Enhanced Secured Voice over Internet Protocol

Prepared by

Omer M. A. Abu Shqeer

Supervisors

Prof. Naim Ajlouni

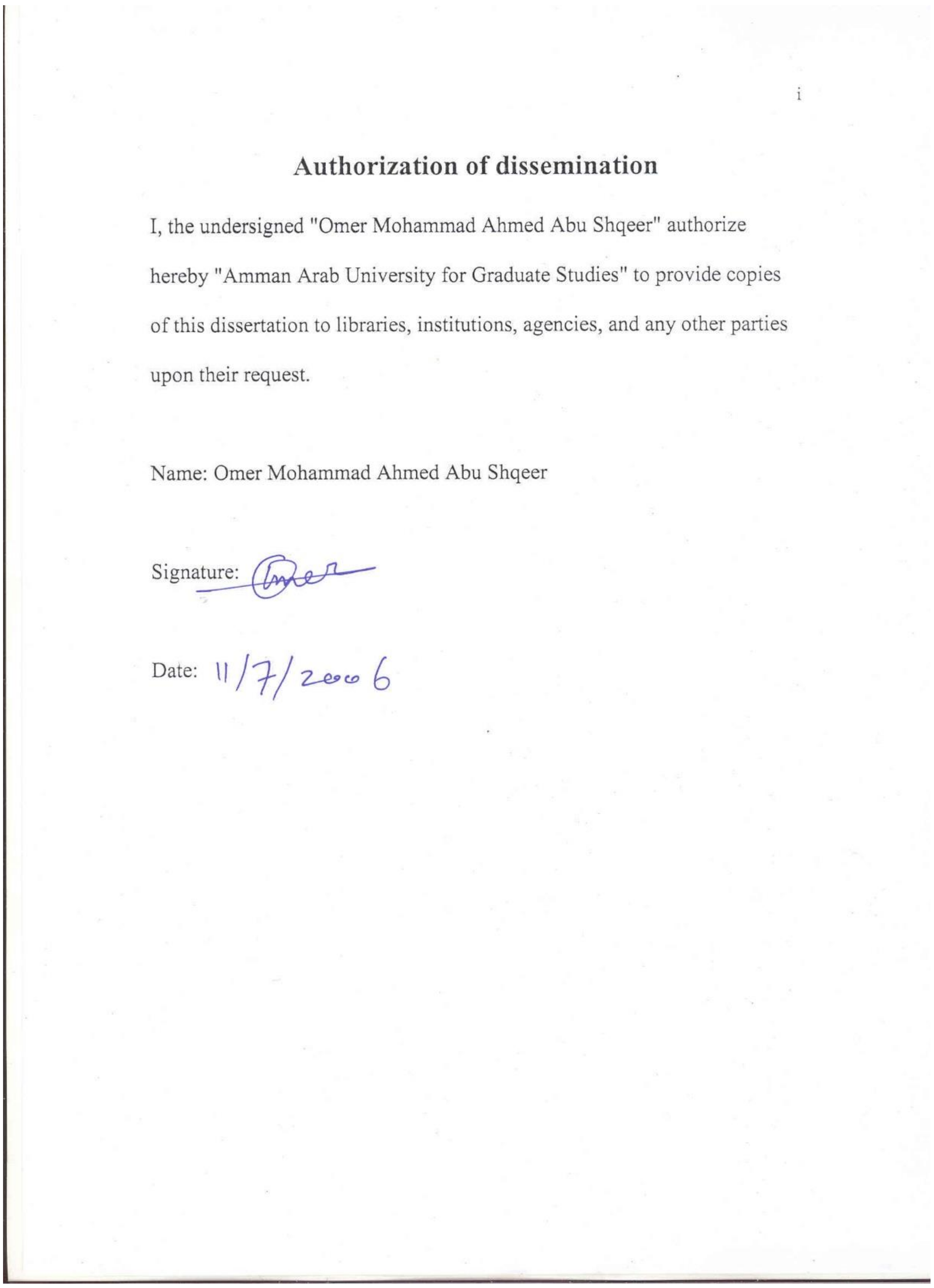
Dr. Basil Kasasbeh

A dissertation submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science.

Graduate College of Computer Studies

Amman Arab University for Graduate Studies

July, 2006



Authorization of dissemination

I, the undersigned "Omer Mohammad Ahmed Abu Shqeer" authorize hereby "Amman Arab University for Graduate Studies" to provide copies of this dissertation to libraries, institutions, agencies, and any other parties upon their request.

Name: Omer Mohammad Ahmed Abu Shqeer

Signature: [illegible]

Date: 11/7/2006

Resolution of the examining committee

This dissertation titled "Enhanced Secured Voice over Internet Protocol ", has been defended and approved on 11/7/2006.

Examining Committee	Title	Signature
Prof. Hilal M. Al-Bayatti	President	[illegible]
Prof. Naim Ajlouni	Member and supervisor	[illegible]
Dr. Sadeq Al Hamouz	Member	[illegible]
Dr. Moayad A. Fadhil	Member	[illegible]

Acknowledgment

"All praises and thanks to ALLAH"

I would like to thank my supervisors Dr. Naim Ajlouni and Dr. Basil Kasasbeh who encouraged and helped me very much in doing this research; they were and are still my brothers.

A great thanks from my heart to my parents for their prayers, my wife for her patience and support, my son and daughters for their patience too, and my father in law, mother in law, brothers, sisters, friends, and colleagues for their encouragements.

Finally, I would like to thank all lecturers, administration, and staff of Amman Arab University for Graduate Studies for their help and support.

Dedication

I dedicate this work to my loves:

Parents, wife, son, daughters, brothers, sisters, and friends

Table of Contents

List of Tables

List of Figures

Acronyms and abbreviations

Acronym/Abbrev.	Description
A/D	Analog-to-Digital
AAUGS	Amman Arab University for Graduate Studies
AC	Authentication Centers
ACELP	Algebraic Code Excited Linear Predictive
ADPCM	Adaptive Differential Pulse Code Modulation
AES	Advanced Encryption Standard
CA	Certificate Authorities
CBC	Ciphertext Block Chaining
CC	CSRC Count
CFB	Ciphertext FeedBack
CS-ACELP	Conjugate Structure Algebraic Code Excited Linear Predictive
CSRC	Contributing Source
DA	Destination Address
D/A	Digital-to-Analog
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DIX	DEC, Intel, Xerox
DS	Differential Service
DSP	Digital Signal Processing
ECN	Explicit Congestion Notification
EF	Expedited Flow
FCS	Frame Check Sequence
FIPS	Federal Information Processing Standards
FTP	File Transfer Protocol
GB	Gigabyte
GHz	Giga Hertz
HDD	Hard Disk

IAB	Internet Architecture Board
ICR	Interface Clock Rate
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPSec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
IV	Initialization Vector
Kbps	Kilo bits per second
LAN	Local Area Network
LDAU	Last Date Added Users
LFI	Link Fragmentation and Interleaving
MB	Megabyte
Mbps	Mega bits per second
MOS	Mean Opinion Score
MP-MLQ	Multi-Pulse Maximum Likelihood Quantization
MTU	Maximum Transmission Unit
NIST	National Institute of Standards and Technology
PBX	Private Branch eXchange
PC	Personal Computer
PCM	Pulse Code Modulation
PLC	Packets Loss Concealment
PSTN	Public Switched Telephone Network
PT	Payload Type
QoS	Quality of Service
RAM	Random Access Memory

RAS	(Registration, Admission, and Status) Signaling
RFC	Request for Comments
RSA	Rivest-Shamir-Adleman
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
RTPC	Real-time Transport Protocol Header Compression
SA	Source Address
SDK	Software Developer's Kit
SFD	Start of Frame Delimiter
SIP	Session Initiation Protocol
SSRC	Synchronization Source
TCP	Transmission Control Protocol
TOS	Type of Service
UDP	User Datagram Protocol
URL	Universal Resource Locator
VAD	Voice Activity Detection
VoIP	Voice over Internet Protocol
WAN	Wide Area Network

Researcher abbreviations

Abbrev.	Description
ATV	Autocorrelation Test Value
DOKS	Direction of the Key Selection
DOKSS	Direction of the Key Segments Selection
FTV	Frequency Test Value
FS	Frame Size
LA	Look-Ahead time
LOKSC	Length of the Key Segments Categories
LS	Line Speed
MD	Maximum Delay
MKL	Master Key Length
mD	Minimum Delay
NCU	Number of Created Users
ND	Network Delay
NODFKS	Number Of Directions For Key Selection
NODFKSS	Number Of Directions For Key Segment Selection
NOKLC	Number Of Key Length Categories
NOKSC	Number Of Key Segment Categories
NOKSLC	Number Of Key Segment Lengths Categories
NOPER	Number Of PERmutations
NOPFUV	Number Of Packets For Useful/Understandable Voice
NOPK	Number Of Possible Keys
NOPVF	Number Of Preceding Voice Frames
NR	Number of Routers
NRU	Number of Registered Users
NU	Number of Users
PD	Processing Delay

PgD	Propagation Delay
PTV	Poker Test Value
QD	Queuing Delay
STV	Serial Test Value
SD	Serialization Delay
TNOPK	Total Number Of Possible Keys

Abstract

Voice over Internet Protocol (VoIP) is the technology that enables voice packets transmission over Internet Protocol (IP). Security is of concern whenever open networks are to be used. VoIP suffers from packet latency and loss due to the nature of IP network. Cryptographic systems may be used to achieve VoIP security, but their impact on the Quality of Service (QoS) should be minimized. The fact that most of the known encryption algorithms are computationally expensive results in a significant amount of time added to packet delay. VoIP is usually used by public users resulting in a key exchange problem; a trusted intermediate authority normally takes this responsibility.

In this research, VoIP security was enhanced via a proposed cryptographic system. The proposed solution consists of a simple, but strong encryption/decryption algorithm as well as an embedded method to exchange the keys between users. In this research, new key is generated in a random fashion from a pool of master keys and then used to encrypt each new voice packet to strengthen the security level. The generated keys passed four implemented randomness statistical tests. Key exchange is carried out by mixing some information about the key with the ciphered voice data; this information is enough for a target receiver to regenerate/extract the key.

The proposed solution was implemented and tested, the results showed that the required time for the security processes is minimized compared to some known algorithms. The analysis improved that the security level has a direct relationship with the number of possible keys that can be generated from the master key; this number can be increased by increasing the value range of the factors that affect this number. Finally, a delay budget for VoIP on variety network configurations was presented for a variety of codecs; which can be used as guidance for VoIP developers.

رفع كفاءة السرية للمكالمات الصوتية عبر شبكة الانترنت

الملخص

Arabic Summary

إن (VoIP) هو مصطلح يستخدم للتعبير عن تراسل البيانات الصوتية عبر شبكة الانترنت. إن سرية المعلومات تعتبر من الأهمية بمكان حيثما استخدمت الشبكات المفتوحة لنقلها. إن تطبيق (VoIP) يعاني من التأخير في وصول حزم البيانات بل وفقدانها أحيانا. إن أنظمة تشفير البيانات يمكن أن تستخدم لتحقيق السرية في تطبيق (VoIP) إلا أن تأثيرها على نوعية الخدمة يجب أن يبقى في حده الأدنى. وبما أن خوارزميات التشفير المعروفة تستخدم عمليات حسابية طويلة فإنه ينتج عن استخدامها مع تطبيق (VoIP) زيادة معنوية على الوقت اللازم لوصول البيانات. إن تطبيق (VoIP) عادة ما يستخدم من قبل العموم ناتجا عن ذلك مشكلة في تبادل مفاتيح التشفير بين المستخدمين والذي يتم بالعادة عن طريق مؤسسة وسيطة موثوقة.

في هذه الأطروحة تم تحسين مستوى السرية في تطبيق (VoIP) من خلال نظام مقترح لتشفير البيانات الصوتية بطريقة سهلة وفعالة وكذلك طريقة ضمنية لتبادل المفاتيح بين المستخدمين. في الحل المقترح يتم توليد مفتاح جديد بطريقة عشوائية من سلة مفاتيح رئيسية ليستخدم في عملية التشفير مع كل رسالة صوتية جديدة من أجل رفع كفاءة مستوى السرية.إن المفاتيح التي يتم توليدها في الطريقة المقترحة إجتازت الاختبارات الإحصائية التي تم برمجتها لقياس مستوى العشوائية لهذه المفاتيح. أما عملية تبادل المفاتيح فإنها تتم من خلال مزج بعض المعلومات عن المفتاح مع البيانات المشفرة وهذه البيانات كافية للمستقبل لكي تعيد توليد المفتاح.

لقد تم برمجة واختبار الحل المقترح وقد أظهرت النتائج أن الوقت اللازم لتنفيذ إجراءات الحماية على البيانات تقلص إلى حد مقبول مقارنة ببعض الخوارزميات المعروفة. كما أثبت التحليل وجود علاقة مباشرة ما بين درجة السرية وعدد المفاتيح التي يمكن توليدها من المفتاح الرئيسي. إن عدد هذه المفاتيح يمكن زيادته بزيادة المديات لقيم العوامل التي يعتمد عليها هذا الرقم. وأخيرا فقد تم احتساب الوقت المتوقع أن تستغرقه حزمة البيانات في تطبيق (VoIP) على عدد من شبكات الحاسوب مما يمكن أن يستخدم كدليل لمطوري تطبيقات (VoIP).

Introduction

1.1. Introduction

Various multimedia applications over Internet Protocol (IP) such as Voice over Internet Protocol (VoIP) increase the user satisfaction on networks that are used for information exchange. However there is some concern of security whenever data is transmitted over an open network. Cryptography plays a major role in maintaining the secrecy of data, but new methods should be used with VoIP systems to minimize the effect of cryptographic process on end-to-end delay.

VoIP is the technology of transferring voice data over IP networks. The main advantages of this technology over the traditional Public Switched Telephone Network (PSTN) are cost, integration with other media services, portability, and bandwidth utilization. At the same time, this technology faces some challenges such as latency, packet loss, and security.

Any VoIP conversation is subject to be compromised by any user on the IP network. The hackers/intruders have sniffing tools; therefore, a security process should be applied on any important and valuable voice packet. Cryptography is widely used to protect data that traverse over open networks but most of the known encryption algorithms were built for text data; these algorithms consume significant amount of time due to their extensive computation; therefore these algorithms are not suitable to be used with VoIP which already suffers from latency. The encryption modes that depend on their decryption process of a packet on its preceding packet such as Cipher Block Chaining (CBC) and Cipher FeedBack (CFB) are not suitable to be used with VoIP security; this is because VoIP usually depends on an unreliable transmission channel and a packet loss yields an error in the decryption process of the next packet since there is no retransmission of the lost packet(s). Because of the above and other reasons an inexpensive, simple, and robust technique should be used in VoIP security.

