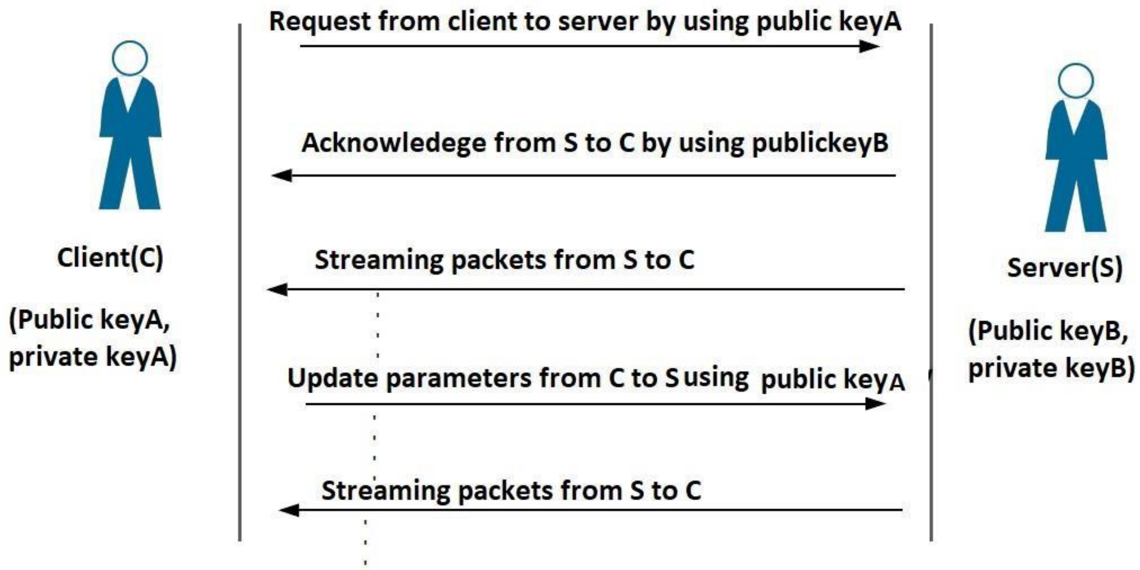


Figure 16: Packets types in RSA

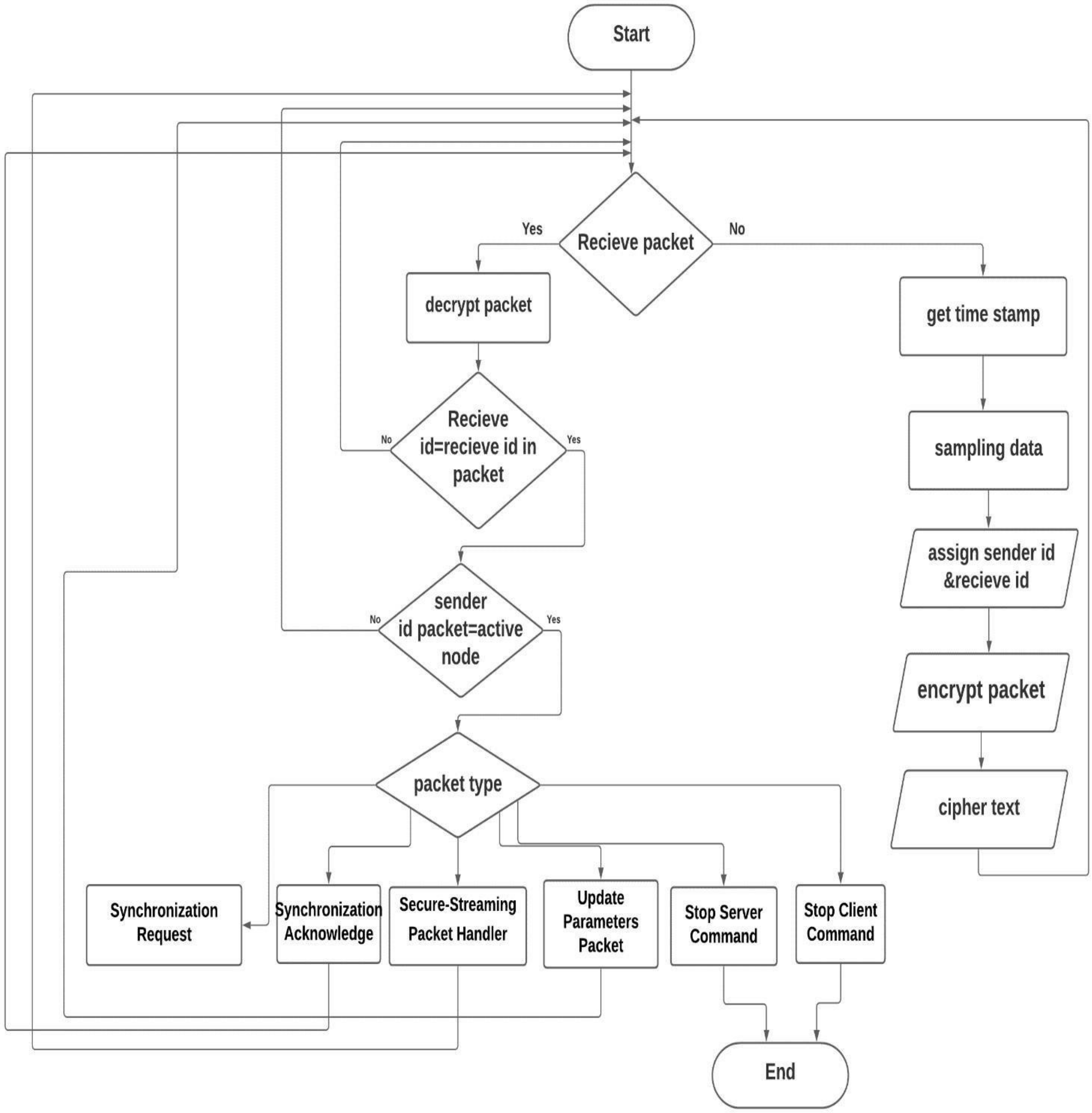


Figure 17: System flowchart

3.5 Secure-Streaming Packet Handler

In this section, it was explained if the sent packet is an encrypted text from the sender to the receiver and the data is the encrypted text and then its arrival to the other party will play the sound and then calculate the transaction time by subtracting the receiving time from the time of the first timestamp and then Check whether the transaction time that produced from the total encryption time and transfer time and decryption time Which is calculated after executing the RSA code If the transaction time is greater than time threshold maximum and the time calculated during transmission, the first error is calculated and then compared and calculated how many times a large time was consumed in the transmission and a large loss of packets, it will be sent to the other party to modify the quality and protect against some variables It is the size of the key and the size of the packet used by decreasing and then this update to the sender. but when the sender calculates the total time and it is less than the transaction time this means that there is no defect in the transmission, and then it calculates again the total time if it is greater than time threshold minimum ,this means that there is little space for time, so it is possible to increase and improve some variables by increasing the key size and the size of the packet to increase confidentiality and quality, and then send this request to the other party, or if it is calculated and the total time there is no room for improvement and to increase security and quality, it does not send and order for improvement and continues to receive packets.

After executing the RSA code, the total time to send was calculated by calculating.

- Encryption time (T2-T1):

The time after the sender obtains the recipient's public key; it can send a message (M) to the recipient. The sender at this point converts plaintext message into an integer, the cipher text is computed using the receiver's public key and sent (c) to the receiver.

The time produced from this operation called encryption time.

- Transfer time:

The time it takes to transmit or move data from one place to another or the time taken for a data transfer between the networks and the host system. This time is dependent on the size of the data transfer and the rate at which it can be transmitted to or from the host and usually it is very small. I calculated it by using this formula:

bytes_per_Packet =byte_per_sample* sample_per_Packet

Then,

Trans _Time =bytes_per_Packet * 10.0 / Baud Rate.

Where Baud Rate is considered to know the amount of changes or the number of changes in the signal per one second through the transmission channel.

- Decryption time (T4-T3):

This time obtains when the receiver can recover the message (m) from the cipher text (c) by using private key. The receiver can then recover the original message (M).

Then calculated the transaction time (encryption _time, transfer _time and decryption time) from all this operations.

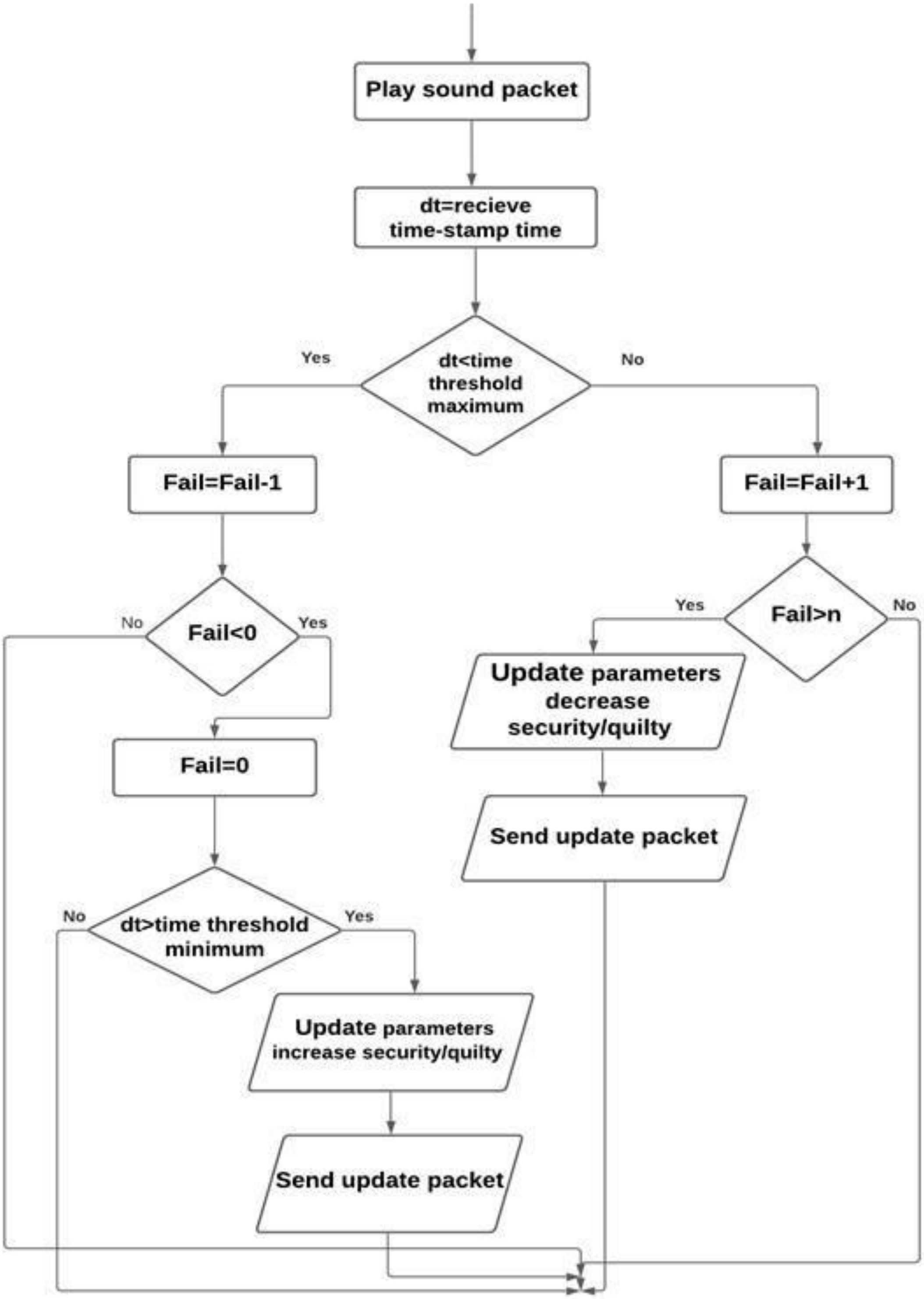


Figure 18: Secure-Streaming Packet Handler

4.RESULTS

4.1: Simulation Results

In this section, all the results that appeared after experiments are shown and shows the effectiveness of the thesis and its achievement of the goals we wanted, which is to achieve a balance between protection and quality and maintain real time during media streaming, especially audio streaming, without affects affect them negatively any of the three elements. Among the most important results that have been reached it is to know the transaction time of transmission by calculating the encryption time in addition to the time of transmission over the network and the time of decoding, after using and programming the RSA code, and using some equations and calculations to calculate this also used different sizes for the public key, from multiplying the two primes numbers and changing their sizes to reach the best results and from them and knowing the total time at each key size such as for key size 221,253,247,341,667 bits and using different sizes of the package size such as 500,700,1000,1500,2000,3000 bytes per second , see table 4.2 , then draw curves to clarify the relationship between the total time and the size of the key and a curve It also shows the relationship between packet size and transaction time.