1.2. Statement of the problem

Current data encryption/decryption techniques introduce an unacceptable delay when they are used with VoIP. In this research, a new encryption/decryption technique which adds a minimum overhead delay to a voice packet in VoIP application is to be introduced. VoIP is expected to replace the traditional PSTN. To transmit voice over data network it should be encoded (i.e. digitized). There exist a number of codecs; the selection of a voice coder among them is not an easy job since each of them has its own data rate, packetization delay, QoS, and bandwidth requirements; therefore, there is no single choice that is suitable for all VoIP deployments. It is also necessary to look for an efficient flexible selection of vocoder. Further more, most of the available VoIP applications transmit the voice packets with no security on the current data networks due to inherent issues of any IP service and unique requirements of VoIP including real-time needs [Collier]. Some data encryption/decryption techniques can be applied to the voice packets transferred over IP; but the problem here is the delay overload that comes from encryption/decryption processes; in this research, an encryption/decryption technique to secure VoIP packets that would not add a significant amount of delay is introduced.

Keys management and distribution form the main problem in the symmetric key encryption; in this research, a self organized and embedded method (i.e. without a third party) for the key distribution is introduced; this is done by mixing some information with the voice packet which gives the receiver the ability to extract the key. Further more, new key is used with each new packet to strengthen this security approach; the effect of this is the extra overhead added to the packet, but it is relatively small (only three bytes).

1.3. Goals of this research

The objectives of this research are summarized as follows:

1. Investigate the effects of encryption/decryption and key management processes on VoIP packets delay.
2. Propose a new encryption/decryption technique to maintain security on VoIP with minimum time delay.
3. Compare the delay introduced by the suggested encryption/decryption technique to the delay introduced by a known encryption/decryption algorithm(s).
4. Propose an embedded key management solution to VoIP system without an intermediate authority.
5. Estimate the delay budget of VoIP system using the suggested solution on variety network configurations.

1.4. Suggested solution

The proposed solution can be summarized as:

1. The sender/caller will issue a request for a call.
2. The receiver will issue a response.
3. If the receiver response is "OK" then a bidirectional call starts with the following operations:

   a. Voice collection.

4. b. Voice coding (A/D conversion and compression).

   c. Packetization.

   d. Encryption key selection at the sender side. An inexpensive, simple, strong, and flexible method was used for the key selection; different keys with different sizes for different packets are used in order to increase the security level.

   e. Encryption of the voice data with the selected key.

   f. Mixing the key information with the ciphered data in order to help the receiver to recompose the same key.

   g. Transmission of the mixed data.

   h. Extraction of the encryption key at the receiver side. Only the target user can extract the key.

   i. Decryption of the voice data with the extracted key.

   j. Voice decoding (decompression and D/A conversion)

5. One of the two parties issue termination.
6. Call finishes.

1.5. Summary

The main contribution of this research is the new encryption and key management method which enhances the VoIP security. This new method uses a simple and robust technique to encrypt the voice packets and applies an embedded method for key management. This new method satisfies the security goal without adding much time to the end-to-end delay of voice packets. Every new voice packet uses new key to strengthen the security level. Delay budget guidance for network designers is another contribution of this research.

1.6. Dissertation structure

The dissertation is structured as follows:

- Chapter two provides an overview of communication convergence, PSTN, VoIP, and cryptographic. It introduces the main concepts, advantages, disadvantages, challenges, security, coders/decoders (codecs) properties, Quality of Service (QoS) measures, end-to-end delay sources, and the related protocols of VoIP. Also it introduces the main concepts of the traditional encryption techniques and their effect on the voice delay if they are used to encrypt the voice over IP. Furthermore, it introduces some previous related works.
- Chapter three presents and discusses the proposed encryption approach as well as a unique method of key selection for the encryption of the voice packets. It also illustrates the time estimation for these algorithms as well as the time required to encrypt the data using AES_Rijndael algorithm for different situations.
- Chapter four presents and discusses how the proposed technique is used to mix the key information with the voice data at the sender side and split them at the receiver side. It also presents time estimation for the algorithms used in different scenarios.
- Chapter five presents and discusses the key extraction and decryption algorithms. It also illustrates the time estimation for the algorithms as well as the time required to decrypt the data using AES_Rijndael algorithm for different situations.
- Chapter six analyzes the test results and evaluates the strength of the proposed security approach. It also includes a lot of examples that compute the voice end-to-end delay budget; these examples can be used as guidelines by the network designers.

- Chapter seven presents the VoIP prototype built to test the proposed solution. It presents the system architecture, database design, algorithms, and interface.
- Chapter eight provides a final conclusion and suggested future work.

Review of related literature

2.1. Introduction

Telecommunications worldwide has experienced a significant revolution over recent years. The rapid convergence of data, voice, and video using IP-based network is delivering advanced services at lower cost across the spectrum, including residential users, business customers of varying sizes, and service providers. Voice over Internet Protocol (VoIP) technology is the focus of this research, it is considered one of the key technologies that is rapidly driving this convergence [Sherburne & FitzGerald, 2004].

VoIP technology has many advantages over the traditional Public Switched Telephone Networks (PSTN); decreasing cost and increasing revenue are the main motivation towards VoIP. Integration with other media services, network bandwidth, and service portability are other factors for the deployment of this technology [Ahuja & Ensor, 2004]. On the other hand, VoIP has a lot of disadvantages, issues, and challenges; the most important among them are: security, performance, and Quality of Service (QoS) [Collier] [Uday & Pabrai, 2004]. Many measures are used to judge QoS of voice applications; among them are: Mean Opinion Score (MOS) and E-Model.

VoIP system suffers from delay introduced by many sources, summarized by: Analog-to-Digital (A/D) Conversion, compression, packetization, transmission, decompression, and Digital-to-Analog (D/A) Conversion; Encryption, decryption, and keys management result in a significant amount of time added to the voice delay; therefore, new techniques should be developed and applied to reduce the overhead introduced from these processes in order to keep the one-way end-to-end delay below the recommended value (150 ms) [SUNYIT, 2003] [ITU-T G.114] [Balliache, 2003]. VoIP uses the User Datagram Protocol (UDP) rather than Transmission Control Protocol (TCP) together with the Internet Protocol (IP); this is because the retransmission of lost packet(s) in the TCP may degrade the QoS in this type of real time applications. Voice application does not need every single packet, but it needs a continuous flow of packets in the correct order. Figure 0-1 [SUNYIT, 2003] shows the effect of packet loss and delay on the QoS. It is clear that 5% packets loss does not affect the QoS; and that the QoS is still good even with 10% packets loss. However missing information degrades the output signal, a Packet Loss Concealment (PLC) algorithm may be used to smooth over the gaps. [Nortel Networks 2001]

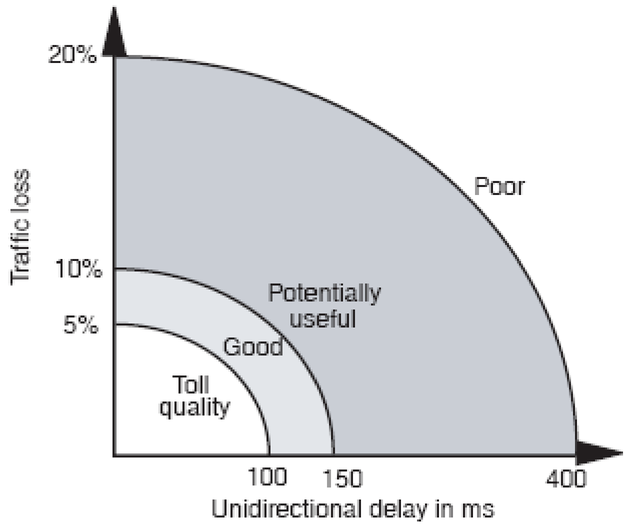


Figure 0-1: Traffic loss versus traffic delay.

Deployment of VoIP application needs mainly the following components: IP-phones/Softphones[1], IP network, voice server including IP-enabled Private Branch eXchange (PBX), media gateway, and trunks [Collier]; PBX and gateway are needed for the interface with the PSTN. A set of protocols are used such as: Session Initiation

[1] The term softphone refers to a telephone capability implemented on an ordinary PC, using only special software and a microphone/headset that plugs into the PC's audio ports [Kuhn et al., 2005].

Protocol (SIP), Real Time Protocol (RTP), UDP, and IP. Each protocol adds its own header to the transmitted packet; RTP, UDP, IP, and Ethernet headers[2] resulting in 58 bytes overhead on each voice packet. In this research three extra bytes will be added to the voice packet for the purpose of key exchange; this number of bytes is added to the packet instead of the long key itself which helps the receiver in extracting the key. As a result, VoIP packet size should be chosen carefully to guarantee end-to-end delay below a minimum threshold (150 ms); at the same time QoS requirements should be achieved; large packet size reduces the ratio of the headers of the packet to its data, but at the same time, the loss of a large packet disrupts the voice quality.

Security is an essential issue in voice applications. PSTN focuses on physical security processes while VoIP depends on cryptographic systems where the signal is digitized. Most of the current encryption/decryption techniques were built for text data; therefore they are not suitable to be used with VoIP systems since they introduce an unacceptable delay due to their extensive computations. Keys distribution through a third-party forms another source of delay; furthermore a single key in this case is used during the whole session which is harmful, where it is enough for a hacker to get the key of one packet to identify the whole data of the session. In this research a new key is used with every new packet; Information about the key is mixed with the voice data at the sender side and will be used at the receiver side to extract the key.

[2] RTP: 12 bytes, UDP: 8 bytes, IP: 20 bytes, Ethernet:18 bytes. [Newport networks, 2005] [Downey, 2005]

Coder/decoder (codec) is used to digitize voice signal; there exist a lot of codecs[3] and the selection among them is not an easy job since each of them has its own data rate, packetization delay, QoS, and bandwidth requirements; therefore, there is no single choice that is suitable for all VoIP deployments; in this research a set of recommendations are introduced in the coming chapters for different network configurations.

2.2. Public Switched Telephone Networks (PSTN)

The first voice transmission through ring-down circuit was invented by Alexander Graham Bell in 1876. It was one-way voice transmission; converged connection between every two parties was required. After that, a centralized operator system (i.e. human switch) was employed; followed by an automatic switch. Figure 0-2 shows these installations. Packet switching networks formed a significant convergence in telephone technology. Nowadays, VoIP started to replace the PSTN [Davidson & Peters, 2004, PP:5-7].

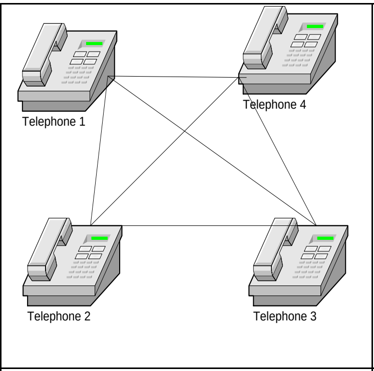
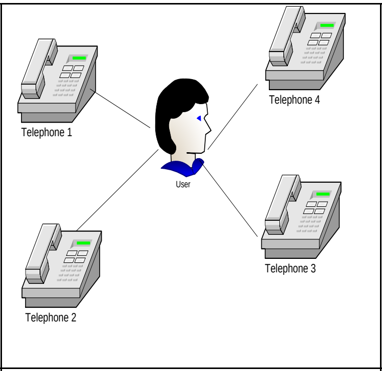
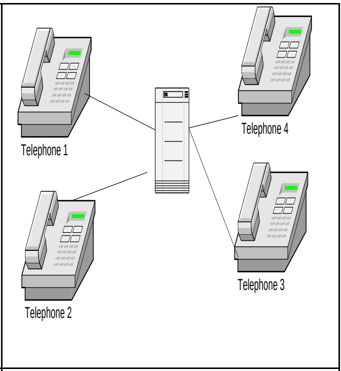
		
a- A link between every two parties	b- Centralized operator (human switch)	c- Centralized operator (automatic switch)

Figure 0-2: Voice transmission convergence

[3] Details of codecs exist in the coming sections.

"Although the PSTN is effective and does a good job for the purpose it was built to do (i.e., switch voice calls), many business drivers are striving to change it to a new network, whereby voice is an application on top of a data network. This is now taking place for several reasons:" [Davidson & Peters, 2004, PP:19-20]

- Data has overtaken voice as the primary traffic on many networks built for voice.
- The PSTN cannot create and deploy features quickly enough.
- Data/Voice/Video cannot converge on the PSTN as currently built.
- The architecture built for voice is not flexible enough to carry data.

2.3. Voice over Internet Protocol (VoIP)

The flow of voice data over packet network is performed as follows: The analog signal from a telephone/microphone connected to PC is digitized into Pulse Code Modulation (PCM) samples using voice coder-decoder (codec) as shown in Figure 0-3; some of the well known codecs are illustrated in Table 0-5. The PCM samples are then passed to the compression algorithm which compresses the voice into packet format for transmission across the network (WAN/Internet). On the other side the same functions are performed in reverse order. This is usually called a (PC-to-PC architecture) which is connected to LAN or via telephone line to Internet Service Provider (ISP) as shown in Figure 0-4; in this case sampling, compression/decompression, encryption/decryption, and packetization of the voice is made in the codec hardware/software of the PC; IP-telephones with computational capabilities can substitute the PCs.

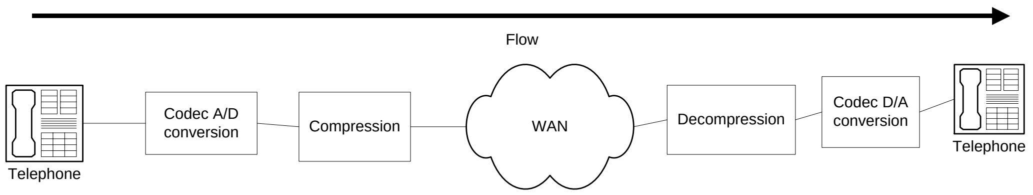


Figure 0-3 end-to-end voice flow

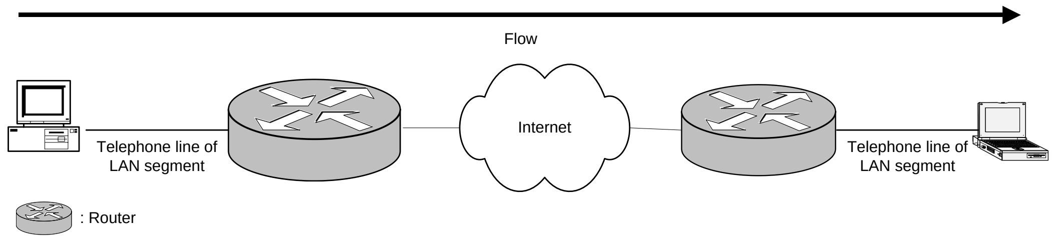


Figure 0-4 end-to-end voice flow (PC-to-PC architecture)

2.3.1. VoIP motivation and benefits

VoIP technology has many advantages over the traditional PSTN, decreasing cost and increasing revenue are the main motivation towards VoIP. Integration with other media services, network bandwidth, and service portability are other factors for deployment of this technology [Ahuja & Ensor, 2004]. "While VoIP represents about 1 percent of current enterprise voice, it is gradually replacing traditional voice system implementations". [Collier]

The advantages and benefits of VoIP can be summarized as follows: [Ahuja & Ensor, 2004] [Davidson & Peters, 2004, PP:129-131] [Kuhn et al., 2005]

1. Less cost per minute compared to PSTN. Per minute cost saving differs from country to another based on the geographical location.
2. One infrastructure network for both data and voice. This adds extra saving.
3. The integration with other media services.
4. The flexibility in adopting new services such as secure calls.