.

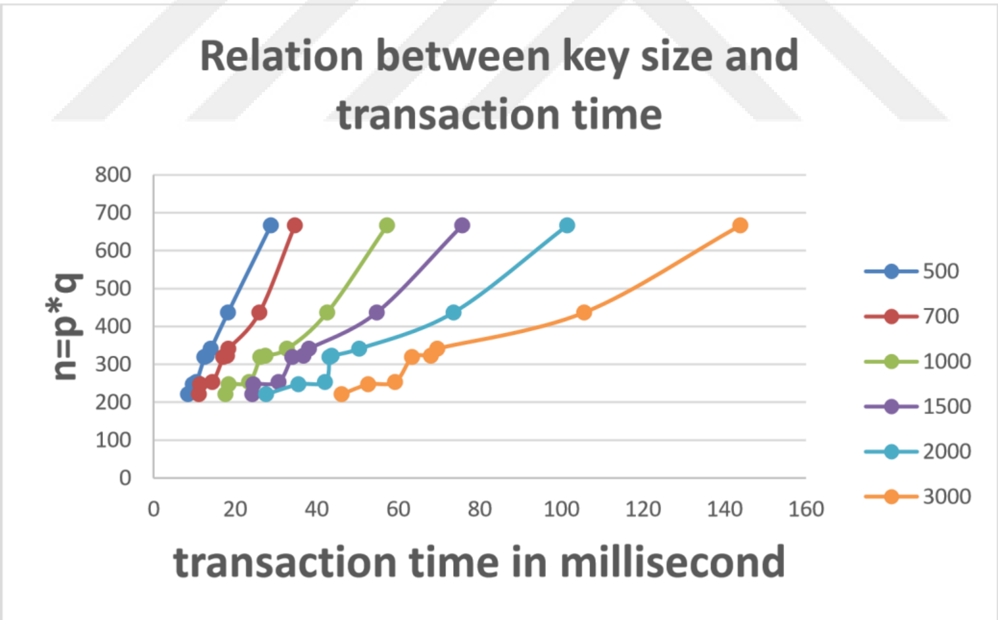


Figure 19: Relation between keysize and transaction time

The table describes the relation between the sample per packets and the transaction time when the key size 221, 247,253,319,323,341,437,667bits.

Table 2: Relation between the sample per packets and the transaction time

n	time	sample per packet	n	time	sample per packet	n	transaction time	sample rate
221	8.46208333	500	247	9.6491666	500	253	10.4910833	500
221	11.184233	700	247	11.424166	700	253	14.4501166	700
221	17.63033	1000	247	18.449166	1000	253	23.477166	1000
221	24.2655	1500	247	24.46925	1500	253	30.68025	1500
221	27.63966	2000	247	35.52233	2000	253	42.09233	2000
221	46.128	3000	247	52.6295	3000	253	59.2335	3000
n	time	sample per packet	n	time	sample per packet	n	transaction time	sample rate
319	12.5050833	500	323	13.1240833	500	341	14.05708333	500
319	17.129233	700	323	18.0761666	700	341	18.402233	700
319	26.202166	1000	323	27.440333	1000	341	32.76133	1000
319	34.09725	1500	323	36.8465	1500	341	38.1745	1500
319	43.13333	2000	323	43.69066	2000	341	50.47766	2000
319	63.4595	3000	323	68.055	3000	341	69.595	3000
n	transaction time	sample rate	n	transaction time	sample rate			
437	18.27416	500	667	28.782166	500			
437	25.989233	700	667	34.657233	700			
437	42.59933	1000	667	57.31233	1000			
437	54.7675	1500	667	75.7145	1500			
437	73.61766	2000	667	101.52466	2000			
437	105.684	3000	667	143.967	3000			

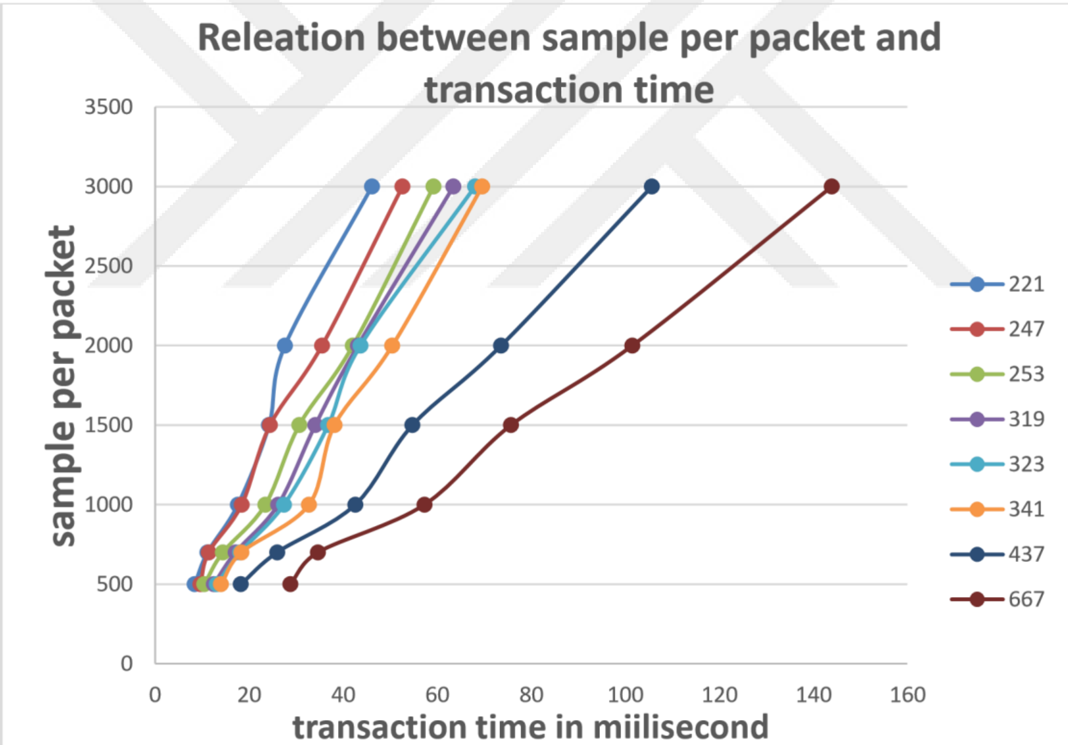


Figure 20: Relation between transaction time and sample per packet

4.2 Optimal Solution

In this section, the relationship between security and quality is explained and clarified and how it is possible to maintain the streaming of real time and achieve a balance between security and quality in the order that any feedback comes from the recipient by modifying some variables such as the size of the key or the size of the packet by decreasing or increasing it blinds either that we decrease protection and quality or increase protection And quality. Therefore, we must preserve the two, by choosing a point and an ideal solution that achieves a balance between protection, quality and real time. This can be achieved by creating a curve that provides the optimal solution, achieves a balance between conserve and quality, and conserve real time streaming. In figure 4.3 describes all curves with different values for sample per second between key size and transaction time after did for it polyfit from the third degree and is chosen third degree because mat lab program give me warring to choose less number because the point on the curve little.

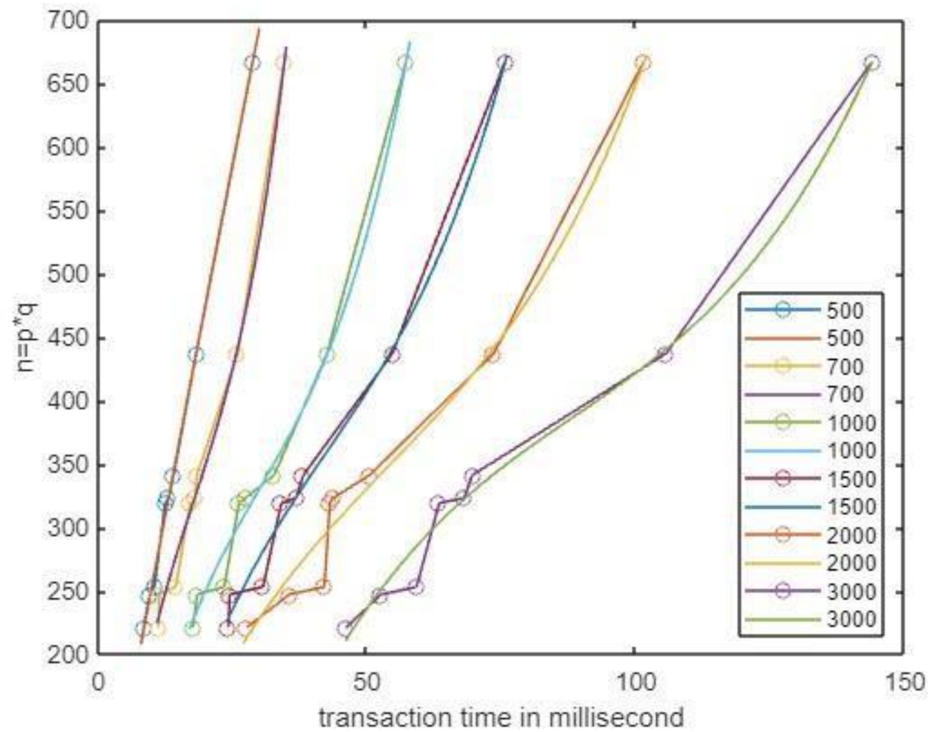


Figure 21: Relation key size and time after converted sample results into polynomial by polyfit

In the figure 4.4 appears system point's reels and the working area, the system performance trapped between maximum security when key size 667 and the maximum time 150ms during used audio 3000 sample per second and the minimum security when key size 221 and minimum time and used 500 sample per second.

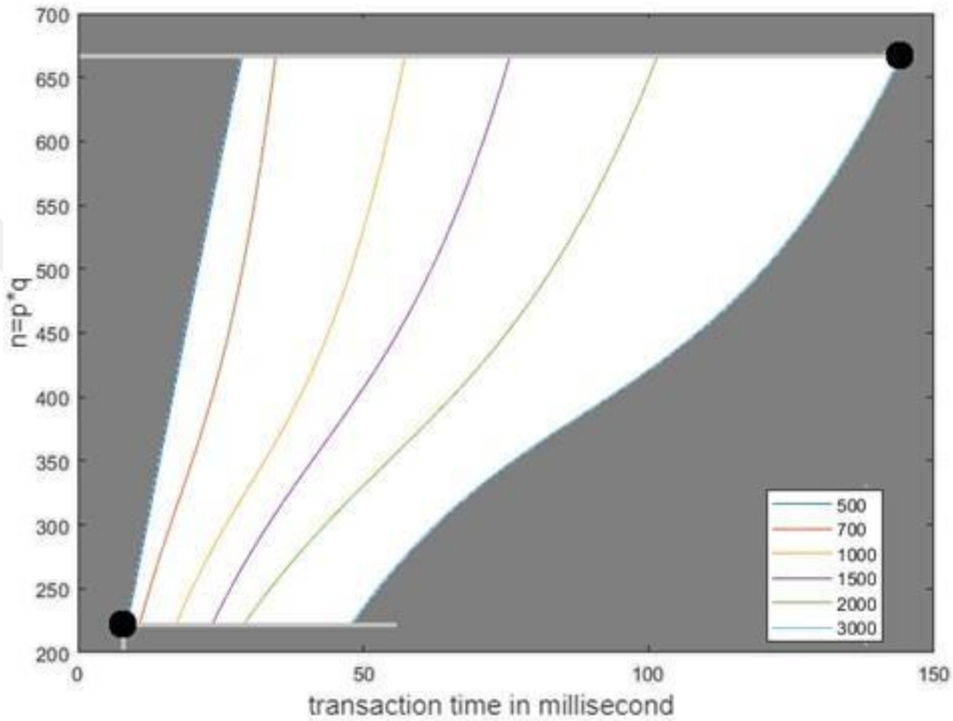


Figure 22: System points reels

In figure 4.5 is begin chosen initial point default and is not necessary to know the values of this point because after one transaction or one step will choose the optimal solution and is not necessary to determine security value and quality value.

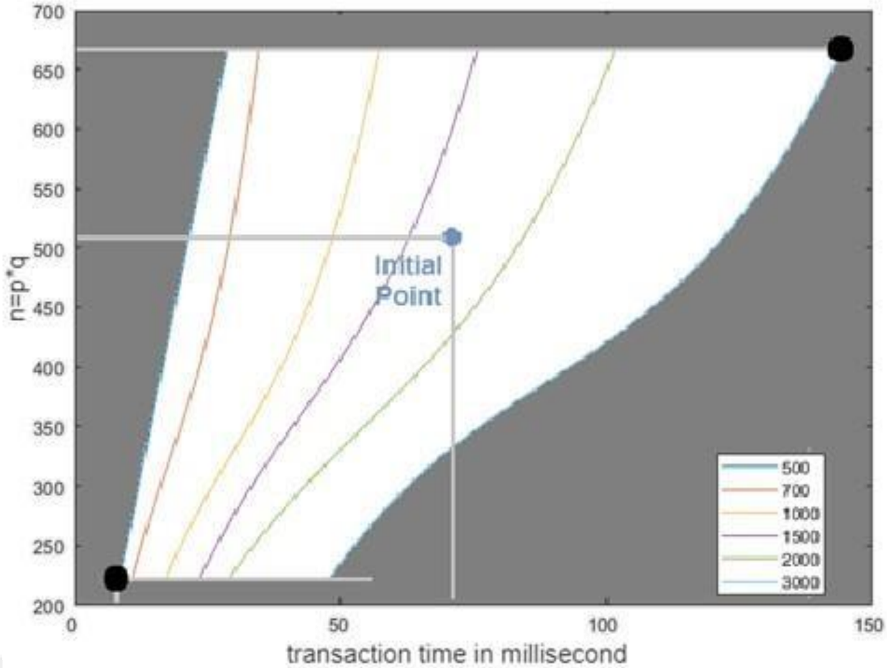


Figure 23: initial point in working area

Now, in figure 4.6, 4.7,4.8 and 4.9 if receiver send to sender update some parameter such as quality and security so if moved in the direction of the red arrow the security will increase and the quality goes down and verse vise if moved in the direction of green arrow the quality will increase and the security goes down. Therefore, the new point should achieve balance between the security and quality.