5. Voice compression and silence suppression result in increasing bandwidth.
6. Portable telephone through using Dynamic Host Configuration Protocol (DHCP). In the case of using soft-phone a user can login to any computer with the application installed on using his/her user ID and password, and then do connect to others.
7. Improved network utilization; the traditional circuit switched networks have to dedicate a full-duplex 64 Kbps channel for the duration of a single call, whereas with VoIP networks the bandwidth is used only when something has to be transmitted. [Zeadally et al, 2004]

2.3.2. VoIP drawbacks and challenges

VoIP has a lot of advantages as mentioned above but at the same time it has a lot of issues and challenges [Uday & Pabrai, 2004] [Kuhn et al., 2005]; the most important among them are: performance, Quality of Service (QoS), and security.

Voice applications suffers from the delay where it should achieve the International Telecommunication Union (ITU) recommendations of 150 ms end-to-end delay; Processing, serialization, queuing, propagation, and network forms the main delay/latency sources of the voice packets traveling over IP networks; an in depth analysis of those delay sources exists in chapter eight of this dissertation. Delay jitter is another source that affects this kind of applications; this delay should not exceed 20 ms with combination of packets loss and 50 ms without packet loss. Packet loss in a packet switching networks is expected; in real time voice applications it is not recommended to retransmit the lost packet(s) where voice is tolerance for packet loss in the range (2 – 5) % of the transmitted packets especially when the loss happens for individual but not in bulk packets. "Studies reveal that losing traffic that is around 32-64 ms (for G.711 traffic) in duration is disruptive, because it means the loss of speech phonemes. On the other hand, cell loss of duration of some 4-16 ms is not noticeable nor disturbing to the listener."[SUNYIT, 2003]

There exist standard measures for QoS in voice application, some of these are: Mean Opinion Score (MOS) and E-Model. MOS gives a grade from 1 (poor) to 5 (excellent), while E-model gives a factor R from 0 (poor) to 100 (excellent). Table 0-1 [Walker, 2001] [Pietrosemoli, 2004] shows the measures of both techniques against user satisfaction. Network bandwidth, congestion, and queuing are main factors that affect the QoS; increasing the network bandwidth and employing a strong queuing method with high priority for voice packets definitely improves the QoS. Avoiding large data packets on the voice trunk minimizes the queuing delays and therefore enhances the QoS.

Table 0-1 Levels of MOS and E-Model measures

USER SATISFACTION	**MOS(Mean Opinion Score)**	**R factor**
Very satisfied	4.3 – 4.4	90 – 100
Satisfied	4.0 – 4.3	80 – 90
Some users dissatisfied	3.6 – 4.0	70 – 80
Many users dissatisfied	3.1 – 3.6	60 – 70
Nearly all users dissatisfied	2.6 – 3.1	50 – 60
Not recommended	1.0 – 2.6	0 – 50

As mentioned in the introduction of this chapter, security is a big challenge for VoIP system developers. It is necessary to seek a new encryption/decryption algorithm to achieve the requirements of the real-time applications –especially the end-to-end delay- such as VoIP; also new methods of key generation, distribution, and management are needed for this type of applications where the users are growing quickly and distributed around the globe.

2.3.3. VoIP security

Although wiretapping is a concern in PSTN, majority of users considered the security provided by PSTN is reasonable and sufficient. An intruder should be physically connected to the telephone line with a device in order to eavesdrop on a conversation. In VoIP systems, the threats are increased since the network/Internet is used by the public, and therefore any conversation over the IP can be compromised by any user who has an access to the network. Sufficient tools and equipment to sniff on VoIP traffic and play it back with/without being noticed are available to the hackers and intruders. As a result, VoIP security is very important for the users to convey sensitive conversations over IP. Known encryption algorithms are used successfully to secure data traffic over IP, but they are computationally expensive; hence when VoIP designers plan to employ these encryption algorithms, they should keep in mind their effect on the QoS and performance since VoIP already suffers from latency.

2.3.4. VoIP related protocols and standards

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the transport layers in the TCP/IP architecture. TCP is a connection oriented while UDP is a connectionless. Real-time Transport Protocol (RTP) is a protocol for real-time applications such as VoIP. Session Initiation Protocol (SIP) is a standard for VoIP deployment. H.323 is a standard for videoconferencing so it is beneficial for VoIP application. Finally, IPSec is a protocol for IP communication security. Following is a brief description of those protocols.

2.3.4.1. Ethernet

Ethernet is the most commonly used LAN protocol; the original DIX (DEC, Intel, Xerox) frame structure is shown in Figure 0-5 (a). When the Ethernet standardized by IEEE as 802.3, the committee made two changes to DIX format as shown in Figure 0-5 (b). The first one was to reduce the preamble to 7 bytes and use the last byte for a Start of Frame Delimiter (SFD), and the second was to change the type field into a length field. Fortunately any of them can be used; if the number in the type field is greater than 1500, then it is a type otherwise it is a length. Varying Ethernet configurations exist, but the detail of them is out of the scope of this research. [Tanenbaum, 2003, PP:275-278] [Stallings, 2004, PP:470-480]

8	6	6	2	0-1500	0-46	4	Bytes
Preamble	Destination address	Source address	Type	Data	Pad	Checksum	

(a)

7	1	6	6	2	0-1500	0-46	4	Bytes
Preamble	S F D	Destination address	Source address	Length	Data	Pad	FCS	

(b)

Figure 0-5: Frame formats. (a) DIX Ethernet. (b) IEEE 802.3

Where;

Preamble: each byte contains the bit pattern (10101010); it is used for synchronization purposes between sender and receiver.

Start of Frame Delimiter (SFD): The sequence 10101011 which indicates the actual start of the frame.

Destination Address (DA): Specifies the station(s) for which the frame is intended. It may be a unique physical address, a group address, or a global address.

Source Address (SA): Specifies the station that sent the frame.

Type/Length: Tells the receiver what to do with the frame in the case of DIX, and the length of the data in the case of IEEE 802.3

Data: A maximum of 1500 bytes of data.

Pad: If the data size is less than 46 bytes, the pad field is used to fill out the frame to its minimum size.

Frame Check Sequence (FCS)/ Checksum: A 32-bits hash code of the data used by the receiver to validate the received data.

2.3.4.2. Transmission Control Protocol (TCP)

For most applications running as part of the TCP/IP architecture, the transport layer is TCP. TCP provides a reliable data transmission between applications where a logical connection temporarily associates the two entities in different systems. The retransmission mechanism is used here until a lost packet arrives to the receiver side. TCP is not suitable to be used with VoIP because retransmission of any lost packets results in delay which may degrade the voice quality. The minimum TCP header consists of twenty octets[4] as shown in Figure 0-6 where HL is the Header Length. [Tanenbaum, 2003, PP:532-537] [Stallings, 2004, PP:228-231]

[4] An octet is 8 bits

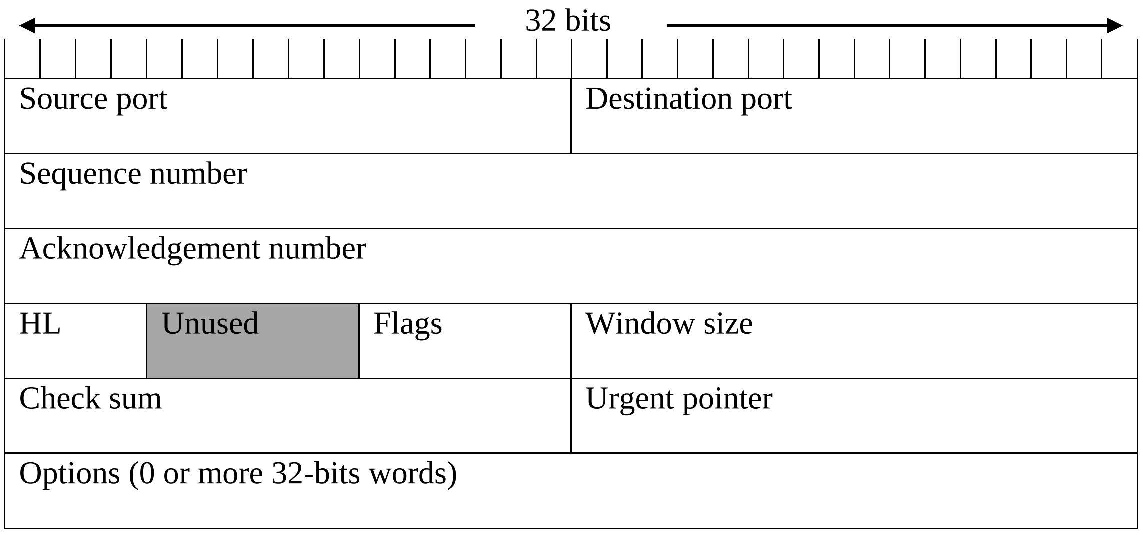


Figure 0-6 TCP header

2.3.4.3. User Datagram Protocol (UDP)

The other transport layer in the TCP/IP architecture is the UDP. UDP is a connectionless oriented protocol where different packets may transmit through different paths; at the other end packets are re-sequenced. UDP is unreliable protocol (packet delivery is not guaranteed) where there is no retransmission for lost packet(s). As voice packets are of small size (usually, a data of not more than 30 ms period of time), VoIP is tolerance to packet loss of 2 – 5%; therefore UDP is a suitable choice to be used with VoIP systems. The UDP header consists of eight octets as shown in Figure 0-7. [Tanenbaum, 2003, PP:524-525] [Stallings, 2004, PP:234-235]

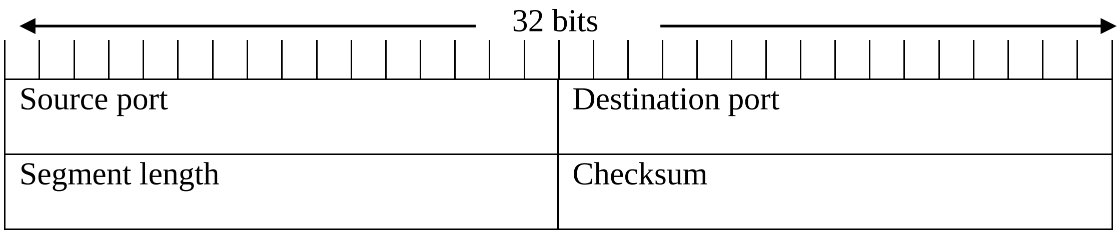


Figure 0-7 UDP header

2.3.4.4. Internet Protocol (IP)

The Internet Protocol (IP) is part of the TCP/IP suite and is the most widely used internetworking protocol. There are two versions of this protocol: version 4 (IPv4) and version 6 (IPv6); IPv6 will ultimately replace IPv4, but IPv4 is widely used currently. The IPv4 header is shown in Figure 0-8. [Tanenbaum, 2003, PP:433-436] [Stallings, 2004, PP:281-284].

IHL: Internet Header Length.

TOS: Type of Service

DS: Differentiated Services.

ECN: Explicit Congestion Notification.

TOS: Type of Service

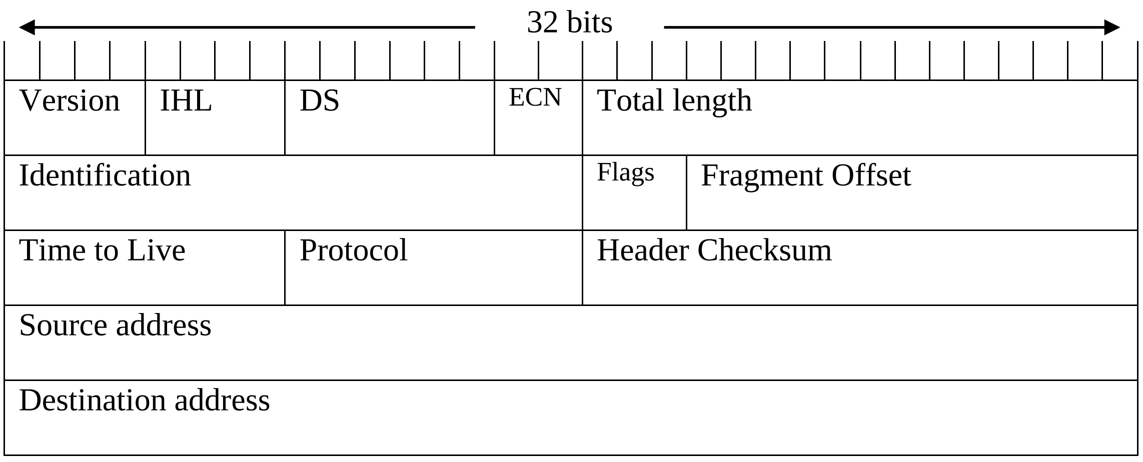


Figure 0-8: Standard IPv4 header

2.3.4.5. Real-time Transport Protocol (RTP)

It was mentioned that VoIP doesn't use TCP because it is too heavy for real time applications, so instead a UDP is used. UDP has no control over the order in which packets arrive at the destination or how long it takes them to get there (datagram concept). Both of these are very important to overall voice quality. RTP solves the problem enabling the receiver to put the packets back into the correct order and not wait too long for packets that have either lost their way or are taking too long to arrive (we don't need every single voice packet, but we need a continuous flow of many of them and ordered). [Arcomano, 2002]. RTP header components are shown in Figure 0-9. [Keagy, 2000, PP:319-325]

V:	Indicates the version of the RTP used.
P:	Indicates presence/absence of padding bytes for fixed block-size encryption algorithms.
X:	Indicates the presence of header extension follow the fixed 12-bytes header.
CC:	CSRC Count: indicates the number of Contributing Source (CSRC) header fields that follow the fixed 12-bytes header.
M:	A marker bit which identifies the beginning of a "speech burst" for VoIP applications.
PT:	Payload Type which identifies the encoded media (i.e. different audio codecs for VoIP).