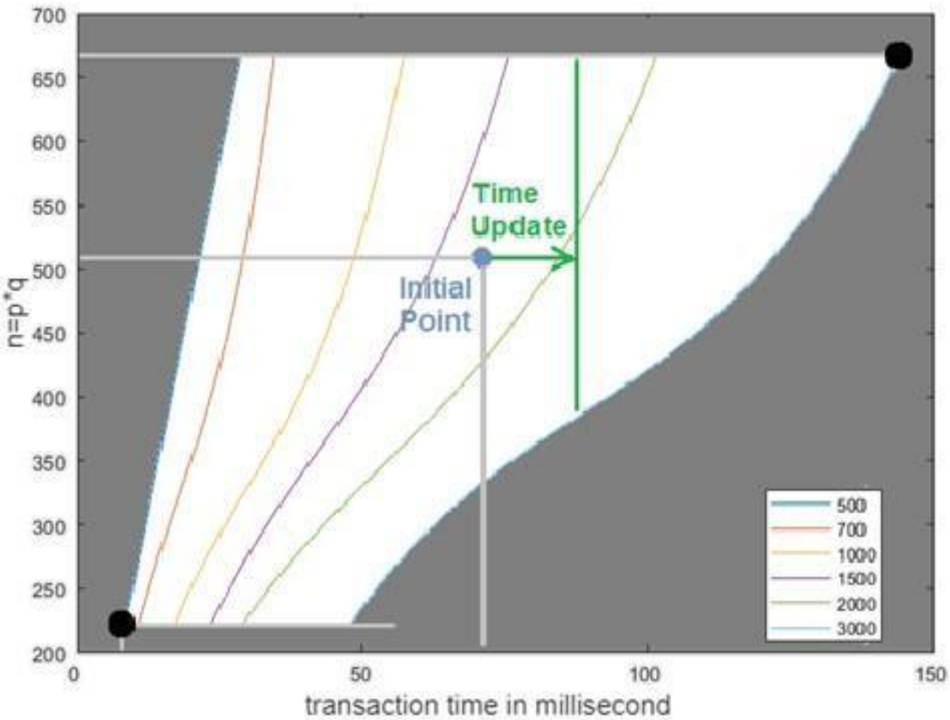


Figure 24: Time update

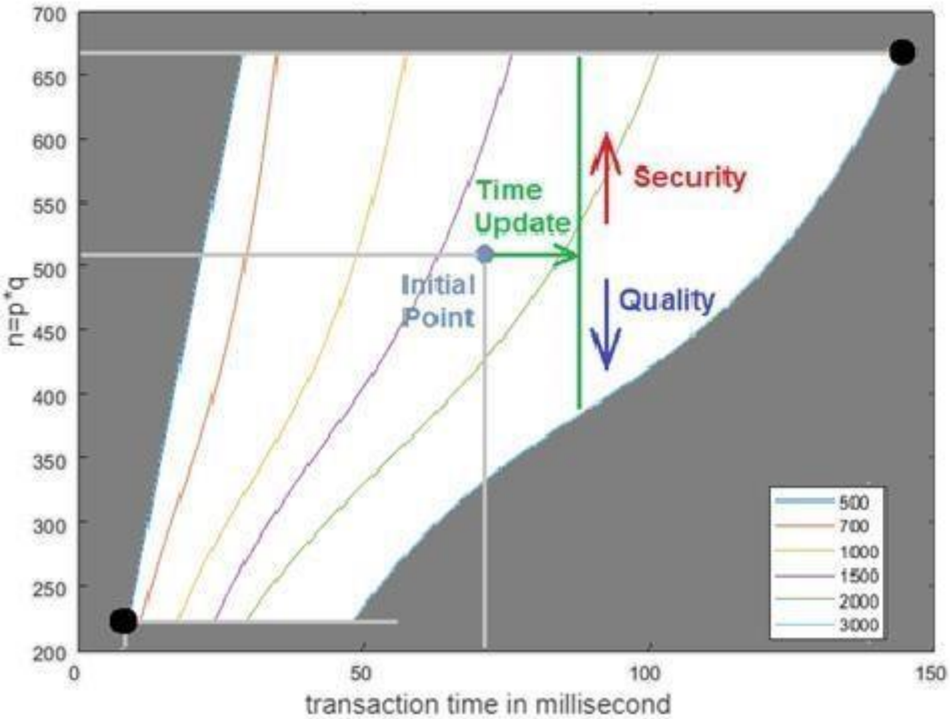


Figure 25: Security increasing or Quality increasing

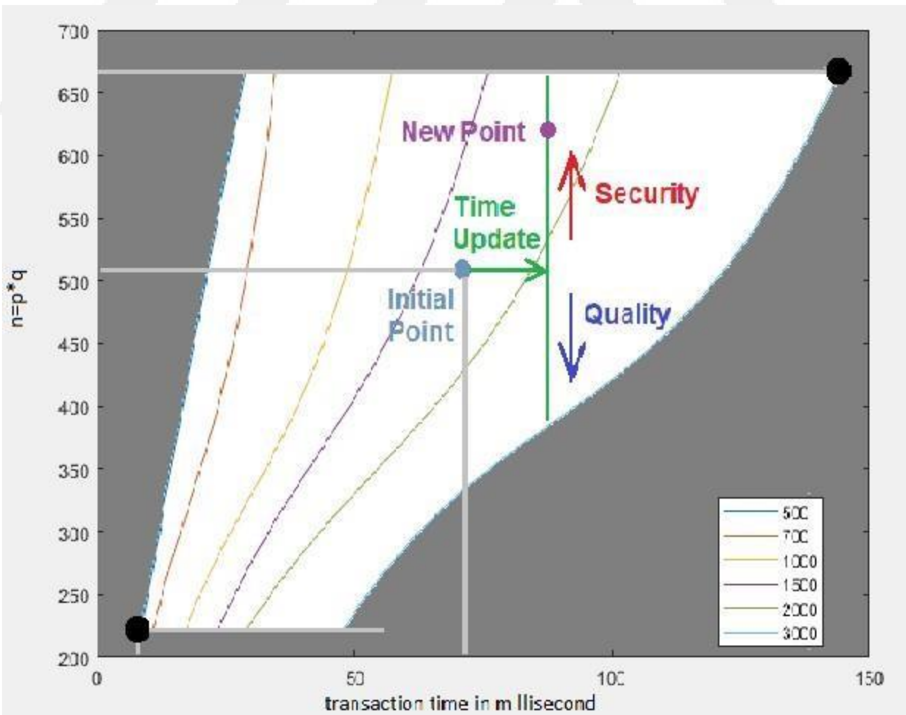


Figure 26: New point to achieve optimal solution

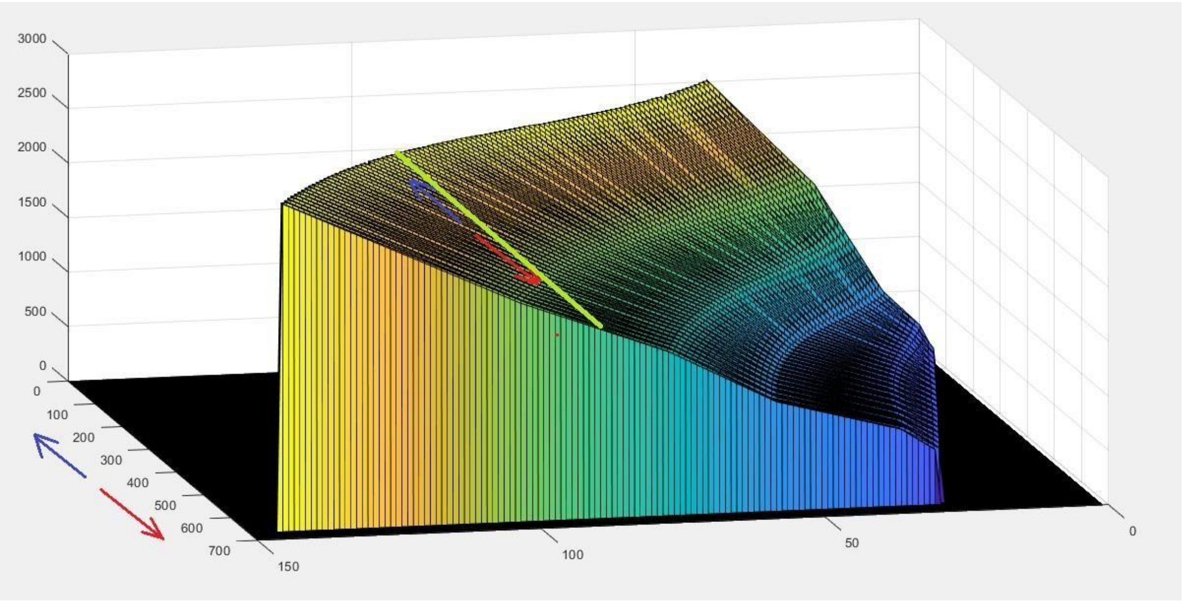


Figure 27: Quality Map

Figure 4.10 shows the method of creating a new point that achieved a balance between the security and the quality and this is obtained from the generated cost map: Cost = Natural Safety / Natural Quality. In addition, the balance between security and quality is also chosen when determining a cost close to one of them, since the scaling differs in terms of quality and security.

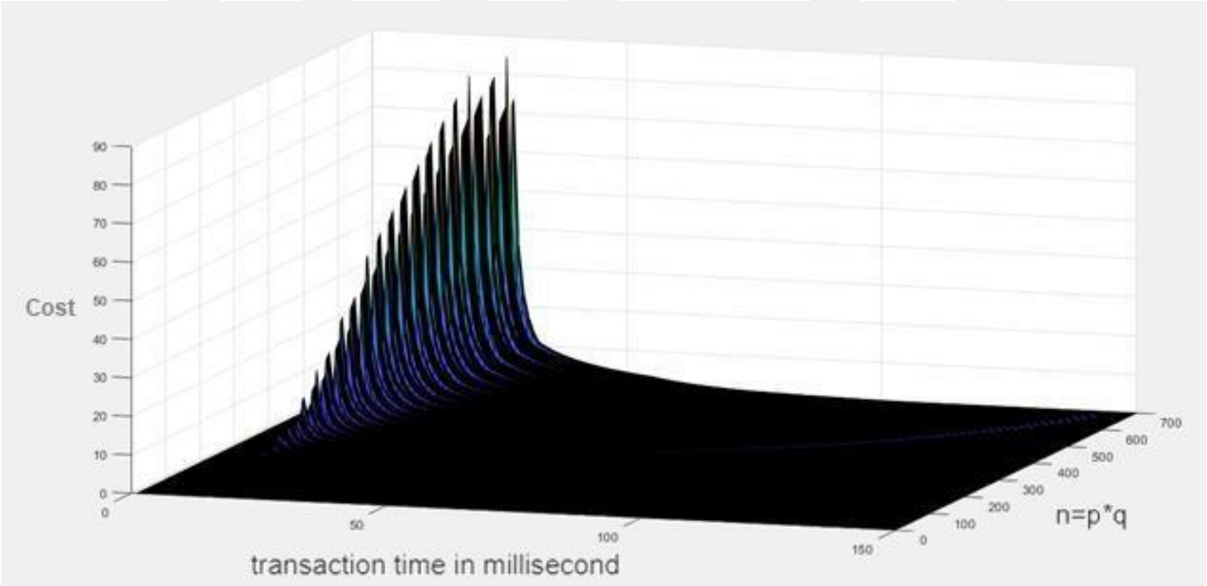


Figure 28: Cost Map

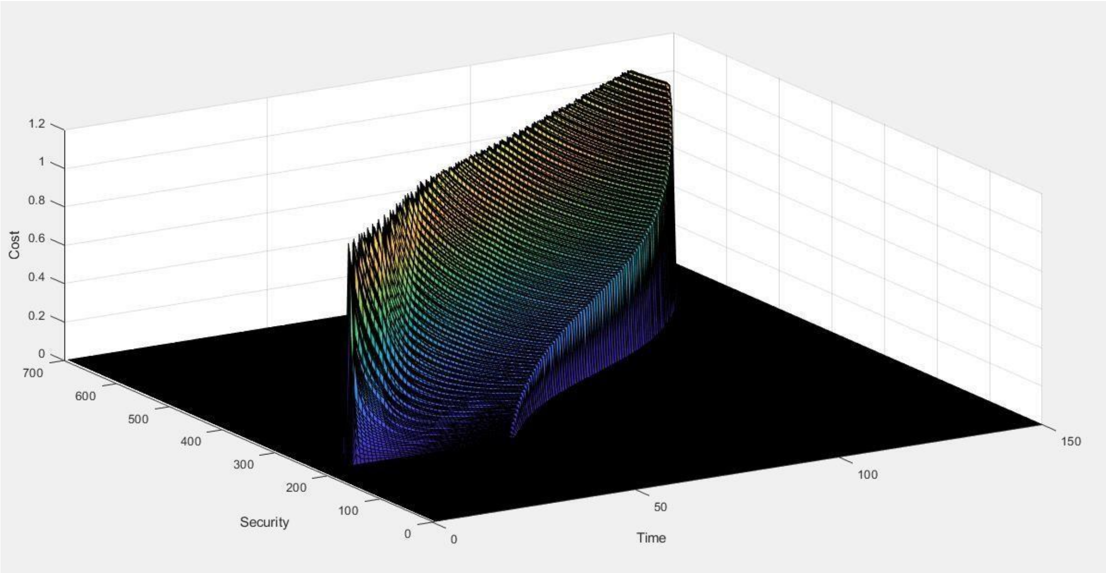


Figure 29: Cost Map in 3D

In figure 4.11 appears optimal solution curve in 3D shape between transaction time(X- axis) and security (Y-axis) and quality (Z-axis) to achieve balance between them so every point in optimal line achieve optimal performance.

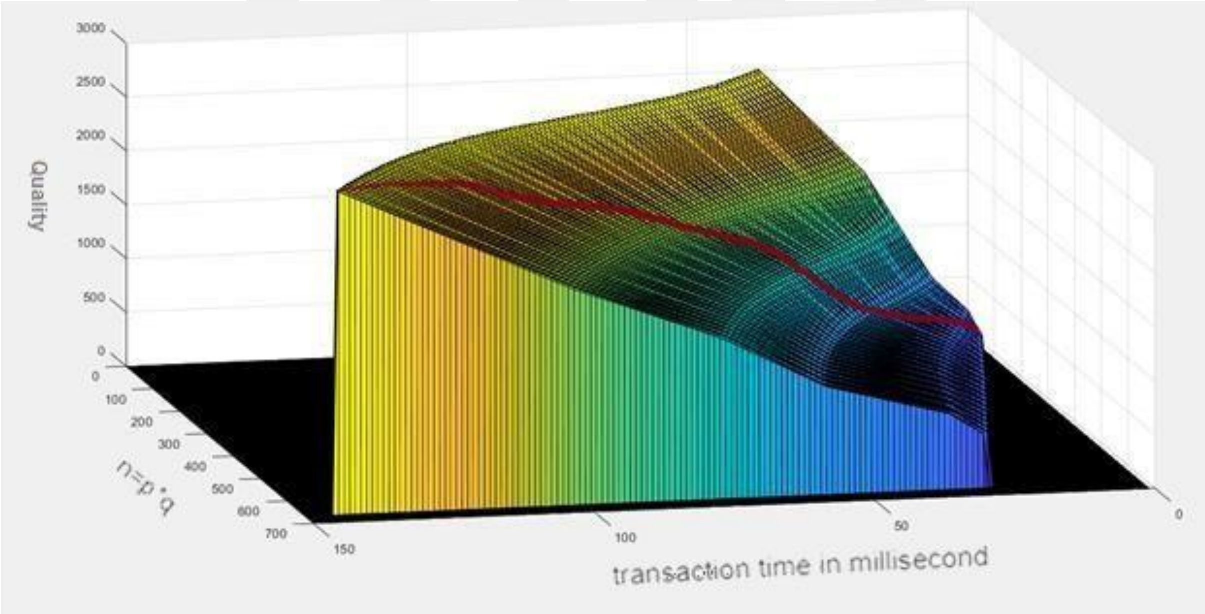


Figure 30: Optimal Solution

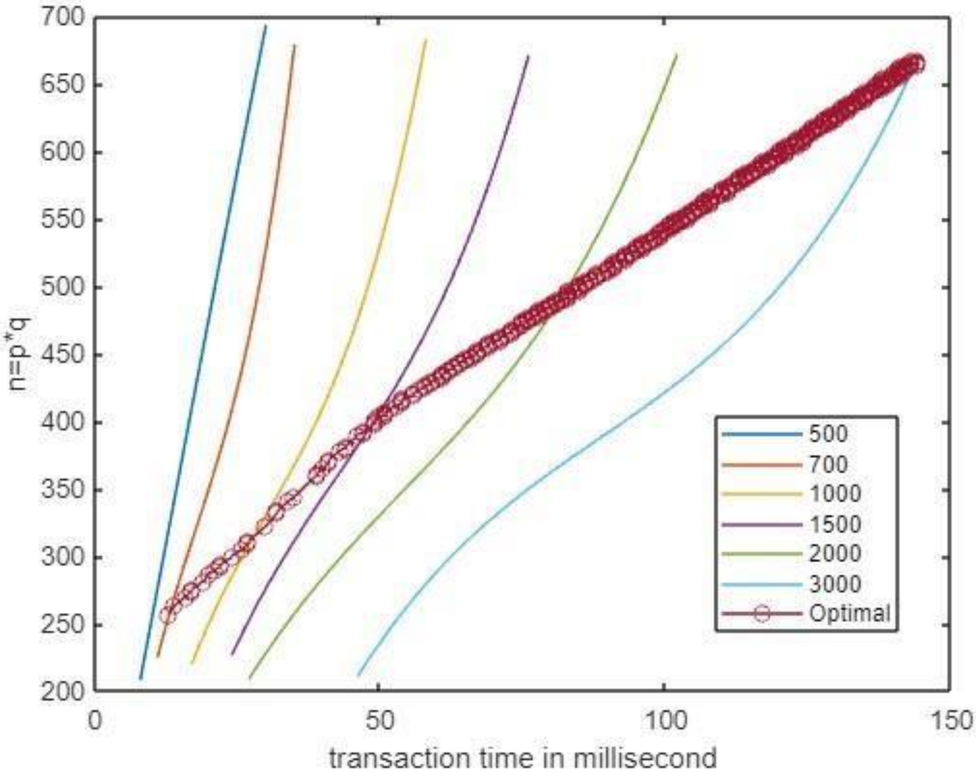


Figure 31: Optimal Solution curve

In figure 4.13, 4.14, for example to determine the optimal solution so when choose time_intial=100 and dt =140 and max_ time=150 then time update = time_inital + (max _time – dt) =100+ (150-140) =110msec.