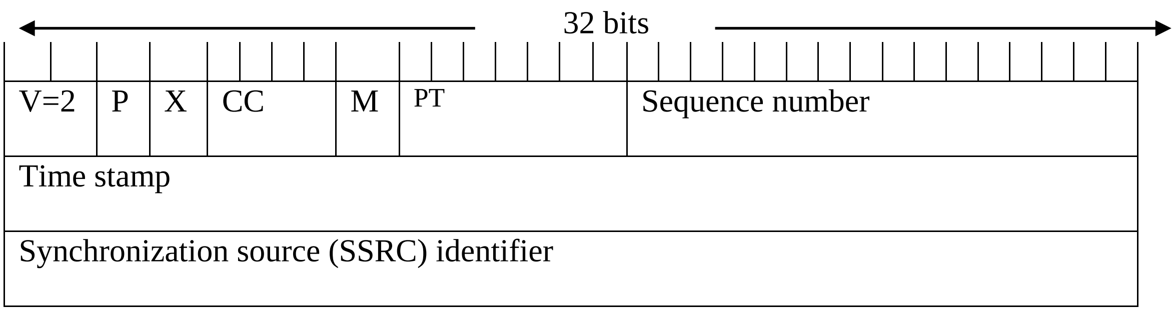


Figure 0-9 RTP header

2.3.4.6. H.323

It is an ITU-T specification for transmitting audio, video, and data across an IP network. H.323 standard addresses call signaling and control, multimedia transport and control, and bandwidth control for point-to-point and multipoint conferences. The H.323 standard consists of the components and protocols shown in Table 0-2.

Table 0-2 H.323 components and protocols

Feature	Protocol
Call signaling	H.225
Media control	H.245

Audio codecs	G.711, G.722, G.723, G.728, G.729
Video codecs	H.261, H.263
Data sharing	T.120
Media transport	RTP/RTCP

The H.323 protocols are supported by both reliable and unreliable packet delivery mechanisms over data networks. "Although most H.323 implementations today utilize TCP as the transport mechanism for signaling, H.323 version 2 does enable basic UDP transport." [Davidson & Peters, 2004, P.234]. Figure 0-10 [Davidson & Peters, 2004, P.234] illustrates the layers of the H.323 protocol suite.

Reliable TCP Delivery		Unreliable UDP Delivery		
H.245	H.225		Audio/Video Streams	
	Call control	RAS	RTCP	RTP
TCP		UDP		
IP				
Data/Physical layer				

Figure 0-10 Layers of the H.323 Protocol Suite

2.3.4.7. Session Initiation Protocol (SIP)

"The Session Initiation Protocol (SIP), defined in RFC 3261, is an application-level control protocol for setting up, modifying, and terminating real-time sessions between participants over an IP data network. The key driving force behind SIP is to enable Internet telephony, also referred to as voice over IP (VoIP). SIP can support any type of single media or multimedia session, including teleconferencing." [Stallings, 2004, P 137] SIP supports unicast and multicast sessions as well as point-to-point and multipoint calls. SIP is a text-based protocol that is part of the overall Internet Engineering Task Force (IETF) multimedia architecture. Although IETF includes other protocols, SIP's functions are independent (i.e. it does not depend on any of these protocols). [Davidson & Peters, 2004, P.251]

Calling and called parties are identified by SIP addresses; SIP address, also called SIP Universal Resource Locator (URL), exists in the form *user@host* similar to e-mail address; the user portion can be a user name or telephone number and the host portion can be a domain name or network address; examples are: *sip:abushqeer@aaugs.edu.jo* and *sip:02637444@176.16.0.1*

2.3.4.8.IPSec

The Internet Architecture Board (IAB) included the authentication and encryption as necessary security features in the IPv6 generation. Fortunately, these security capabilities were designed to be used with both IPv4 and IPv6. Therefore, the vendors do not need to wait for IPv6 to offer these features but they can offer them with the current IP generation (i.e. IPv4). IPSec provides the capability to secure communications across LAN, private and public WANs, and internet. IPSec encrypts and/or authenticates all traffics at the IP level; therefore it can support a variety of applications. A typical scenario of IPSec is shown in Figure 0-11 [Stallings, 2003, P.484]. In this figure, an organization maintains LANs at dispersed locations; nonsecure IP traffic is conducted on each LAN; for traffic offsite, through WAN, IPSec protocols are used; These protocols operate in networking devices such as routers or firewalls that connect each LAN to the outside world. IPSec networking device encrypts/decrypts and compresses/decompresses the traffic going/coming into/from the WAN; these operations are transparent to workstations and servers on the LAN.

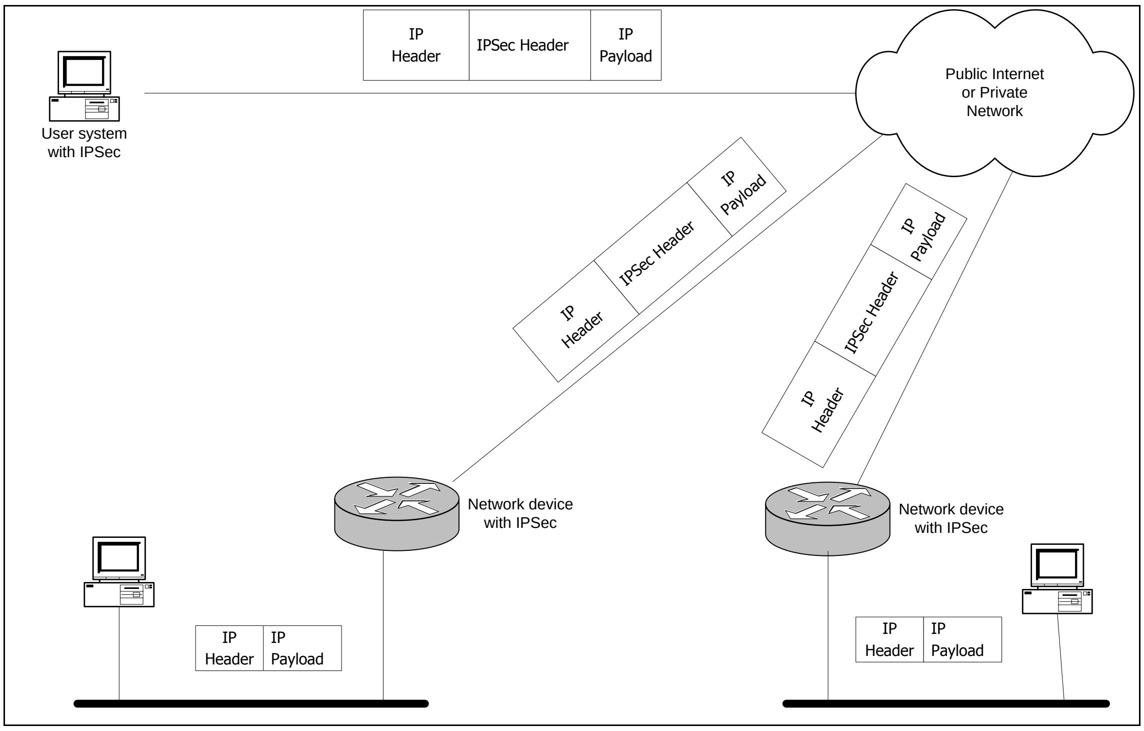


Figure 0-11 Typical IPSec Scenario

2.3.5. Voice quality

The ITU-T recommendations G.114 [ITU-T G.114] noted that if the one-way end-to-end delay is kept below 150 ms, then most applications would not be significantly affected. Furthermore, 400 ms is set to be an upper limit for any network planning. Figure 0-1 [SUNYIT, 2003] shows the quality of service using two parameters: end-to-end delay and packet loss.

Mean Opinion Score (MOS) and E-model are two techniques used to measure the voice quality. MOS gives a grade from 1 (poor) to 5 (excellent), while E-model gives a factor R from 0 (poor) to 100 (excellent). Table 0-3 [Walker, 2001] [Pietrosemoli, 2004] shows the measures of both techniques against user satisfaction, while Table 0-4 [Walker, 2001] [Pietrosemoli, 2004] [Boger] [Angus, 2001] shows MOS and the

amount subtracted from the R factor for some known codecs. It is clear that there is an inverse relationship between codec data rate and voice quality; this relationship is shown in the column chart of Figure 0-12. This issue should be kept in mind whenever a codec is selected to be used in such application.

Table 0-3 Levels of MOS and E-Model measures

USER SATISFACTION	MOS(Mean Opinion Score)	R factor
Very satisfied	4.3 – 4.4	90 – 100
Satisfied	4.0 – 4.3	80 – 90
Some users dissatisfied	3.6 – 4.0	70 – 80
Many users dissatisfied	3.1 – 3.6	60 – 70
Nearly all users dissatisfied	2.6 – 3.1	50 – 60
Not recommended	1.0 – 2.6	0 - 50

Table 0-4 Voice codecs against MOS, R factor, required CPU, and delay

Vocoder			**Amount subtracted from the R factor. (see** Error! Reference source not found.**)**	**MOS**	**Required CPU resources**	**Added delay**
Codec	**Reference**	**Data Rate (kbps)**				
PCM	G.711	64	0	4.4	Non Required	N/A
ADPCM	G.726	32	11	4.2	Low	Very low
CS-ACELP	G.729AB	8	11	4.2	High	Low
MP-MLQ	G.723.1m	6.3	15	3.98	Moderate	High
ACELP	G.723.1a	5.3	19	3.5	Moderate	High

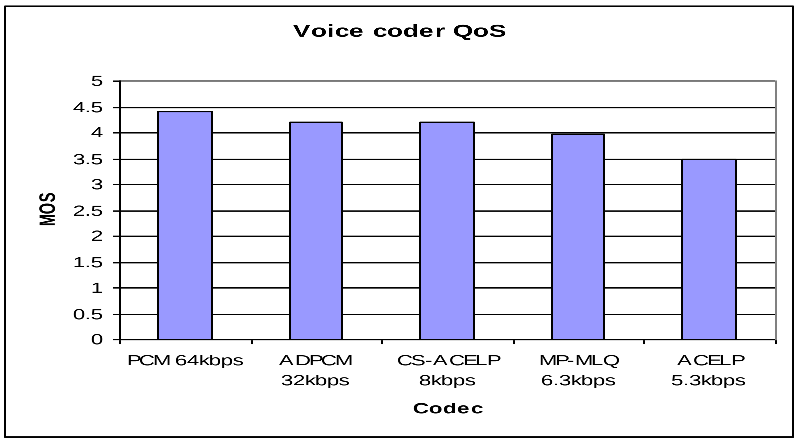


Figure 0-12 MOS for variety codecs

2.3.6. VoIP Delay/Latency

"VoIP delay or latency is characterized as the amount of time taken for speech to exit the speaker's mouth and reach the listener's ear" [Davidson & Peters, 2004, P.167]. Many types of delay are inherent in today's telephony networks. Some of them are fixed while others are variables. This section discusses these delays in details. Figure 0-13 shows most of the delay sources:

- Processing/handling delay:

  - coder delay (capturing (per frame), look ahead, packetization, and compression);

  - decoder (decompression and play back);

  - other Digital Signal Processing (DSP) features including the security process delay.

- Serialization delay.
- Queuing delay.
- Propagation delay.
- Network delay.

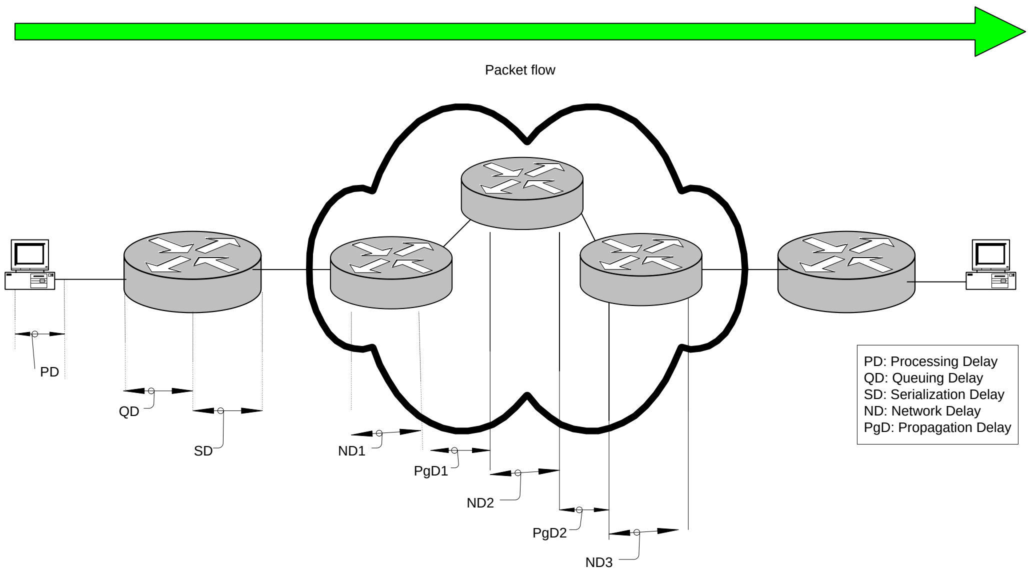


Figure 0-13 Delay sources of a voice packet

2.3.6.1. Processing/handling delay

Processing delay includes the time needed for encoding and decoding the speech, collecting the voice data into packets, and other Digital Signal Processing (DSP) features such as echo cancellation, noise reduction, Packet Loss Concealment (PLC), encryption, and decryption; these functions may be indispensable for achieving acceptable voice quality, but their contribution to delay must be taken into account. [Nortel networks, 2001]

2.3.6.1.1. Coder delays

Speech coders operate on a collection of speech samples known as frames. Each block of input speech samples is processed into a compressed frame. The coded speech frame is not generated until all speech samples in the input block have been collected by the encoder.

Thus, there is a delay of one frame before processing can begin. In addition many coders also look into the succeeding frame to improve compression efficiency. The length of this advance look is known as the ***look-ahead*** time of the coder. [ITU-T G.114] [Newport Networks, 2005]

Packetization is the time taken to fill a packet of encoded/compressed speech. It can be also called accumulation delay, as the voice samples accumulate in a buffer before they are released. Since each voice sample experiences look-ahead and packetization delays, then these processes are overlap and there is a significant benefit of this pipelining. [ITU-T G.114] [Newport Networks, 2005]

The time required to process an input frame is assumed to be the same as the frame length since efficient use of processor resources will be accomplished when an encoder/decoder pair fully uses the available processing power. Thus, the delay through codec is normally assumed to be as given in the following equation [ITU-T G.114].

$$D = 2 \times FS + LA \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \textbf{Equation 0.1}$$

Where;

D: Codec delay

FS: Frame Size

LA: Look Ahead time

Delay in IP environment (one frame per packet)

If the output facility is an IP network, then the frame output by the encoder will instantaneously be dropped into an IP packet. The additional delay required for IP packet assembly and presentation to the underlying link layer will depend on the link layer. When the link layer is a LAN (e.g. Ethernet) this additional time will usually be quite small. Thus, the minimum delay attributable to codec-related processing in IP-based systems is given in the following equation [ITU-T G.114].

$mD = 2 \times FS + LA$ ………………………………………….. **Equation 0.2**

Where;

mD: Minimum codec Delay in IP environment with single frame per packet.

When the link layer is one with lower clock rate (e.g. Modem connection) or one with high traffic load, the additional delay will increase substantially. In order to clock compressed frames at least with the same rate to the facility as the speech samples are collected at the input of the encoder, the additional delay should not exceed one frame size. Thus, the maximum delay attributable to codec-related processing in IP-based systems operating in real-time is given in the following equation [ITU-T G.114].

$MD = 3 \times FS + LA$ ……………………………………… **Equation 0.3**

Where;

MD: Maximum codec Delay in IP environment with single frame per packet.

Delay in IP environment (multiple frames per packet)

If multiple voice frames are grouped together into a single IP packet, further delay is added to the speech signal. This delay will be at least the duration of one extra voice frame at the encoder for each additional voice frame added to the IP packet. Thus, the minimum delay attributable to codec-related processing in IP-based systems with multiple frames per packet is given in the following equation [ITU-T G.114].

$mD = (N + 1) \times FS + LA$ ……………………………………. **Equation 0.4**

Where;

mD: Minimum codec Delay in IP environment with multiple frames per packet.

N: number of frames in each packet.

When the link layer is one with lower clock rate (e.g. Modem connection) or one with high traffic load, additional delay will be incurred in delivering the packet to the facility. In order to clock compressed frames at least with the same rate to the facility as the speech samples are collected at the input of the encoder, the additional delay should, in case of multiple frames per packet, not exceed the length of the frames contained in one packet. It should be noted that clocking out a packet to the IP facility cannot start before all speech frames for this packet are available. Thus, the maximum delay attributable to codec-related processing in IP-based systems operating in real-time with multiple frames per packet is given in the following equation [ITU-T G.114].

$MD = (2N + 1) \times FS + LA$ …………………………………….. **Equation 0.5**

Where;

MD: Maximum codec Delay in IP environment with multiple frames per packet.

N: number of frames in each packet.

Table 0-5 [ITU-T G.114] illustrates the characteristics of many known coders. The values of the last four columns in the table were computed using Equations 2.2, 2.3, 2.4, and 2.5 respectively. The relationship between the coder data rate and its delay is an inverse relationship as shown in Figure 0-14.

Table 0-5 Coders characteristics

Vocoder		Data rate (kbps)	Frame size (ms)	Frame size (bits)	Look-ahead (ms)	Mean one-way delay introduced by coder-related processing (ms)				
						one frame/packet		Multiple frames of 30 ms per packet		
Coder	Reference					Min.	Max.	# of frames/ packet	Min.	Max.
PCM	G.711	64	0.125	8	0	0.25	0.375	240	30.125	60.125
ADPCM	G.726	40	0.125	5	0	0.25	0.375	240	30.125	60.125
ADPCM	G.726	32	0.125	4	0	0.25	0.375	240	30.125	60.125
ADPCM	G.726	24	0.125	3	0	0.25	0.375	240	30.125	60.125
ADPCM	G.726	16	0.125	2	0	0.25	0.375	240	30.125	60.125
CS-ACELP	G.729 AB	8	10	80	5	25	35	3	45	75
MP-MLQ	G.723.1m	6.3	30	189	7.5	67.5	97.5	1	67.5	97.5
ACELP	G.723.1a	5.3	30	159	7.5	67.5	97.5	1	67.5	97.5

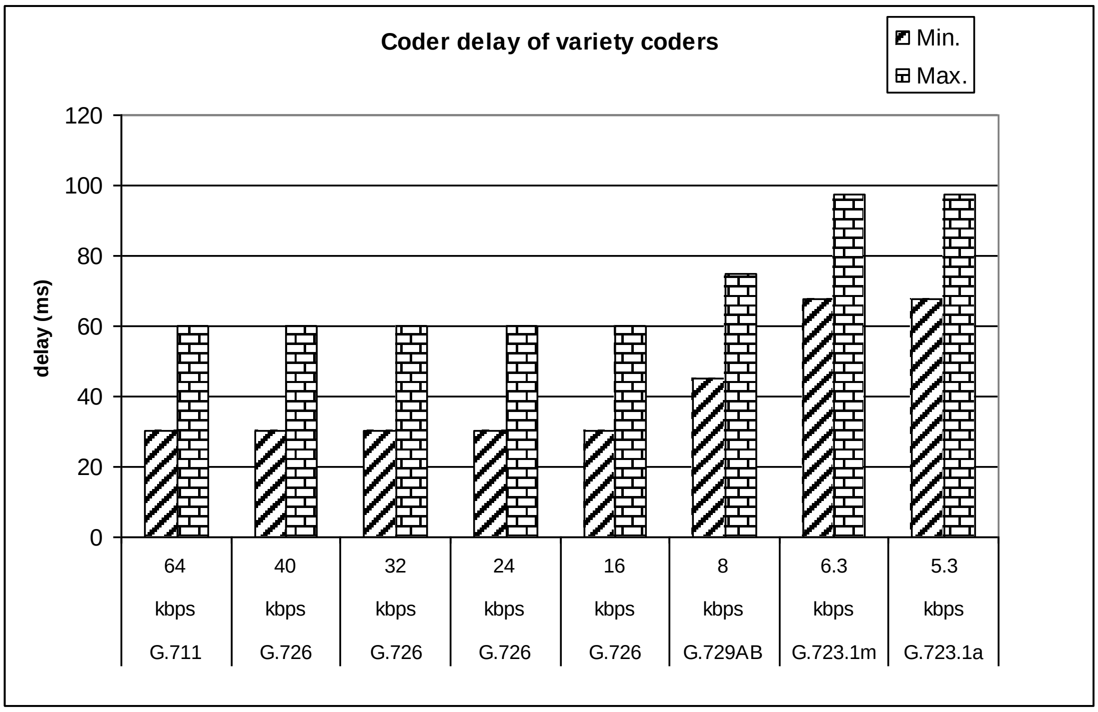


Figure 0-14 Coder delay for variety codecs

2.3.6.1.2. Decoder delay

When the voice packet arrives to its destination, decompression and decoding processes take place. Decode delays can be assumed to be half of the encode delays [Kostas et al., 1998].

2.3.6.1.3. Security process delay

Nothing can be achieved without a penalty. Delay time of encryption/decryption processes is the penalty a system should pay in order to secure the voice data from hackers and intruders. In this research, the delay time was kept as possible as minimum; details exist in the coming chapters.

2.3.6.2. Serialization delay

Serialization delay is the fixed delay required to clock a voice or data frame onto the network interface. It is directly related to the frame size and clock rate on the trunk [Cisco, 2006]. Therefore, we can determine the maximum serialization delay caused by a frame using the following equation:

$$SD = FS / ICR \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \textbf{Equation 0.6}$$

Where;

SD:	Serialization delay
FS:	Frame Size
ICR:	Interface Clocking Rate.

Furthermore, the serialization delay is incurred whenever the packet passes through another store-and-forward device such as a router or a switch [Angus, 2001]. Table 0-6 shows the serialization delay of 30 ms voice packet at different link speeds for a variety of codecs. For example, a G.711 (30 ms voice packet) plus its header (totaling 2408 bits) has 4.7 ms serialization delay at 512 kbps trunk. It is clear from the table that the serialization delay is negligible for all codecs at high speed lines; while it is significant at low speed lines even with the codecs of high compression rates. For example, a G.711 voice packet consumed 43 ms at 56 kbps line (i.e. 29% of the recommended 150 ms end-to-end delay). Thus, the large values in the table should be avoided in the voice applications. As shown in Figure 0-15 the relationship between the serialization delay and the packet size is a direct relationship, while it is an inverse relationship with the link speed.

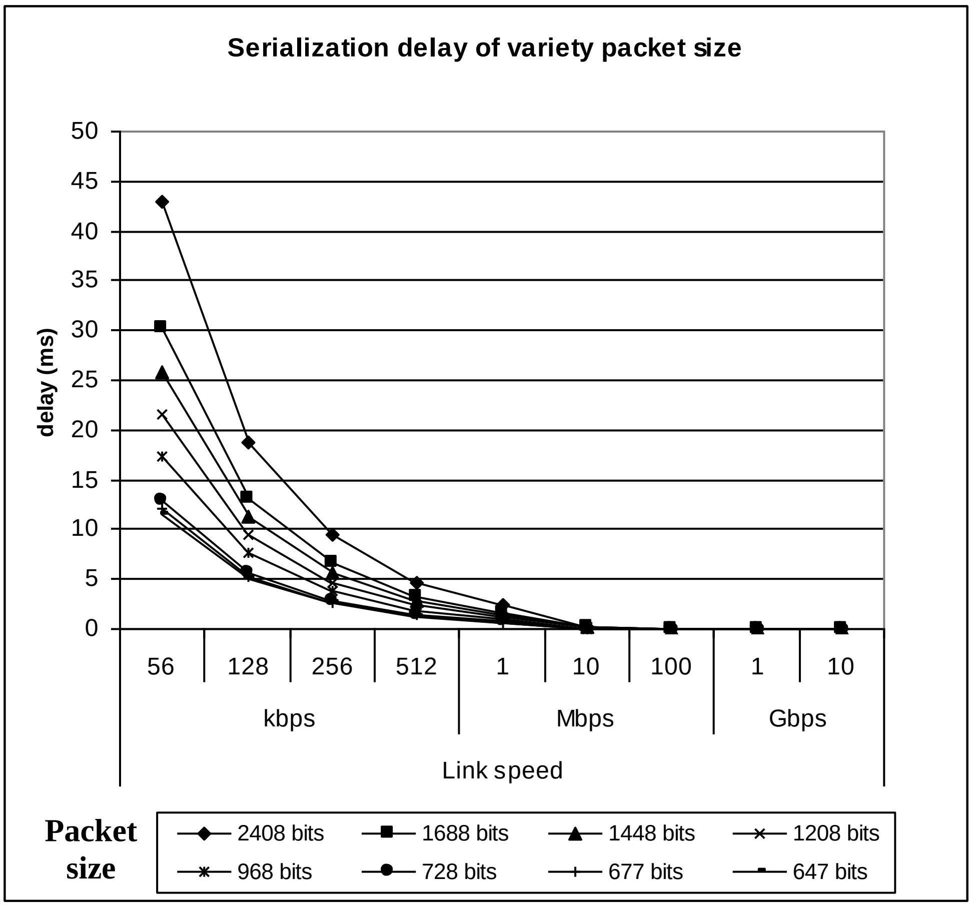


Figure 0-15 Serialization delay against packet size on variety link speeds

Table 0-6 serialization delay (ms) against link speed for variety codecs

Fixed frame – serialization delay (ms)[5]											
Vocoder		Total packet size (bits)[6]	Link speed								
			Kbps				Mbps			Gbps	
Coder	Reference		56	128	256	512	1	10	100	1	10
PCM	G.711	2408	43.0000	18.8125	9.4063	4.7031	2.4080	0.2408	0.0241	0.00241	0.00024
ADPCM	G.726	1688	30.1429	13.1875	6.5938	3.2969	1.6880	0.1688	0.0169	0.00169	0.00017
ADPCM	G.726	1448	25.8571	11.3125	5.6563	2.8281	1.4480	0.1448	0.0145	0.00145	0.00014
ADPCM	G.726	1208	21.5714	9.4375	4.7188	2.3594	1.2080	0.1208	0.0121	0.00121	0.00012
ADPCM	G.726	968	17.2857	7.5625	3.7813	1.8906	0.9680	0.0968	0.0097	0.00097	0.00010
CS-ACELP	G.729AB	728	13.0000	5.6875	2.8438	1.4219	0.7280	0.0728	0.0073	0.00073	0.00007
MP-MLQ	G.723.1m	677	12.0893	5.2891	2.6445	1.3223	0.6770	0.0677	0.0068	0.00068	0.00007
ACELP	G.723.1a	647	11.5536	5.0547	2.5273	1.2637	0.6470	0.0647	0.0065	0.00065	0.00006

2.3.6.3. Queuing delay

After the voice packet becomes ready, the headers are added and the frame is queued for transmission on the network connection; this waiting time plus the processing time at the router is known as queuing delay. Processing time is very short and processing can be carried during the waiting time, therefore from this point and forward in this research it is assumed that queuing delay is the waiting time in the router. Queuing occurs because of congestion when more frames are sent out than the interface can handle at a given

[5] Using Equation 0.6

[6] See Table 0-7

interval. This delay should be kept to a minimum value as much as possible by using the optimal queuing methods for different network configurations. Queuing delay depends on the bandwidth and utilization of the link. Figure 0-16 [Barlow] shows a diagram that provides a comparison of queuing delay values, for a specific service class based on connection bandwidth and bandwidth utilization. This figure shows a direct relationship between the bandwidth and the queuing delay as the utilization increases.

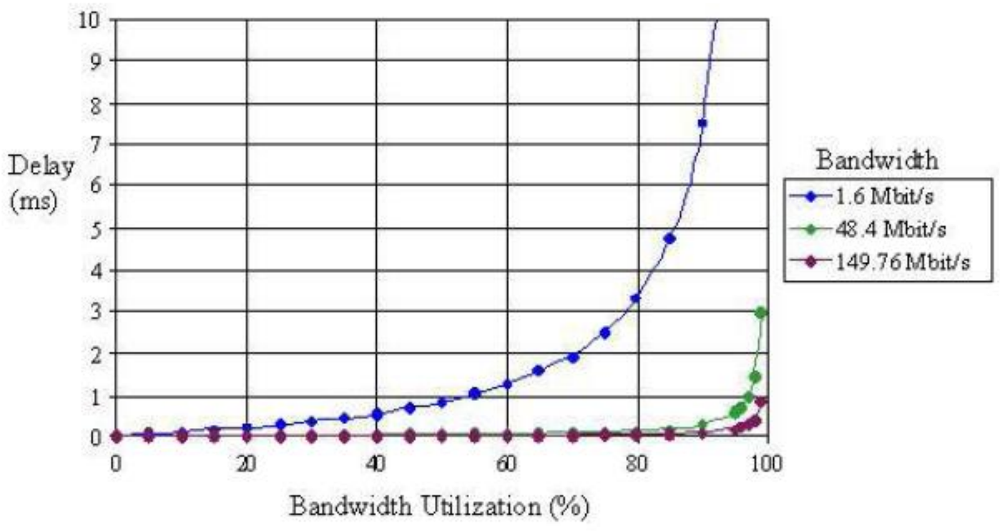


Figure 0-16 Queuing delay against bandwidth utilization on different bandwidths

Voice frames have a high priority value (101) in the Type of Service (TOS) field in the IP header; therefore a voice frame waits only for one data frame that already plays out and the preceding voice frames -if any-. Essentially, the voice frame waits for the serialization delay of any preceding frames in the output queue. Table 0-7 shows queuing delays of 30 ms voice frame for different coders at different link speeds with the assumption that one data frame of size 10240 bits (i.e. 1280 octets, the minimum Maximum Transmission Unit (MTU) that must be supported by each network), and another voice frame are preceding the frame in the queue. In this table, it was assumed that half of the data packet in average is already serialized (i.e. the computations were carried on 10240/2 = 5120 bit of data). Thus queuing delays were computed as given in the following equation.