So the optimal and best security for y(i) =568bits and the transaction time =110ms and the quality = 2450bytes per second also in 568 for security can increase this value because this number even and not prime number so choose less than this number such as 561bits to conserve the system real time streaming .

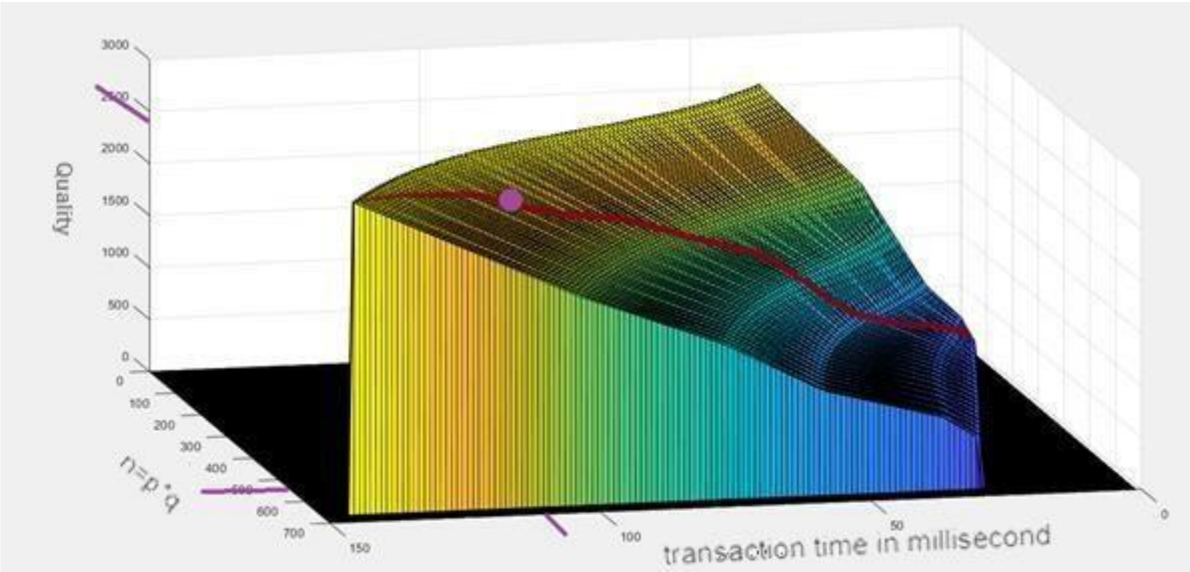


Figure 32: 3-Dimiontion Optimal Solution Example

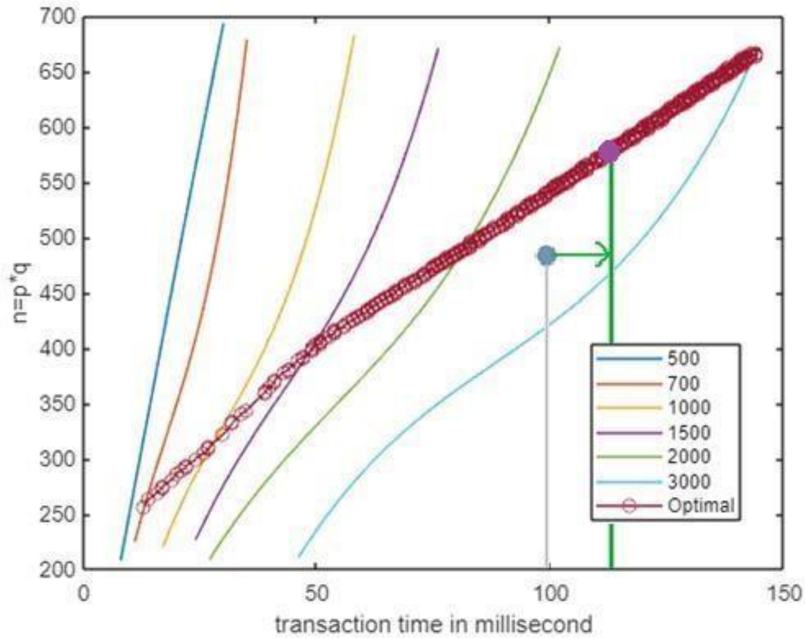


Figure 33: Optimal Solution example

5. CONCLUSION AND FUTURE WORK

The thesis studied and improved security and quality according to the voice flow in the communication application and achieved a balance between all parameters, security, quality and maintaining the response in real time without delay, in addition to that when implementing the idea, the RSA algorithm was used and transactions were controlled such as key size, packet size, increase or Reducing transactions based on recipient feedback If the transaction time is correct and there is no delay, it will either send to the other side to increase security and quality, or if it is delayed and lose quality, the voice will be sent to the other side Reduce some transactions to improve quality. so security can be improved by using dynamic parameters such as key size and sample size, moreover, this preserves the real time and avoids packet loss during data transmission because it depends on the UDP protocol through which the media flow is fast so that this protocol preserves and guarantees the real time of the voice and this The point is important in our research, also the loss of some frames or packets in the process is acceptable and this is very important for my research to achieve real time in audio broadcasting, also, after a lot of experimenting and trying in RSA code, when wanting to send a message and sample for each packet was 1000, 1500 packets per second, 1 byte, 2 byte, 3 byte was selected to send a message but I noticed when selecting 2,3 byte the send time was very large specifically the decryption time and this depends on the RSA equations if you decide to send a 2 byte message the text will increase Encryption and the private key increases the decryption time and this takes a lot of time and decryption time in particular, which causes a time delay and consumes a lot of time when decrypting the message, so the best solution is to send a message of 1 byte per second to keep broadcast time. Then some mathematical equations were used and the optimum solution was chosen to strike a balance between real time, quality and security. In the future, it is recommended that other researchers test and implement audio streaming security using other algorithms such as AES and DES, choose different sample lengths for each packet and key size to achieve the highest real-time security in the communication process, and implement this idea in other streaming media such as video. The code will also be executed in the network simulation so I used TCL because this language is suitable for ns2. System point reels will use maximum safety at 3000 samples per second and time 150 milliseconds but will operate at this time and safety points in future work to achieve an optimal time limit value to achieve real-time streaming with the highest level of security. We also recommend using a different quality sample rate and in future work it will use more than 3000 bytes, in the last in the future work will implement the streaming system by adding SA and SSL.

References

Abdullah A. M. (2017) Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data, Department of Applied Mathematics & Computer Science Eastern Mediterranean University – Cyprus.

Agrawal M. and Mishra P.,(2012) “A comparative survey on symmetric key encryption techniques,” Comput, I. J Sci. Eng., vol. 4, no. 5, pp. 877–882.

Aumasson J. P. (2018) SERIOUS CRYPTOGRAPHY A Practical Introduction to Modern Encryption, San Francisco: No Starch Press, Inc.

Apostolopoulos J. G., Tan W. t., Wee S. J. (2005) Video Streaming: Concepts, Algorithms, and Systems Mobile and Media Systems Laboratory.

Bodur H., Kara R. (2015), Secure SMS Encryption Using RSA Encryption Algorithm on Android Message Application.

Bokhari M. U., Shallal Q. M.(2016), A Review on Symmetric Key Encryption Techniques in Cryptography, Dept. of Computer Science.

Brown G., (2021) Audio Sample Rate and Bit Depth.

Caesar cipher information at brilliant.org (2017) . Brilliant.org. Archived from the original .

Coron J. S., Joye M., Naccach D., Joye M., (2004) Universal Padding Schemes for RSA.

Durresi A., Jain R.,(2005) RTP, RTCP, and RTSP - Internet Protocols for Real-Time Multimedia Communication, the industrial information technology.

Geetha Dr. R. K. (2020), A Review Paper on Concepts of Cryptography and Cryptographic Hash Function.

Gandhi R. A., Gosai A. M. (2015), Ph.D. A Study on Current Scenario of Audio Encryption, International Journal of Computer Applications.

Hassan N., Al-Mukhtar F. S. and Ali E. H.,(2020) Encrypt Audio File using Speech Audio File As a key, Baghdad, Iraq, AL-NAHRIAN UNIVERSITY, DEPARTMENT OF COMPUTER SCIENCES.

Irwin D. and Slay J.,(2011) EXTRACTING EVIDENCE RELATED TO VoIP CALLS.

Jain, S., Chhibber, N., Kandi, S. (2018), Cryptanalysis of Mono-Alphabetic Substitution Ciphers using Genetic Algorithms and Simulated Annealing.

JADHAV P., (2019) Advanced Encryption Standard.

Khan, S. A. (2015), Design and Analysis of Playfair Ciphers with Different Matrix Sizes, Computer Engineering Department, College of Information Technology.

Kaur N., Sodhi S. (2016), Data Encryption Standard Algorithm (DES) for Secure Data Transmission, International Conference on Advances in Emerging Technology.

Kanzow, K. J., Kretschmer P. M. (2015) Audio Streaming on the Internet Experiences with RealTime Streaming of Audio Streams.

Lavry D. (2017) Sampling Theory for Digital Audio.

Lone A. H., Uddin Prof. M. (2016), Common Attacks on RSA and its Variants with Possible Countermeasures.

NWE1 T. Z. , WAIPHYO2 S., (2014) Performance Analysis of RSA and Security.

Khalil M.I.,(2016) Real-Time Encryption/Decryption of Audio Signal, Computer Network and Information Security .

Nikki G., (2013) et al. "Surveillance Privacy Protection." Intelligent Multimedia Surveillance. Springer Berlin Heidelberg.

Nisha S., Farik M., (2017) RSA Public Key Cryptography Algorithm.

Piper, F. and Murphy, S. (2009) Cryptography: A Very Short Introduction, London: Oxford University Press.

Qadir A. M., Varol N. (2019) A Review Paper on Cryptography, Software Engineering Department.

Oguntunde, B.O, Arekete, S.A, Odim, M.O and Olakanmi, O.I. (2017) A Comparative Study of Some Traditional and Modern Cryptographic Techniques.

Poduval A. , Rai N., Khan,P., Sane A.(2021) A SURVEY ON DIFFERENT ENCRYPTION TECHNIQUES FOR IMAGE, VIDEO, AUDIO AND DOCS, International Journal of Engineering Applied Sciences and Technology.

Ratnadewi , Adhie R. P., Hutama Y. , Ahmar A. S., Setiawan M. I.,(2018) Implementation Cryptography Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES) Method in Communication System Based Near Field Communication (NFC), Electrical Engineering Department, Universitas Kristen Maranatha , Bandung, Indonesia.

Rivera, L. (2020), Symmetric vs. Asymmetric Encryption.

Richards K. (2021), cryptography, Kathleen Richards is a former features editor of Information Security magazine.

Saikumar I. (2017), DES- Data Encryption Standard, Post Graduate Student, Electronic and Communication Engineering.

Schneier B. (2015) Applied Cryptography, Electr. Engingeering.

Singh B., Athithan G. and Pillai R.,(2021) On extensions of the one-time-pad, Centre for Artificial Intelligence and Robotic DRDO, Bangalore, INDIA.

Sobti R., Geetha G.(2012), Cryptographic Hash Functions, School of Computer Science.

STUART J. R. (2007). Coding High Quality Digital Audio .

Srividya, L., Dr. Sudha. P.N.,(2016) LITERATURE SURVEY ON RECENT AUDIO ENCRYPTION TECHNIQUES, International Journal of Electronics and Communication Engineering and Technology.

Wang, Cong, (2012) et al. "Toward secure and dependable storage services in cloud computing." Services Computing, IEEE Transactions.