QD = (FS + 5120) / LS ………………………………………. **Equation 0.7**

Where;

QD: Queuing Delay

FS: Frame Size

LS: Line speed

5120: Half of the data frame size (fixed in the given table)

The chart in Figure 0-17 shows a direct relationship between the packet size and the queuing delay as well as an inverse relationship between the packet size and the link speed. As it is shown in the chart, the delay is high for voice applications with low link speeds and approaches to zero with high link speeds (10 Mbps and more).

Table 0-7 Queuing delay against link speed for variety codecs

Queuing delay[7] (ms) with assumption of 5120 bits (half of 10240 bits data packet) is already play out and another voice packet precedes the current packet											
Vocoder		**Total packet size (bits)[8]**	**Link speed**								
			kbps				**Mbps**			**Gbps**	
Coder	**Reference**		**56**	**128**	**256**	**512**	**1**	**10**	**100**	**1**	**10**
PCM	G.711	2408	134.43	58.81	29.41	14.70	7.5280	0.7528	0.07528	0.00753	0.00075
ADPCM	G.726	1688	121.57	53.19	26.59	13.30	6.8080	0.6808	0.06808	0.00681	0.00068
ADPCM	G.726	1448	117.29	51.31	25.66	12.83	6.5680	0.6568	0.06568	0.00657	0.00066
ADPCM	G.726	1208	113.00	49.44	24.72	12.36	6.3280	0.6328	0.06328	0.00633	0.00063

[7] Using Equation 0.7

[8] See Table 0-7

ADPCM	G.726	968	108.71	47.56	23.78	11.89	6.0880	0.6088	0.06088	0.00609	0.00061
CS-ACELP	G.729AB	728	104.43	45.69	22.84	11.42	5.8480	0.5848	0.05848	0.00585	0.00058
MP-MLQ	G.723.1m	677	103.52	45.29	22.64	11.32	5.7970	0.5797	0.05797	0.00580	0.00058
ACELP	G.723.1a	647	102.98	45.05	22.53	11.26	5.7670	0.5767	0.05767	0.00577	0.00058

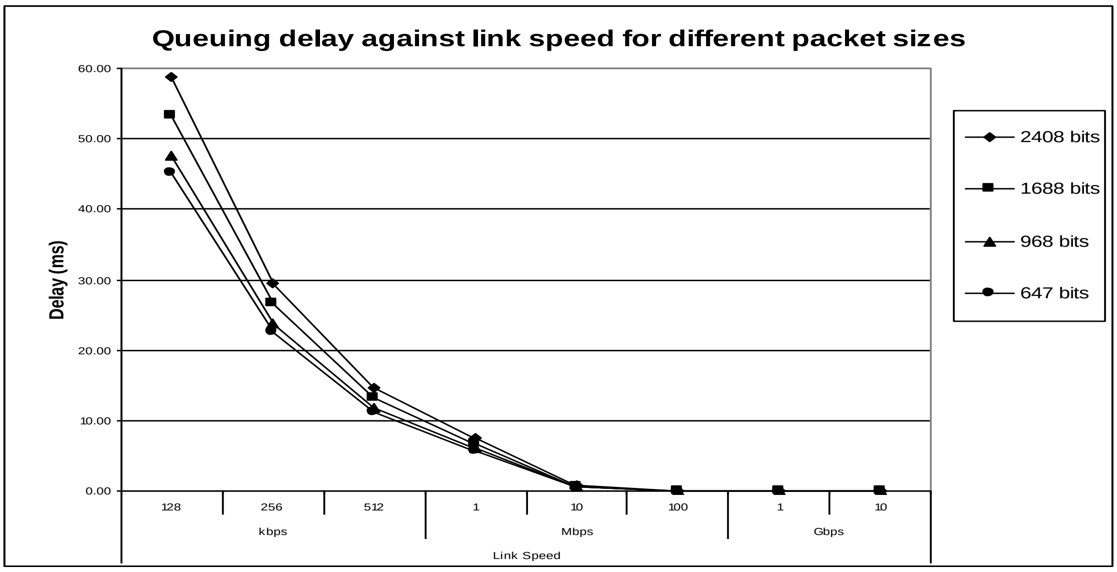


Figure 0-17 Queuing delay against packet size on variety link speeds

2.3.6.4. Propagation delay

Propagation delay is caused by the speed of light (186,000 km/sec or 125,000 miles/sec) in fiber or copper-based networks. Because the transmission medium is silica glass, light travels at a lower speed than in a vacuum. The difference in speed is accounted for by the index of refraction of the core material in the optical fiber. As a fact, propagation delay is independent from the link rate and depends on the distance.

"In order to estimate propagation delay, a popular estimate of 10 microseconds/mile or 6 microseconds/km (G.114) is widely used"[Cisco, 2006]. Depending on this estimation, Table 0-8 illustrates the estimation propagation delay for different distances. It is clear that propagation delay does not cause a significant delay on LANs or national WANs.

Although propagation delay over half of the earth's circumference (about 13000 km), is about 78 ms; it is not too big compared to the satellite applications where the journey takes more than 200 ms upward and more than 200 ms downward. Even with this latency, millions of calls take place through satellite everyday and people are satisfied; this is because there is no solution to the speed of light up to the date of doing this research. Figure 0-18 shows the direct relationship between the distance and the propagation delay.

Table 0-8 Propagation delay

Distance (Km)	Comment	Maximum propagation delay (ms)[9]
1	LAN	0.006
5		0.03
10	National networks	0.06
100		0.6
1000		6
5000	International networks	30
10000		60
13000	Half of the earth circumference	78
37000	Satellite	222

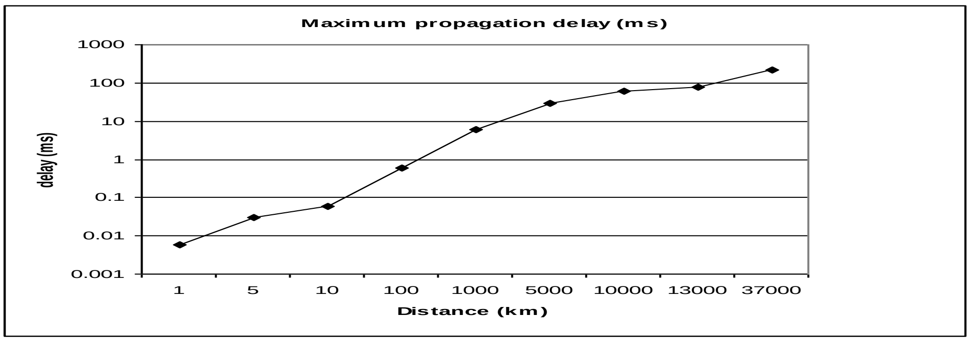


Figure 0-18 Maximum propagation delay against distance

2.3.6.5. Network delay

Queuing delays on the intermediate routers of a wide-area network is the largest delay for voice connection, and the most difficult to quantify. In private networks, it is possible to measure or estimate the queuing delays, but it is very difficult to do this in a public

[9] PgD = Distance (km) × 6 µs

network. The number of routers, network architecture, routing algorithm, queuing method, network load, line speed, and other factors cause this delay and make it difficult to quantify. The network delay in this context is the summation of the queuing and serialization delays on all the routers on the path of the packet. For simplicity in estimating the network delay of a voice packet, it is assumed that a voice packet waits the same amount of time on each router; it is also assumed that the queuing delay on each router is as it is computed in Table 0-7; therefore the network delay according to those assumptions is given by the following equation.

$$ND = NR \times (QD + SD) \qquad \textbf{Equation 0.8}$$

………………………………………….……..

Where;

ND: Network Delay

NR: Number of Routers

QD: Queuing Delay

SD: Serialization Delay

Example: Given three routers on the path of a G.711 voice packet (i.e. 2408 bits); find the network delay on a link speeds 512 kbps, 1 Mbps, 10 Mbps, and 100 Mbps.

By using Equation 0.8 and Tables 7-6 and 7-7 the SD, QD, and ND for the given problem are as shown in the next table.

Link Speed	**Serialization Delay (ms)**	**Queuing Delay (ms)**	**Network Delay (ms)**
512 kbps	4.7	14.7	58.2
1 Mbps	2.41	7.53	29.82
10 Mbps	0.241	0.753	2.982
100 Mbps	0.024	0.0753	0.2979

2.4. Cryptography

Data that can be read and understood without any special measures is called plaintext. The method of disguising plaintext in such a way as to hide its substance is called encryption. Encrypting plaintext results in unreadable ciphertext. Encryption is used to make sure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting ciphertext to its original plaintext is called decryption. [Zimmerman, 1999]

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography provides a way to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient. [Zimmerman, 1999]

Cryptographic strength is measured in the time and resources it would require to recover the plaintext. The result of strong cryptography is ciphertext that is very difficult to decipher without possession of the appropriate decoding tool. In a strong cryptographic, it is not possible to decipher the ciphertext before the end of the universe given today's computing power and available time. [Zimmerman, 1999]

While cryptography is the science of securing data, cryptanalysis is the science of analyzing and breaking secure communication. Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck. Cryptanalysts are also called attackers. [Zimmerman, 1999]

A cryptographic algorithm works in combination with a key. The same plaintext encrypts to different ciphertext with different keys. The security of encrypted data entirely depends on two things: the strength of the cryptographic algorithm and the secrecy of the key.

2.4.1. Symmetric key encryption

Symmetric encryption also referred to as conventional encryption or single-key encryption is the most widely used approach. It has five ingredients as shown in Figure 0-19 [Stallings, 2003, P.25].

- **Plaintext**: It is the original intelligible message fed into the algorithm as input.
- **Encryption algorithm**: An algorithm that performs various substitutions and transformations on the plaintext.
- **Secret key**: An input value to the algorithm that is used in the substitutions and transformations; different keys produce different outputs. The key is independent of the plaintext.

- **Ciphertext**: It is the scrambled message produced as output of the encryption algorithm. The ciphered text is unintelligible.
- **Decryption algorithm**: The reverse of the encryption algorithm; it uses the ciphered text and the secret key to produce the original plaintext.

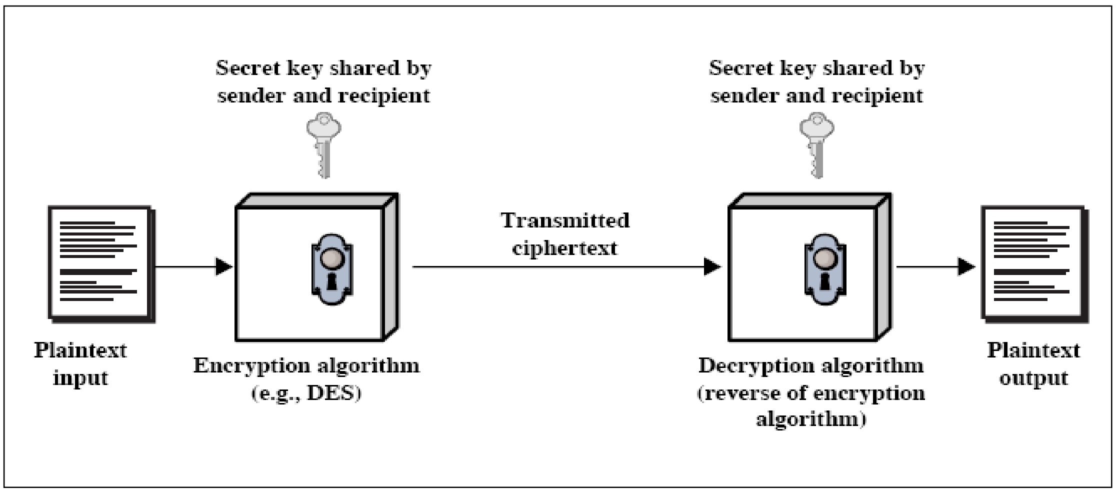

Secret key shared by sender and recipient
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Figure 0-19 A simple model of Symmetric key encryption

Symmetric encryption was built for text data but it can be used with other media. Two requirements for secure use of symmetric encryption are:

1. Strong encryption algorithm is needed. At minimum, an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the original text or figure out the key.

2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep it secure; therefore keys should be distributed through a secured channel; a model is shown in Figure 0-20 [Stallings, 2003, P.26].

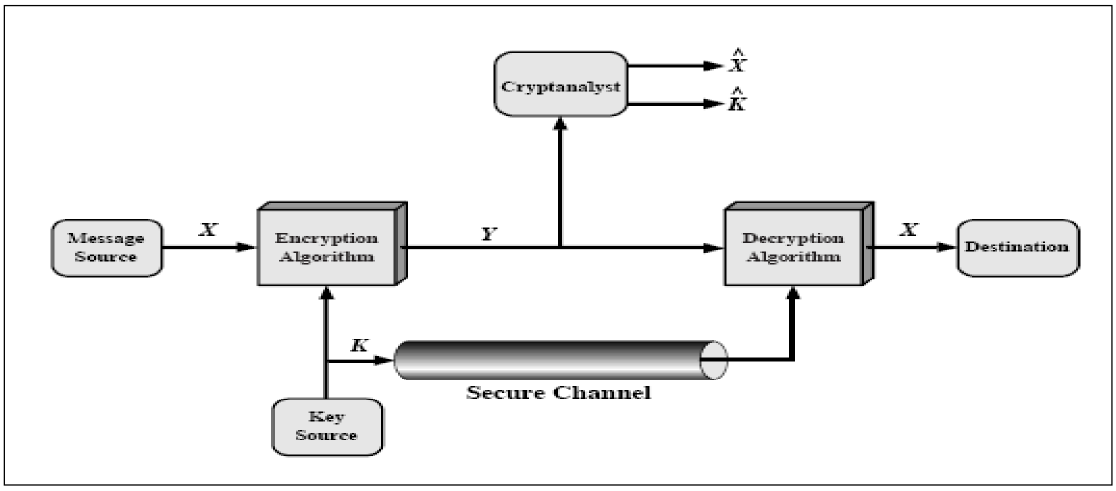


Figure 0-20 Model of symmetric key cryptosystem

An opponent observing Y but not having access to K or X, may attempt to recover X or K or both. It is assumed that the opponent knows the encryption and decryption algorithms. If the opponent is interested in this message only, then the focus is to recover X by generating an estimated plaintext $\hat{X}$. If the opponent is interested in reading further messages, then an attempt is made to recover K by generating an estimated $\hat{K}$.

2.4.2. AES_Rijndael algorithm

Rijndael algorithm -designed by Vincent Rijmen and Joan Daemen- has been selected by NIST to be the AES algorithm. It was adopted by the U.S. government in December 2001. Rijndael is a fast algorithm. It has a strong mathematical foundation. It primarily uses substitutions, transposition, and the shift, exclusive OR, and addition operations. It uses repeat cycles of 9, 11, and 13 for keys of 128, 192, and 256 bits respectively. Each cycle (called round in Rijndael) consists of four steps byte substitution, shift row, mix column, and add subkey as shown in Figure 0-21 [Pfleeger and Pfleeger 2003, Pp 69-71].

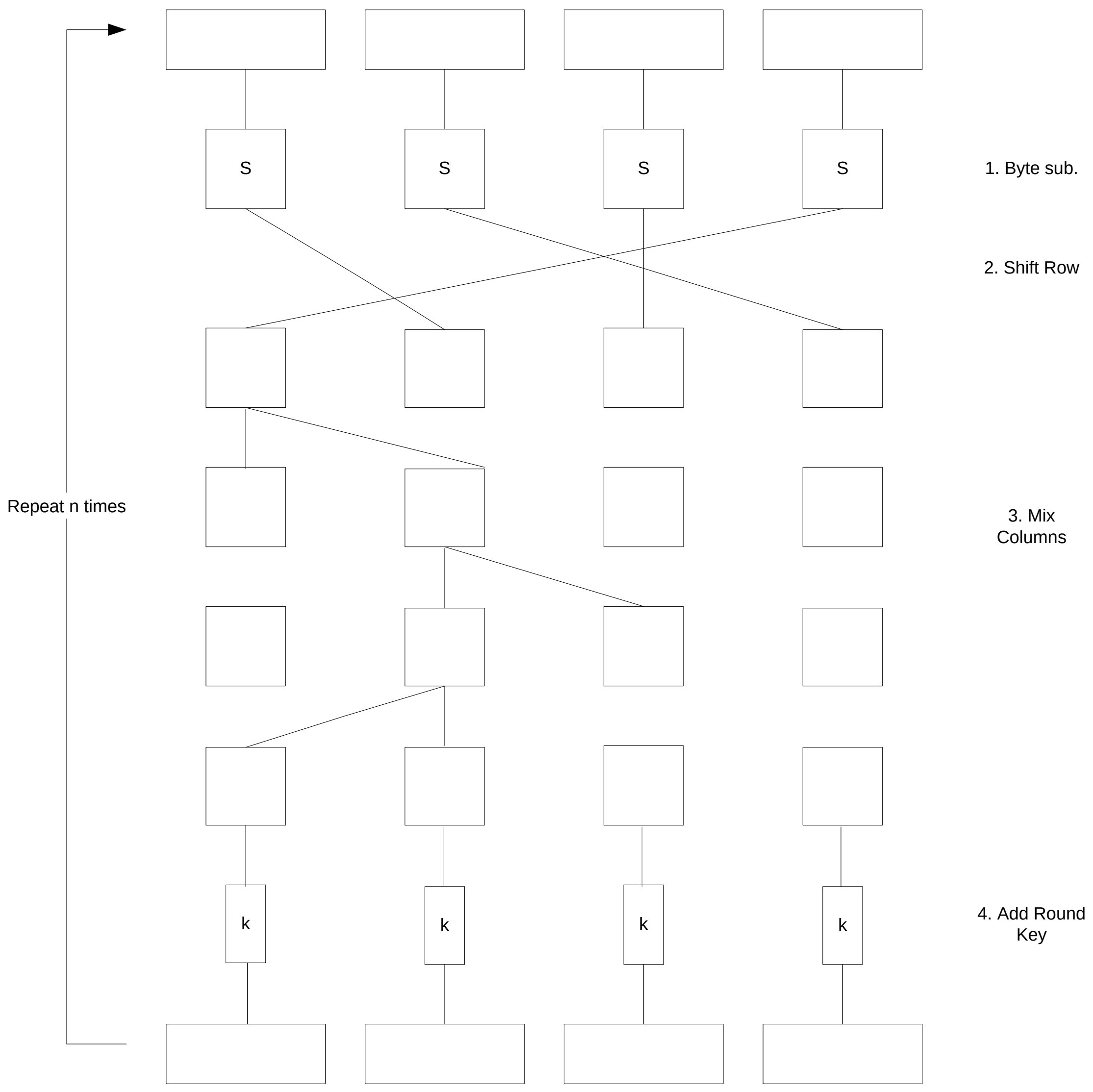


Figure 0-21: AES_Rijndael algorithm

2.4.3. Public key encryption

In this type of algorithms, a key for encryption and another key for decryption are used; the encryption key known as public key is intelligible for all while the decryption key known as private key is intelligible only to its owner; each user creates a pair of keys, if one is used for encryption the other is used for decryption. An important characteristic of public key encryption algorithms is that it should be computationally infeasible to determine the decryption key given only knowledge of the algorithm and the encryption key. Figure 0-22 [Stallings, 2003, P.261] shows the main ingredients of public key cryptography system. Public key encryption can be used to exchange the secret key between the parties in a symmetric key cryptosystem.

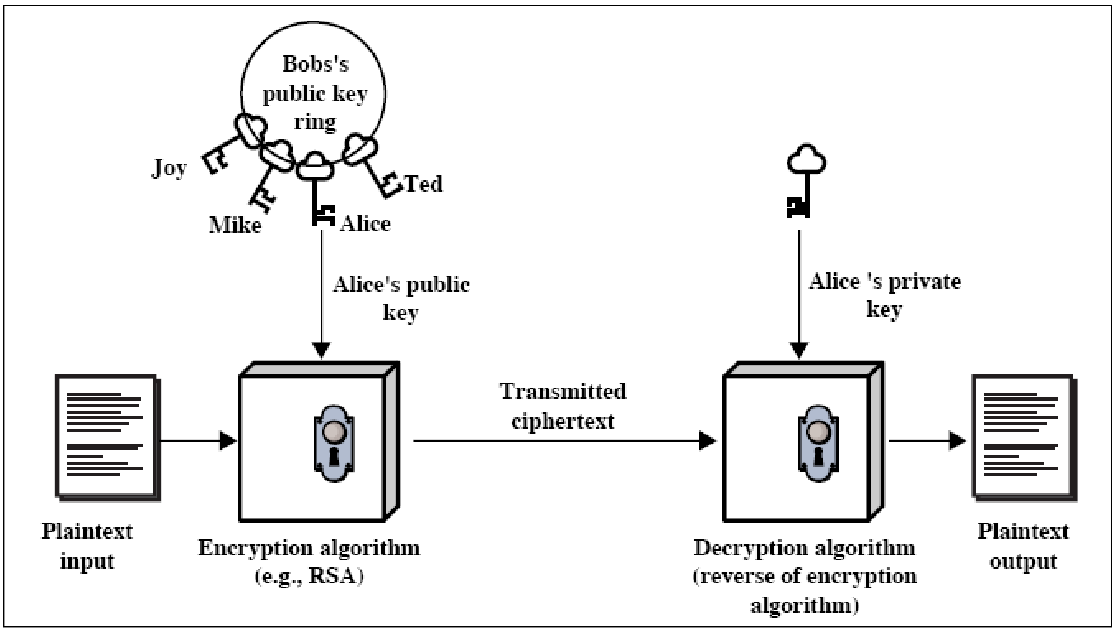


Figure 0-22 Public key encryption model

2.5. Steganography

While cryptography renders the message unintelligible to outsiders by various substitutions and transformations, the steganography conceals the existence of the message and hides it from outsiders.

A simple form of steganography is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message; other techniques were used too such as: character marking, invisible ink, pin punctures, and typewriter correction ribbon. [Stallings, 2003, PP:47-49]

Steganography can be also applied to digital data; for example an alteration of the least significant digit for the color value of each pixel in a digital image will not affect much the quality of the image therefore a message can be sent within an image using these bits. In this research, steganography is employed to send the encryption key information within the voice data. This supplies the system with an efficient method of key distribution, and allows the use of new key for each new packet.

2.6. Related work

2.6.1. Real-time Transport Protocol Header Compression (RTPC)

VoIP packet size is small; therefore, the ratio of the IP/UDP/RTP headers (40 bytes) to VoIP packet payload (20-150 bytes) is big (200% - 26.67%). Cisco resolves this problem by using a compressed RTP header. [Balliache, 2003]

"***RTPC (Realtime Transport Protocol Header Compression)***: RTP is a protocol used for carrying multimedia application traffic, including audio and video, over an IP network. RTP packets have a 40-byte header and typically a 20 to 150 payload. RTP protocol travels over UDP. Given the size of the IP/UDP/RTP header combination, it is inefficient to transmit those small payloads using an uncompressed header. RTPC is a technology that helps RTP run more efficiently, especially over lower-speed links, by compressing the RTP/UDP/IP header from 40 bytes to (2 – 5) bytes. This is especially beneficial for smaller packets (such as IP voice traffic) on slower links, where RTP header compression can reduce overhead and transmission delay significantly." [Balliache, 2003]

The drawback of this solution is the overhead delay caused by compression and decompression processes where each router will decompress this header and recompress it again. If the number of routers is relatively big such as the case of internet, then this compression/decompression processes result in a significant amount of time added to the end-to-end delay which should be kept less than a recommended threshold (150 ms).

2.6.2. Link Fragmentation and Interleaving (LFI)

VoIP packets are small, when they travel throughout lines transmitting bulk traffic, with big packets (1000-1500 bytes, and even bigger), they have to make queues on routers and therefore, they have to wait their turn on the routers to be forwarded behind, perhaps several, big packets. *Cisco* resolves this using LFI which is an incredible tool from Cisco. It's explained as this: interactive traffic like VoIP is susceptible to increase latency and jitter, when the network processes large packets (for example, LAN-to-LAN FTP big packets transverse a low bandwidth WAN link), especially when their packets (from interactive flows) are queued on these slower links. LFI reduces delay and jitter by breaking up large datagrams and interleaving low-delay traffic packets with the resulting smaller packets. [Balliache, 2003]

It is mentioned in [Balliache, 2003] that LFI has the following drawbacks:

1. Both ends have to be implemented using the same configuration. This means, you have to have *Cisco routers* on both sides implementing the same configuration.
2. Be careful when using *LFI* (*Link Fragmentation and Interleaving*). If users are planning to use some other application through the same link, they should check them first because some do not accept or permit packet fragmentation.

2.6.3. DS/TOS bits in IP frame

Differential Service (DS) field or TOS byte in IP specification can be used to assign a priority for VoIP packets. Many developers such as Cisco set the DS field to Expedited Flow (EF) as it is described in RFC 2474 (i.e. it is set to 101000). [Walker, 2001]

The door is open to use any setting of the 64 possibilities (the permutations of 6 bits). In this case a higher priority value can be assigned to voice packets.

2.6.4. Voice Activity Detection (VAD)

"Studies have shown that telephony speech patterns are on average made up of 50% silence" [comdial.com]. Enabling VAD reduces the amount of required bandwidth and the number of jitter buffers where the empty packets will not be transmitted. Voice packets are transmitted only when voice or speech is detected. [comdial.com] [Davidson & Peters, 2004, PP:177-178]

2.6.5. Voice privacy service over the public telephone network

This is a PhD dissertation prepared by Sharif, M. In this dissertation, the researchers described voice privacy architecture and protocols developed to provide a point-to-point privacy service with telephone and subscriber authentication. The architecture consists of certificate authorities (CA), authentication centers (AC), and telephone sets with cryptographic capabilities on top of the existing public telephone network infrastructure. [Sharif, 2004]

The researchers claimed that the connection establishing delay for such voice privacy is approximately 17, 19, and 21 seconds for local/regional, long distance, and international connections respectively. This delay is four times longer than the typical normal call connection establishing delay. [Sharif, 2004]

There are many differences between Sharif's research and the research presented in this dissertation, the differences are as follows:

1. Sharif's research is on PSTN while VoIP is the concern of this research.
2. They achieved their goal by increasing the call connection causing a delay four times longer than the typical normal connection; while the research of this dissertation enhances the security on VoIP with a minimum delay cost (0.00747 ms / byte).

3. They used CA and AC while embedded key management method was used in this research.
4. They used telephones with encryption capabilities while IP-phones or typical PCs are to be used in the proposed approach of this research.

2.6.6. Multiple packet-streams in encrypted voice over IP

This is a Master thesis prepared by Shuman, J. The researchers investigated the effects of packet loss, jitter buffers and decryption buffers in a proposed multiple packet-streams in encrypted voice over IP; they claimed that it can relax the time-constraint of voice packets for decryption purposes. They claimed that the security is increased by using different encryption key for each packet stream. They also claimed that the security is increased without introducing much extra resource consumption and delay. VAD and frame interleaving techniques were used in their system. [Shuman 2003]

Over-buffering allows for higher bursting capacity and lower packet loss probability, but increases the router's queue length and the experimented packet latency. On the contrary, under-buffering reduces latency but increases the packet loss probability. Therefore, a good buffering strategy should be applied on routers to make balance between packet loss and packet latency.

The research presented in this dissertation is different from that research, the differences are as follows:

1. They used available cryptosystem while new method was used in this research.
2. They concerned their self with the effect of packet loss while this research is concerned with packet delay.

2.6.7. Intelligent encryption decryption system using limited genetic algorithm and Rijndael algorithm

This is a doctoral dissertation prepared by Al Rashed, A. The researchers proposed a genetic algorithm to produce a pool of randomly generated keys; different methods were proposed to create/pick up a key among that huge number of possible keys. They proposed to send the encryption key within the ciphered data; different methods were suggested to mix the key with the ciphered data. AES_Rijndael algorithm was designed and developed using software engineering approach [Al Rashed, 2004]. That research is different from the research presented in this dissertation; differences are as follows:

1. They proposed methods to mix the key with the ciphered data; while in this research, some information about the key (not the key itself) is to be mixed with the ciphered data, this information is enough for a target receiver to regenerate/extract the key.
2. They didn't present how the target receiver will be the only one who can split the key from the ciphered data; in this research a simple and strong embedded approach was implemented and tested.
3. They didn't evaluate in details the time requirements and time complexity for their approach; while details of this were presented in this research.
4. They didn't evaluate the effects of the proposed approach on the performance of the applications; while in this research the effects of the new security method on a real-time application (i.e. VoIP) were analyzed and discussed.

Encryption and Key selection

3.1. Introduction

Data is valuable and therefore it should be kept and transmitted in a secured manner. The science that is concerned with this is known as cryptography. There are many known techniques that are used to secure data; they are classified mainly in two categories; symmetric and asymmetric methods. Symmetric encryption requires a key and a new Initialization Vector (IV) to encrypt and decrypt data. Whenever a user encrypts data, then for anyone to decrypt this data must possess the same key and IV and use the same encryption algorithm. Generally, a new key and IV must be created for every session. Neither the key nor the IV should be stored for use in a later session.