Wikipedia (2013) [https://en.wikipedia.org/wiki/Encapsulation_(networking)] Access date: November 2013.

Zager, R. P. (2015) et al. "Rapid identification of message authentication." U.S. Patent Application.

APPENDIX A.RSA Code

```
namespace eval Crypto {

	# functions that implement useful mathmatical algorithms used in cryptography
	# Auth: Reese Krome
	namespace eval RSA {
		# functions that implement RSA math and primality tests
		# simple encryption and decryption with non-standard key formats

		# num is a randomly selected number that will be checked for primality
		variable num 0
		variable pubkey {}
		variable prvkey {}

		proc getpubkey {} {
			variable pubkey
			return $pubkey
		}

		proc getprvkey {} {
			variable prvkey
			return $prvkey
		}

		proc setprime {prime} {
			# set the global prime variable
			variable num
			set num $prime
		}

		proc miller-rabin {max witnesscounter} {
			# An implementation of the Miller-Rabin primality test
			#	max: the maximum number of base 2 factors to find
			#	witnesscounter: the maximum number of witnesses to find

			variable num
```

```
variable basecounter 0
variable d 0
variable r 0

set n [expr {$num - 1}]

# find base 2 factors of (unknown prime - 1)
for {set idx 1} {$idx < $max} {incr idx} {
	if {[expr {$n % (2 ** $idx) == 0}]} {
		set d [expr { $n / (2**$idx) }]
		set r $idx
		# puts "$n has a factor 2^$r * $d"
		# a base 2 factor has been found
		incr basecounter
	}
}
# puts "\$n = $n\n\$d = $d\n\$r = $r"

if {[expr $basecounter == 0]} {
	# if no base 2 factors, check if its a perfect square
	if {[isperfectsquare $n]} {
		return {composite}
	}
}

# puts "Entering WitnessLoop"
for {set k 0} {$k < $witnesscounter} {incr k} {
	variable x
	set a [Crypto::randrange 2 [expr $num - 2]]
	set x [expr { $a ** $d % $num } ]
	# puts "$a ** $d % $num = $x"
	if {[expr {$x == 1 || $x == $n}]} {
		continue
	}

	# for the number of found base 2 factors
```

```
        for {set index 0} {$index < $basecounter} {incr index} {
            variable x
            # puts "($a ^ $d mod $num) = $x"
            set x [expr { $x ** 2 % $num }]
            if {[expr { $x == $n}]} {
                # if the continued squaring of x, mod num
                # returns num-1 or 1, then num is probably prime or 'a' is a strong liar
                break
            }
            # if the continued squaring of x, mod num
            # returns [2,num-2], then 'a' is a witness for the compositeness of num
            # and previous 'a's were in fact strong liars
            puts "$a is a witness to the compositeness of $num. "
            return {composite}
        }
    }
    return {num is probably prime}
}

proc trialdivision {max} {
    # check if the number has small prime factors
    # returns 1 if the number is divisible by small numbers
    variable num

    for {set idx 2} {$idx < $max} {incr idx} {
        if {[expr {$num % $idx == 0}]} {
            return 1;
        }
    }
    return 0;
}

proc isprime {value} {
    # Based on the Baillie-PSW primality test
    # Lucas pseudoprimes have not been implemented but this does an OK job
```

```
        variable num
        set num $value

        # Try trial division up to 99 for efficiency. If not small factors are found, do a primality test
        if {[trialdivision 99]} {
                return {composite}
        } else {
                return [miller-rabin 4 10]
        }
}

proc test-primes {} {
        # tests each number in the list and tests primality
        # requires a file with a space separated list of primes

  set primes [gets [open {./primes.txt}]]
        foreach prime $primes {
                puts [isprime $prime]
        }
}

proc rsa-encrypt-decrypt-test {} {
        #set i  1
        set N1(0) 11
        set N1(1) 13
        set N1(2) 29
        set N1(3) 17
        set N1(4) 13
        set N1(5) 19
        set N1(6) 19
        set N1(7) 11
        set N1(8) 13
        set N1(9) 11
        set N1(10) 31
        set N1(11) 29
```

```
set N1(12) 233
set N1(13) 509
set N1(14) 941
set N1(15) 1031
set N1(16) 1237
set N1(17) 1933
set N1(18) 5011
set N1(19) 8233
set N1(20) 12853
set N1(21) 15443
set N1(22) 24533
set N1(23) 51511
set N1(24) 99991
set N1(25) 186653
set N1(26) 486637
set N1(27) 231859
set N1(28) 399989
set N1(29) 565427
set N1(30) 70001
set N1(31) 799993
set N1(32) 3641718173

set N2(0) 2
set N2(1) 19
set N2(2) 23
set N2(3) 13
set N2(4) 19
set N2(5) 17
set N2(6) 23
set N2(7) 23
set N2(8) 17
set N2(9) 23
set N2(10) 11
set N2(11) 11
set N2(12) 239
set N2(13) 521
```

```
set N2(14) 947
set N2(15) 1033
set N2(16) 1249
set N2(17) 1931
set N2(18) 5021
set N2(19) 8231
set N2(20) 12841
set N2(21) 15439
set N2(22) 19207
set N2(23) 51503
set N2(24) 99989
set N2(25) 186671
set N2(26) 486617
set N2(27) 231841
set N2(28) 486617
set N2(29) 565429
set N2(30) 699967
set N2(31) 799999
set N2(32) 2276689159

set plaintext {}
set plain_data {}
set ciphertext {}

#step #1 :to spifey bautrate and sampe per packet
set     BaudRate 60000
set sample_per_Packet 3000

for {set index 0} {$index < $sample_per_Packet} {incr index} {
        set temp_1 [expr rand() * 1000]
        set temp_2 [expr round($temp_1)]
        set plain_bytes [expr $temp_2 % 256]
        lappend plain_data $plain_bytes
}
puts "plain_data: $plain_data"
puts ""
```

```
#step #2 :encrypt
for {set j 1} {$j <12} {incr j} {
        # generate keys
        rsa-keygen $N1($j) $N2($j)

        variable pubkey
        variable prvkey
        set e [lindex $pubkey 0]
        set d [lindex $prvkey 0]
        set n [lindex $pubkey 1]

        set encrypted_data {}
        set decrypted_data {}

        set T1 [clock microseconds]

        foreach plain_bytes $plain_data {
                set plain_int [expr int($plain_bytes)]
                set encrypted_bytes [expr ($plain_int ** $e) % $n]
                lappend encrypted_data $encrypted_bytes
        }
        puts "encrypted_data: $encrypted_data"

        set T2 [clock microseconds]

        #puts "T1:$T1"
        #puts "T2:$T2"

        set maxmax 0
        foreach encrypted_bytes $encrypted_data {
                if {[expr { $encrypted_bytes > $maxmax }]} {
                        set maxmax [expr $encrypted_bytes]
                }
        }


          #xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

          if { $maxmax < 256 } {
                  set byte_per_sample 1
          } elseif { $maxmax < 65535 } {
                  set byte_per_sample 2
          } elseif { $maxmax < 1677215 } {
                  set byte_per_sample 3
          }

          set bytes_per_Packet [expr $byte_per_sample * $sample_per_Packet]
          set Trans_Time [expr $bytes_per_Packet * 10.0 / $BaudRate]

          #puts "WORKS"
          #puts "maxmax: $maxmax"
          #puts "byte_per_sample: $byte_per_sample"
          #puts "sample_per_Packet: $sample_per_Packet"
          #puts "bytes_per_Packet: $bytes_per_Packet"
          #puts "BaudRate: $BaudRate"
          puts "Trans_Time : $Trans_Time "

          #xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
          #xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

          #step #3: decrypt
          set T3 [clock microseconds]
          foreach encrypted_bytes $encrypted_data {
                  # c^d mod phi
                  set encrypted_int [expr int($encrypted_bytes)]
                  set decrypted_bytes [expr ($encrypted_int ** $d) % $n]
                  lappend decrypted_data $decrypted_bytes
          }
          set T4 [clock microseconds]

          puts "decrypted_data: $decrypted_data"
          puts "Enc Time [expr {($T2- $T1)}] microseconds per iteration"
```

```
            puts "decrypted_data: $decrypted_data"
            puts "Enc_Time [expr {($T2- $T1)}] microseconds per iteration"
            puts "Dcr_Time [expr {($T4- $T3)}] microseconds per iteration"
            #puts "$sample_per_Packet\t\t$bytes_per_Packet\t\tEnc_Time\tTrans_Time\tDcr_Time\t"

            puts ""
            puts "======================================================"
            puts ""
        }

        puts "Job Finished"
}

proc rsa-keygen {p q} {
        # computes RSA keys as per Burt Kaliski at RSA Labs
        # http://www.mathaware.org/mam/06/Kaliski.pdf

        # tries to select large primes p and q
        # computes n as pq
        # computes e to be coprime with p-1, q-1
        # sometimes fails at this
        # computes d to be the modular inverse of e mod n
        # sets the pubkey variable to be e, n
        # sets the prvkey variable to be d, n

        set prand $p
        set qrand $q
        set n [expr $prand * $qrand]
        set phi [expr ($prand - 1) * ($qrand - 1)]

        # calculate e
        for {set i 2 } {$i < $phi} {incr i} {
                set gcd_n [Crypto::gcd $i $n]
                set gcd_phi [Crypto::gcd $i $phi]
                if {$gcd_n != 1 || $gcd_phi != 1} {
```

```
                # calculate e
                for {set i 2 } {$i < $phi} {incr i} {
                        set gcd_n [Crypto::gcd $i $n]
                        set gcd_phi [Crypto::gcd $i $phi]
                        if {$gcd_n != 1 || $gcd_phi != 1} {
                                continue
                        }
                        set e $i
                        break
                }

                # calculate d
                for {set i 1} {$i < 9999999} {incr i} {
                        set temp_1 [expr ($i * $e) % $phi]
                        if {$temp_1 != 1} {
                                continue
                        }
                        set d $i
                        break
                }

                puts "p: $prand, q: $qrand, n: $n, phi: $phi, e: $e, d: $d"

                variable pubkey
                variable prvkey
                set pubkey "$e $n"
                set prvkey "$d $n"
        }
}

proc isperfectsquare {testnumber} {
        # check if a number is a perfect square
        variable square
        set square [::tcl::mathfunc::sqrt $testnumber]
        return [expr abs($square - int($square)) > 0 ? 0 : 1]
```

```
        }

        proc gcd {p q} {
                while 1 {
                        if {![set p [expr {$p % $q}]]} {return [expr { $q>0 ? $q : -$q }]}
                        if {![set q [expr {$q % $p}]]} {return [expr { $p>0 ? $p : -$p }]}
                }
        }

        proc gcdExt {a b} {
                if {$b == 0} {
                        return [list 1 0 $a]
                }
                set q [expr {$a / $b}]
                set r [expr {$a % $b}]
                lassign [gcdExt $b $r] s t g
                return [list $t [expr {$s - $q*$t}] $g]
        }

        proc gcdB {p q} {
     set p [expr {abs($p)}]
     set q [expr {abs($q)}]
     while {$q != 0} {
        set r [expr {$p % $q}]
        set p $q
        set q $r
     }
     set p
  }
        proc modInv {a m} {
                lassign [gcdExt $a $m] i -> g
                if {$g != 1} {
                        return -code error "no inverse exists of $a %! $m"
                }
                while {$i < 0} {
                        incr i $m
        set r [expr {$p % $q}]
        set p $q
        set q $r
     }
     set p
  }
        proc modInv {a m} {
                lassign [gcdExt $a $m] i -> g
                if {$g != 1} {
                        return -code error "no inverse exists of $a %! $m"
                }
                while {$i < 0} {
                        incr i $m
                }
                return $i
        }

        proc lcm {p q} {
                set m [expr {$p * $q}]
                if {!$m} {return 0}
                while 1 {
                        set p [expr {$p % $q}]
                        if {!$p} { return [expr {$m / $q}] }
                        set q [expr {$q % $p}]
                        if {!$q} { return [expr {$m / $p}] }
                }
        }

        proc randrange {min max} {
                # get a random number within a range
                return [expr {int(rand() * ($max - $min + 1)) + $min}]
        }
}
```

Crypto::RSA::rsa-encrypt-decrypt-test

```
hadeel@DESKTOP-2JMNS12:/mnt/c/ns-allinone-2.35$ ns swish044.tcl
plain_data: 171 67 194 24 95 71 247 78 146 79 186 155 209 138 234 26 67 156 200 107 189 194 151 134 76 227 185 193 45 42 29 28 13 72 80 20 133 107 112 79 32 224 155 47
220 217 46 213 176 132 112 246 22 100 191 167 108 157 239 191 77 158 206 86 131 223 123 4 84 32 88 115 98 150 253 142 98 108 50 8 132 194 161 64 178 136 80 26 131 34 82
 90 43 128 134 21 197 158 41 203 30 128 169 144 125 14 209 154 156 243 128 164 72 249 82 41 92 217 241 155 100 27 113 155 109 215 69 63 132 54 20 80 156 168 245 242 163
 179 244 19 174 135 70 151 177 53 2 254 130 232 202 212 207 111 176 200 212 50 86 106 72 15 33 204 154 112 225 153 16 90 14 68 182 234 112 211 2 59 117 215 203 192 78 1
59 71 219 34 64 100 162 139 238 246 226 172 34 29 245 219 204 111 143 162 188 152 85 170 46 119 196 121 140 163 246 18 39 72 105 243 252 78 246 145 88 64 90 197 55 179
232 196 104 206 56 165 144 44 50 186 186 224 194 165 25 50 206 190 165 123 140 177 77 64 221 155 18 84 200 33 61 218 1 176 129 59 93 247 60 225 9 200 132 9 177 145 225
106 24 95 183 206 127 53 188 16 17 226 46 80 133 160 212 123 50 162 233 4 251 228 192 16 110 101 65 214 153 30 104 112 154 154 26 101 121 127 47 8 113 0 180 116 4 93 16
8 169 140 84 181 182 67 143 250 80 33 158 131 88 34 30 215 66 66 102 70 240 170 108 39 225 228 160 94 203 204 152 109 164 218 98 220 141 42 46 139 228 221 40 60 94 91 3
9 178 84 210 108 107 209 177 90 68 34 251 123 172 196 15 158 58 66 115 99 59 15 20 61 13 208 49 253 26 85 18 50 194 117 200 228 250 136 87 107 183 109 52 123 132 182 19
9 78 34 68 221 104 102 3 62 118 130 40 69 27 26 137 6 205 47 77 148 141 211 26 72 255 69 187 161 78 123 180 132 74 158 28 58 174 181 131 138 171 18 61 101 151 219 126 8
6 205 142 40 180 47 173 75 62 16 175 203 157 195 34 10 20 81 96 245 104 204 216 53 194 232 159 125 101 152 38 184 109 5 75

======================================================

p: 13, q: 19, n: 247, phi: 216, e: 5, d: 173
encrypted_data: 19 136 207 85 114 67 0 13 204 53 192 129 209 47 195 182 136 169 174 217 37 207 151 153 228 132 48 72 106 74 22 149 52 154 188 115 152 217 177 53 223 230
 129 73 64 107 50 226 180 227 177 246 224 237 172 59 166 66 83 172 77 214 137 60 196 162 54 36 145 223 160 20 167 63 119 168 167 166 46 164 227 207 187 220 68 15 188 18
2 196 21 62 181 218 124 153 203 201 214 110 242 140 124 234 235 216 105 209 241 169 211 124 8 154 32 62 110 118 107 128 129 237 183 94 129 200 24 179 176 227 175 115 18
8 169 194 215 86 102 69 4 171 148 70 109 151 138 40 32 11 156 150 141 205 25 232 180 174 205 46 60 45 154 97 219 29 241 177 23 134 61 181 105 178 26 195 177 165 32 89 9
1 24 242 108 13 87 67 98 21 220 237 41 81 231 246 44 191 21 22 215 98 29 232 117 41 158 133 206 170 50 123 157 49 30 102 246 18 39 154 79 211 161 13 246 84 160 220 181
201 139 69 150 157 130 137 75 185 235 5 46 192 192 230 207 185 233 46 137 190 185 54 30 138 77 220 65 129 18 145 174 219 55 225 1 180 116 89 6 0 34 23 16 174 227 16 138
 84 23 45 85 114 27 137 147 40 158 61 101 44 50 188 152 88 205 54 46 41 142 36 36 76 108 61 2 43 221 28 134 140 130 177 241 241 182 43 49 147 73 164 94 0 111 51 36 6 19
4 234 30 145 155 26 136 117 243 188 219 214 196 160 21 140 24 92 92 163 109 236 170 166 39 23 76 88 113 242 29 133 200 8 225 167 64 202 74 50 81 76 65 222 34 113 78 39
68 145 58 166 217 209 138 181 178 21 36 54 191 157 97 214 210 92 20 112 89 97 115 55 52 208 121 119 182 206 18 46 207 91 174 76 243 15 159 217 27 200 143 54 227 26 244
13 21 178 65 130 163 243 199 131 156 222 179 183 182 245 119 173 73 77 135 202 165 182 154 164 179 213 187 13 54 111 227 120 214 149 210 148 155 196 47 19 18 55 43 151
98 198 60 173 168 222 111 73 127 56 199 61 93 242 66 104 21 212 115 9 96 215 130 29 125 40 207 150 87 216 43 133 38 71 200 161 56
Trans_Time : 0.083333333333333329
decrypted_data: 171 67 194 24 95 71 0 78 146 79 186 155 209 138 234 26 67 156 200 107 189 194 151 134 76 227 185 193 45 42 29 28 13 72 80 20 133 107 112 79 32 224 155 4
7 220 217 46 213 176 132 112 246 22 100 191 167 108 157 239 191 77 158 206 86 131 223 123 4 84 32 88 115 98 150 6 142 98 108 50 8 132 194 161 64 178 136 80 26 131 34 82
 90 43 128 134 21 197 158 41 203 30 128 169 144 125 14 209 154 156 243 128 164 72 2 82 41 92 217 241 155 100 27 113 155 109 215 69 63 132 54 20 80 156 168 245 242 163 1
79 244 19 174 135 70 151 177 53 2 7 130 232 202 212 207 111 176 200 212 50 86 106 72 15 33 204 154 112 225 153 16 90 14 68 182 234 112 211 2 59 117 215 203 192 78 159 7
1 219 34 64 100 162 139 238 246 226 172 34 29 245 219 204 111 143 162 188 152 85 170 46 119 196 121 140 163 246 18 39 72 105 243 5 78 246 145 88 64 90 197 55 179 232 19
6 104 206 56 165 144 44 50 186 186 224 194 165 25 50 206 190 165 123 140 177 77 64 221 155 18 84 200 33 61 218 1 176 129 59 93 0 60 225 9 200 132 9 177 145 225 106 24 9
5 183 206 127 53 188 16 17 226 46 80 133 160 212 123 50 162 233 4 4 228 192 16 110 101 65 214 153 30 104 112 154 154 26 101 121 127 47 8 113 0 180 116 4 93 168 169 140
84 181 182 67 143 3 80 33 158 131 88 34 30 215 66 66 102 70 240 170 108 39 225 228 160 94 203 204 152 109 164 218 98 220 141 42 46 139 228 221 40 60 94 91 39 178 84 210
 108 107 209 177 90 68 34 4 123 172 196 15 158 58 66 115 99 59 15 20 61 13 208 49 6 26 85 18 50 194 117 200 228 3 136 87 107 183 109 52 123 132 182 199 78 34 68 221 104
 102 3 62 118 130 40 69 27 26 137 6 205 47 77 148 141 211 26 72 8 69 187 161 78 123 180 132 74 158 28 58 174 181 131 138 171 18 61 101 151 219 126 86 205 142 40 180 47
173 75 62 16 175 203 157 195 34 10 20 81 96 245 104 204 216 53 194 232 159 125 101 152 38 184 109 5 75
Enc_Time 3563 microseconds per iteration
Dcr_Time 8184 microseconds per iteration
```

```
p: 29, q: 23, n: 667, phi: 616, e: 3, d: 411
encrypted_data: 379 613 402 484 280 399 359 315 581 126 307 14 100 92 501 234 613 519 2 431 562 402 564 235 90 571 461 131 413 51 377 608 196 395 411 663 128 431 226 12
6 85 474 14 438 12 540 621 101 385 152 226 163 643 167 389 469 416 626 430 389 305 341 114 405 301 25 604 64 408 85 465 115 55 647 184 524 55 416 271 512 152 402 529 13
 267 199 411 234 301 618 426 636 134 104 235 590 219 341 220 580 320 104 397 492 149 76 100 439 519 403 104 73 395 534 426 220 299 540 526 14 167 340 176 14 382 75 345
589 152 52 663 411 519 596 109 72 583 473 191 189 58 479 162 564 462 136 8 208 569 261 289 33 644 281 385 2 33 271 405 421 395 40 586 88 439 226 266 454 94 636 76 275 2
22 501 226 570 8 610 146 75 580 351 315 337 399 210 618 13 167 70 277 535 163 74 572 618 377 109 210 88 281 79 70 18 53 485 545 621 317 440 9 629 583 163 496 623 395 38
0 403 344 315 163 435 465 13 636 219 292 473 261 440 302 114 195 547 492 475 271 307 307 474 402 547 284 271 114 239 547 604 629 462 305 13 467 14 496 408 2 586 201 388
 1 385 283 610 622 359 559 266 62 2 152 62 462 435 266 421 484 280 91 114 26 136 18 94 244 74 621 411 128 620 33 604 271 70 349 64 15 429 351 94 335 453 488 113 454 320
 302 226 439 439 234 453 9 26 438 512 176 0 419 116 64 622 596 397 629 408 111 222 613 79 525 411 586 341 301 465 618 320 75 19 19 11 162 425 545 416 623 266 429 620 16
9 580 88 53 382 73 388 55 12 487 51 621 277 429 467 635 559 169 528 623 267 408 372 416 431 100 462 636 275 618 15 604 572 440 40 341 348 19 115 481 610 40 663 201 196
415 257 184 234 485 496 271 402 146 2 429 525 199 174 431 91 382 538 604 152 222 661 315 618 275 467 302 11 27 209 211 569 635 345 340 234 68 216 153 438 305 172 487 57
0 234 395 422 345 602 529 315 604 419 152 355 341 608 348 58 111 301 92 379 496 201 453 564 210 43 405 153 524 635 419 438 463 331 209 94 30 580 626 503 618 333 663 509
 294 109 302 88 660 136 402 261 337 149 453 53 178 391 382 125 331
Trans_Time : 0.16666666666666666
decrypted_data: 171 67 194 24 95 71 247 78 146 79 186 155 209 138 234 26 67 156 200 107 189 194 151 134 76 227 185 193 45 42 29 28 13 72 80 20 133 107 112 79 32 224 155
 47 220 217 46 213 176 132 112 246 22 100 191 167 108 157 239 191 77 158 206 86 131 223 123 4 84 32 88 115 98 150 253 142 98 108 50 8 132 194 161 64 178 136 80 26 131 3
4 82 90 43 128 134 21 197 158 41 203 30 128 169 144 125 14 209 154 156 243 128 164 72 249 82 41 92 217 241 155 100 27 113 155 109 215 69 63 132 54 20 80 156 168 245 242
 163 179 244 19 174 135 70 151 177 53 2 254 130 232 202 212 207 111 176 200 212 50 86 106 72 15 33 204 154 112 225 153 16 90 14 68 182 234 112 211 2 59 117 215 203 192
78 159 71 219 34 64 100 162 139 238 246 226 172 34 29 245 219 204 111 143 162 188 152 85 170 46 119 196 121 140 163 246 18 39 72 105 243 252 78 246 145 88 64 90 197 55
179 232 196 104 206 56 165 144 44 50 186 186 224 194 165 25 50 206 190 165 123 140 177 77 64 221 155 18 84 200 33 61 218 1 176 129 59 93 247 60 225 9 200 132 9 177 145
225 106 24 95 183 206 127 53 188 16 17 226 46 80 133 160 212 123 50 162 233 4 251 228 192 16 110 101 65 214 153 30 104 112 154 154 26 101 121 127 47 8 113 0 180 116 4 9
3 168 169 140 84 181 182 67 143 250 80 33 158 131 88 34 30 215 66 66 102 70 240 170 108 39 225 228 160 94 203 204 152 109 164 218 98 220 141 42 46 139 228 221 40 60 94
91 39 178 84 210 108 107 209 177 90 68 34 251 123 172 196 15 158 58 66 115 99 59 15 20 61 13 208 49 253 26 85 18 50 194 117 200 228 250 136 87 107 183 109 52 123 132 18
2 199 78 34 68 221 104 102 3 62 118 130 40 69 27 26 137 6 205 47 77 148 141 211 26 72 255 69 187 161 78 123 180 132 74 158 28 58 174 181 131 138 171 18 61 101 151 219 1
26 86 205 142 40 180 47 173 75 62 16 175 203 157 195 34 10 20 81 96 245 104 204 216 53 194 232 159 125 101 152 38 184 109 5 75
Enc_Time 3135 microseconds per iteration
Dcr_Time 21247 microseconds per iteration

======================================================

p: 17, q: 13, n: 221, phi: 192, e: 5, d: 77
encrypted_data: 188 84 181 215 23 158 195 91 74 27 101 168 14 151 13 195 84 39 200 48 219 181 138 36 111 41 87 7 197 9 139 214 13 89 71 141 165 48 125 27 2 22 168 47 22
0 81 37 161 24 149 125 77 133 172 55 46 75 157 18 55 25 201 202 86 105 32 106 140 67 2 56 98 115 63 2 194 115 75 33 60 149 181 213 64 94 119 71 195 105 34 114 116 127 5
9 36 21 6 201 45 203 166 59 169 196 177 131 14 137 39 133 59 112 89 214 114 45 79 81 141 168 172 40 146 168 96 180 205 20 149 175 141 71 39 155 215 21 193 212 160 15 17
4 135 83 138 164 66 32 50 78 163 206 179 90 76 24 200 179 33 86 123 89 19 50 68 137 125 140 17 152 116 131 204 156 13 125 113 32 128 104 180 203 82 91 126 158 189 34 64
 172 93 107 153 77 31 100 34 139 215 189 68 76 130 93 171 16 102 170 37 136 144 49 4 193 77 18 65 89 209 133 148 91 77 110 56 64 116 6 191 212 163 144 117 202 218 3 196
 57 33 101 101 22 181 3 77 33 202 73 3 106 4 164 25 64 0 168 18 67 200 50 159 199 1 24 142 128 162 195 8 140 42 200 149 42 164 110 140 123 215 23 183 202 43 66 171 152
153 31 37 71 165 62 179 106 33 93 207 140 166 11 82 152 145 186 182 210 17 166 117 125 137 137 195 186 49 43 47 60 146 0 176 12 140 162 155 169 4 67 129 156 84 130 139
71 50 201 105 56 34 166 180 53 53 85 83 15 170 75 65 140 11 62 178 203 68 16 96 112 199 115 220 150 9 37 107 11 0 92 8 178 143 65 94 67 58 75 48 14 164 116 204 34 166 1
06 100 144 19 201 28 53 98 216 128 19 141 159 13 208 121 2 195 102 18 33 181 104 200 11 139 119 185 48 183 96 52 106 149 156 88 91 34 204 0 117 85 22 95 118 78 92 205 4
0 195 154 41 69 47 25 122 150 113 195 89 34 205 187 213 91 106 176 149 211 201 214 28 174 129 105 151 188 18 159 186 138 189 198 86 69 194 92 176 47 192 147 95 152 184
203 157 26 34 108 141 217 44 215 117 68 190 66 181 163 126 177 186 16 38 97 96 31 147
Trans_Time : 0.083333333333333329
```

Appendix B. Optimal Solution

```
clc
clear all
close all
%132

 Max_Time = 150;
 Max_Security_667 = 667;
 Min_Security_221 = 221;


T500 = [8.46208333 9.6491666 10.4910833 12.5050833 13.1240833 14.05708333 18.27416 28.782166];
T700 = [11.184233 11.424166 14.4501166 17.129233 18.0761666 18.402233 25.989233 34.657233];
T1000 = [17.63033 18.449166 23.477166 26.202166 27.440333 32.76133 42.59933 57.31233];
T1500 = [24.2655 24.46925 30.68025 34.09725 36.8465 38.1745 54.7675 75.7145];
T2000 = [27.63966 35.52233 42.09233 43.13333 43.69066 50.47766 73.61766 101.52466];
T3000 = [46.128 52.6295 59.2335 63.4595 68.055 69.595 105.684 143.967];
N = [221 247 253 319 323 341 437 667];

cof500 = polyfit(T500,N,3);
cof700 = polyfit(T700,N,3);
cof1000 = polyfit(T1000,N,3);
cof1500 = polyfit(T1500,N,3);
cof2000 = polyfit(T2000,N,3);
cof3000 = polyfit(T3000,N,3);

TT500 = [8:30];
TT700 = [11:35];
```

```
TT1000 = [17:58];
TT1500 = [24:76];
TT2000 = [27:102];
TT3000 = [46:144];
NN500 = polyval(cof500, TT500);% Quality Minmum curve
NN700 = polyval(cof700, TT700);
NN1000 = polyval(cof1000, TT1000);
NN1500 = polyval(cof1500, TT1500);
NN2000 = polyval(cof2000, TT2000);
NN3000 = polyval(cof3000, TT3000);%Quality maximum curve


plot(T500,N,'-o', TT500, NN500, ...
     T700,N,'-o', TT700, NN700, ...
     T1000,N,'-o', TT1000, NN1000, ...
     T1500,N,'-o', TT1500, NN1500, ...
     T2000,N,'-o', TT2000, NN2000, ...
     T3000,N,'-o', TT3000, NN3000);

 legend('500','500','700','700','1000','1000','1500','1500','2000','2000','3000','3000');


figure, plot( TT500, NN500, ...
     TT700, NN700, ...
     TT1000, NN1000, ...
      TT1500, NN1500, ...
      TT2000, NN2000, ...
      TT3000, NN3000);

  legend('500','700','1000','1500','2000','3000');


 %Generate Quality Map
 Q_Map = zeros(700,150);
 QQ_Map = zeros(700,150);
 QQQ_Map = zeros(700,150);

 for i=1:0.01:150
     X = round(i);
     Y500 = round(polyval(cof500, i));
     Y700 = round(polyval(cof700, i));
     Y1000 = round(polyval(cof1000, i));
     Y1500 = round(polyval(cof1500, i));
     Y2000 = round(polyval(cof2000, i));
     Y3000 = round(polyval(cof3000, i));

     if((Y500>=Min_Security_221)&&(Y500<=Max_Security_667))
         Q_Map(Y500,X) = 500;
     end
     if((Y700>=Min_Security_221)&&(Y700<=Max_Security_667))
         Q_Map(Y700,X) = 700;
     end

       if((Y1000>=Min_Security_221)&&(Y1000<=Max_Security_667))
           Q_Map(Y1000,X) = 1000;
       end
       if((Y1500>=Min_Security_221)&&(Y1500<=Max_Security_667))
           Q_Map(Y1500,X) = 1500;
       end
       if((Y2000>=Min_Security_221)&&(Y2000<=Max_Security_667))
           Q_Map(Y2000,X) = 2000;
       end
       if((Y3000>=Min_Security_221)&&(Y3000<=Max_Security_667))
           Q_Map(Y3000,X) = 3000;
       end
   end

   Peaks = [500 700 1000 1500 2000 3000 1000000];
   u=1;
   Edge_Start=Peaks(u);
   Edge_End=Peaks(u+1);

   for Y = 1:700
       flag1 = 0;
       flag2 = 0;
       u=1;
       Edge_Start=Peaks(u);
       Edge_End=Peaks(u+1);
       for X=1:150
           if(flag1==0)
```

```
            if(Q_Map(Y,X)==Edge_Start)
                flag1=X;
            end
        elseif((flag2==0)&&(flag1~=0))
            if(Q_Map(Y,X)==Edge_End)
                flag2=X;
            end
        elseif((flag2~=0)&&(flag1~=0))
            m = (Edge_End-Edge_Start)/(flag2-flag1);
            Z0 = Edge_Start-m*flag1;
            for i=flag1+1:flag2-1
                Q_Map(Y,i) = i*m+Z0;
            end
            flag1=flag2;
            flag2=0;
            u=u+1;
            Edge_Start=Peaks(u);
            Edge_End=Peaks(u+1);
        end
    end
end

figure, surf(Q_Map);

%Generate Cost Map : Cost = Normalized Security / Normalized Quality
%Balance between Security and and Quality when cost close to 1.
Cost_Map = zeros(700,150);
```

```
for X=1:150
    for Y=1:700
        Norm_Security = (Y - Min_Security_221)/(Max_Security_667 - Min_Security_221);
        Norm_Quality = (Q_Map(Y,X)-500)/(3000 - 500);
        if(Norm_Quality>0)
            Cost_Map(Y,X) = Norm_Security/Norm_Quality;
        end
    end
end

figure, surf(Cost_Map);


%Optimal Cost Function
N=1;
for X=1:150
    for Y=1:700
        if((Cost_Map(Y,X)<1.01)&&(Cost_Map(Y,X)>0.99))
            Optimal_X(N) = X;%Time
            Optimal_Y(N) = Y;%Security
            Optimal_Z(N) = Q_Map(Y,X);%Quality (packet size)
            N=N+1;
        end
    end
end
```

```
figure, plot( TT500, NN500, ...
     TT700, NN700, ...
     TT1000, NN1000, ...
     TT1500, NN1500, ...
     TT2000, NN2000, ...
     TT3000, NN3000, ...
     Optimal_X,Optimal_Y, '-o');

 legend('500','700','1000','1500','2000','3000', 'Optimal');


%Ex1: How to calculate New parameters:
 %Given dT
 Time_Initial = 100;
 dT = 140;
 ZZZ_Time_Update = Time_Initial+ (Max_Time - dT);
 for i=1:N
     if(Optimal_X(i)>=ZZZ_Time_Update)
         ZZZ_Best_Security = Optimal_Y(i);
         ZZZ_Best_Quality = Optimal_Z(i);
         break;
     end

 end
```