In order to communicate a symmetric key and IV to a remote party, a user should usually encrypt the symmetric key and IV using asymmetric encryption. Sending these values across an insecure network without encrypting them is extremely unsafe, as anyone that intercepts these values has the ability to decrypt the data.

Most encryption algorithms execute many iterations of substitutions and transformations on original data (known as plaintext), in order to complicate the process of identifying the data by a hacker or intruder. This long process takes a significant time; and therefore, it is not suitable for real time applications such as voice chatting over networks. Table 0-7 presents the time needed to encrypt data of different sizes using one of these algorithms (AES_Rijndael).

Creating and managing keys is an important part of any cryptographic process. Symmetric algorithms require the creation of a key and IV that must be kept secret from any unauthorized user. Asymmetric algorithms require the creation of a public key and

a private key. The public key can be made public to all, while the private key must be known only by the party who will decrypt the encrypted data with the public key. There exist many techniques used to generate keys for each of the encryption algorithms.

Voice chatting over the internet is a real time application. The ITU-T recommendations G.114 set a constraint of 150 ms one-way end-to-end delay as a ceiling of a good service. Because of this it is aimed to implement a system that employs an encryption algorithm that executes only a simple process. It seems that this approach makes it easier for a hacker to identify a data packet; this is correct if we have only one packet, but new key is used with every new packet[10]; and therefore it is not significant for a hacker to break the key for one packet of the voice data, especially when it is known that a single packet carries only a small amount of voice data – almost about 30 ms - which will not give a hacker any useful information. The security process in this research performs the following steps for each voice packet:

[10] Section 3.3 evaluates the randomness of the generated keys

Sender side:

1. Select/generate a key from a pre-generated pool of master keys.
2. Encrypt the voice data by the selected/generated key.
3. Mix the encrypted data and the key information together.

Receiver side:

4. Split the encrypted data from the key information.
5. Extract the key from the pool of master keys.
6. Decrypt the data by the extracted key.

This chapter and the following two chapters present and discuss the details for each of the mentioned steps. All the algorithms introduced in these chapters have been implemented and tested as follows:

- Computer: laptop (Centrino 1.6 GHz, 256 MB RAM, and 2 MB L2 cache).
- Operating system: Windows XP.
- Implementation language: VB.NET 2003.

3.2. Key selection/generation

In this research, a pool of master keys is generated randomly. When ever a new user subscribes, one of these master keys is assigned to the new user. The system administrator is the only user who is responsible and able to generate these master keys. These master keys will be saved as hexadecimal digits in a database[11] that is accessible by all other subscribers. A number of sub keys can be selected from any master key as follows:

a. Compose the key from a number of segments (i.e. don't select the key as single block of consecutive hexadecimal digits, but let it be composed from different and separated segments). This number of segments will be selected randomly.
b. Select randomly the key length. (i.e. a varied key length will be used).
c. Select randomly the position (index) in the master key where the first segment of the current key will start.

Assuming that:

[11] DB description exists in chapter 7.

Master key length is MKL.

Number of key segment categories is NOKSC.

Number of key length categories is NOKLC.

Number of possible keys that can be generated from each master key is NOPK.

Then,

$$NOPK = MKL \times NOKLC \times NOKSC$$ …………………………………………….. **Equation 0.1**

Example 3.1: Assume that each master key consists of 256 hexadecimal digits, possible number of key segments is 5, and number of key lengths is 7 then from Equation 0.1, the number of possible keys is:

$NOPK = 256 \times 7 \times 5 = 8960$ keys.

This number can be increased easily by increasing the values of the factors in the previous equation.

Example 3.2: Assume that each master key consists of 1024 hexadecimal digits, possible number of key segments is 10, and number of key lengths is 32 then from the previous equation, the number of possible keys is:

$NOPK = 1024 \times 32 \times 10 = 327680$ keys.

The number of possible keys that can be selected from any master key can be increased dramatically by adding more factors to the selection process; some of the factors that can be added are:

a. Direction of the key selection (DOKS) (i.e. LTR or RTL). When this parameter is selected randomly it doubles the number of possible keys since we have two categories LTR and RTL.
b. Direction of the key segments selection (DOKSS) (i.e. LTR or RTL). If this parameter is selected randomly; then NOPK is almost doubled since LTR and RTL produce different sequences except few cases which produce the same sequence in both directions.
c. By making the selection from bits level instead of the hexadecimal digits level, the possible number of keys will be increased by four multiples.
d. Length of the key segments can be classified to a number of categories (LOKSC); say four categories as follows:

Table 0-1 Categories of key segment length

Category No.	Description
0	All segments of the same length (i.e. L: L: L: …etc)
1	Segments of different lengths as L : 2L : L : 2L : …etc.
2	Segments of different lengths as 2L : L : 2L : L : … etc.
3	Segments of different lengths as L : L : 2L : L : L : 2L : … etc.

According to the above modifications, Equation 0.1 becomes:

$$NOPK = MKL \times NOKLC \times NOKSC \times 2 \times 2 \times 4 \times LOKSC \quad \ldots\ldots\ldots\ldots \quad \textbf{Equation 0.2}$$

Therefore, the number of possible keys for the above example will be:

$NOPK = 1024 \times 32 \times 10 \times 2 \times 2 \times 4 \times 4 = 20{,}971{,}520$ keys.

Algorithm: Key selection

Executed by: Conversation process

Location: Client

Time: During conversations.

Input: Receiver user name

Output: Encryption key length, start position, and number of key segments, direction of the key selection, direction of the key segments selection, key segments length category, and a key

Comments: This algorithm assumes that the master key of the receiver already retrieved / available.

Algorithm steps:

1. Select randomly number of key segments.

2. Select randomly key length category.

3. Select randomly start position of the key.

4. Select randomly direction of the key selection.

5. Select randomly direction of the key segments selection.

6. Select randomly key segments length category.

7. Compute the length of each segment

8. Compute the distances between consecutive segments.

9. Compose the key from the master key according to the selections and calculations in steps 2-8 above.

10. Pass the key to the encryption algorithm

11. Pass the key parameters selected in 2-6 above to the mixing algorithm (i.e. the algorithm that mixes key information with the encrypted data).

12. END

13.

Example 3.3: Assume that the master key consists of 256 hexadecimal digits as follows:

```
0ACDFFBFF1D4C6E5E16D259AF2F26CE20146EC2882C566DBFF

40A40A12ED09932B4A1772FE40833854267D7B0A5CE57E993B

2297821CF01DE5862B438A72B3A656F0863ABE5CBCF87D8DB8

6DE7A724CE76A91D5F51B3678CAFBDBCADF210BB862644156A

798AC366EED29E3ABD9FB43C9851033A580614F0CBB2DE5C33

ABC77B
```

Case 1:

Selected parameters

A random number of key segments = 6

A random key length category = 2 (i.e. 128 bits = 32 hexadecimal digits)

A random start position = 233

A random key selection direction is LTR

A random key segments selection direction is LTR

A random key segments length category is 0 (i.e. segments of same length)

Calculated parameters

Segment length = ceiling (32/6) = 6 hexadecimal digits

Last segment length = 32 – 5*6 = 2 hexadecimal digits

Distance between key segments = ceiling ((256-32)/6) = 38 hexadecimal digits

Last distance between key segments = (256-32) – 5 * 38 = 34 hexadecimal digits

Key composition

According to the above parameters; the key segments will be the bold segments as shown below ***where the underlined segment is the starting segment***.

0ACDFFBFF1D4C6E5E16D**259AF2**F26CE20146EC2882C566DBFF

40A40A12ED0993**2B4A17**72FE40833854267D7B0A5CE57E993B

2297821C**F01DE5**862B438A72B3A656F0863ABE5CBCF87D8DB8

6D**E7A724**CE76A91D5F51B3678CAFBDBCADF210BB862644**15**6A

798AC366EED29E3ABD9FB43C9851033A**580614**F0CBB2DE5C33

ABC77B

Therefore; the selected/generated key is:

580614259AF22B4A17F01DE5 E7A72415

Case 2:

Selected parameters

A random number of key segments = 5

A random key length category = 6 (i.e. 256 bits = 64 hexadecimal digits)

A random start position = 70

A random key selection direction is RTL

A random key segments selection direction is RTL

A random key segments length category is 0(i.e. segments of same length)

Calculated parameters

Segment length = ceiling (64/5) = 13 hexadecimal digits

Last segment length = 64 – 4*13 = 12 hexadecimal digits

Distance between key segments = ceiling ((256-64)/5) = 39 hexadecimal digits

Last distance between key segments = (256-64) – 4 * 39 = 36 hexadecimal digits

Key composition

According to the above parameters; the key segments will be the bold segments as shown ***where the underlined segment is the starting segment*****.** ***Observe that we start counting and selecting from right-to-left.***

0ACDFFBFF1D4C6E5E1**6D259AF2F26CE**20146EC2882C566DBFF

40A40A12ED09932B4A17**72FE408338542**67D7B0A5CE57E993B

2297821CF01DE5862B438A**72B3A656F0863**ABE5CBCF87D8DB8

6DE7A724CE76A91D5F51B367**8CAFBDBCADF21**0BB862644156A

798AC366EED29E3ABD9FB43**C9851033A580**614F0CBB2DE5C33

ABC77B

Therefore; the selected/generated key is:

12FDACBDBFAC83680F656A3B27245833804EF27EC62F2FA952D6085A33 01589C

Time requirements

Key selection/generation in this approach does not take a significant amount of time assuming that the master key is available, as this is usually retrieved at the beginning of the call. The time required is only the time needed to select six random numbers to represent the key factors and then composing the key from the master key by selecting the relevant key segments.

This algorithm has been tested to generate 1,500,000 keys of different key lengths and different master key lengths. The results of the tests are presented in Table 0-2. A column chart showing the relationship between the master key length and the time required to generate a key is presented in Figure 0-1.

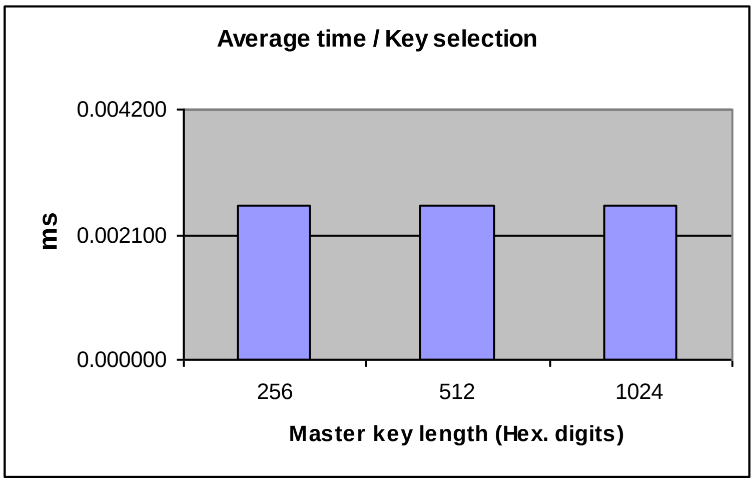


Figure 0-1 Average time / key generation

Table 0-2 Time reqirments of key generation/selection

Key Selection/Generation Process (at Sender Side)											
Master Key Length (Hex. Digits)	**Trial No.**	**Key Length (bits)**							**Total**	**Total time (ms)**	**Average Key Length (bits)**	**Average time /Key (ms)**
		64	**96**	**128**	**160**	**192**	**224**	**256**				
256	1	14210	14307	14294	14365	14328	14352	14144	100000	260.374	159.98	0.002604
	2	14382	14188	14184	14437	14266	14251	14292	100000	260.374	159.98	0.002604
	3	14256	14439	14145	14263	14233	14313	14351	100000	260.374	160.04	0.002604
	4	14331	14232	14523	14425	14186	14165	14138	100000	260.374	159.66	0.002604

	5	14218	14053	14280	14275	14155	14534	14485	100000	260.374	160.52	0.002604
	Subtotal	**71397**	**71219**	**71426**	**71765**	**71168**	**71615**	**71410**	**500000**	**1301.87**	**160.04**	**0.002604**
512	1	14509	14287	14319	14312	14251	14223	14099	100000	260.374	159.54	0.002604
	2	14067	14400	14164	14268	14339	14335	14427	100000	260.374	160.36	0.002604
	3	14042	14155	14311	14383	14185	14530	14394	100000	260.374	160.54	0.002604
	4	14288	14327	14368	14190	14293	14178	14356	100000	260.374	159.95	0.002604
	5	14414	14304	14069	14294	14433	14255	14231	100000	260.374	159.91	0.002604
	Subtotal	**71320**	**71473**	**71231**	**71447**	**71501**	**71521**	**71507**	**500000**	**1301.87**	**160.06**	**0.002604**
1024	1	14301	14116	14317	14316	14231	14235	14484	100000	260.374	160.22	0.002604
	2	14126	14330	14140	14220	14384	14321	14479	100000	260.374	160.41	0.002604
	3	14287	14313	14239	14340	14401	14194	14226	100000	260.374	159.92	0.002604
	4	14260	14257	14159	14098	14269	14620	14337	100000	260.374	160.34	0.002604
	5	14549	14265	14276	14179	14162	14114	14455	100000	260.374	159.78	0.002604

	Subtotal	**71523**	**71281**	**71131**	**71153**	**71447**	**71484**	**71981**	**500000**	**1301.87**	**160.13**	**0.002604**
Totals		**214240**	**213973**	**213788**	**214365**	**214116**	**214620**	**214898**	**1500000**	**3905.62**	**160.08**	**0.002604**

3.3.Evaluation of key randomness

The National Institute of Standards and Technology (NIST) issued a statistical test suite for evaluation of the pseudo random number generators (PRNGs). The most commonly used tests are mono-bit or frequency, serial, poker, and autocorrelation. Each of them is applied to the random sequence to get a value and compare it with a predefined range or a threshold depending on the significance level to be achieved. Each test aims to test such characteristic(s) of the random sequence. [FIPS PUB 140-2].

Monobit/Frequency test: The purpose of this test is to determine whether the number of ones and zeros in a sequence is approximately the same as would be expected for a truly random sequence. This test can be accomplished as given in Equation 0.3 [Jarrar 2004].

$FTV = (n_0 – n_1)^2 / n$ ……………………………………….. **Equation 0.3**

Where;

FTV: Frequency Test Value

n_0: Number of 0's

n_1: Number of 1's

n: The sequence length

Serial Test: The purpose of this test is to determine whether the number of occurrences of m-bit overlapping patterns is approximately the same as would be expected for a random sequence. With the assumption that m=2, this test can be accomplished as given in Equation 0.4 [Jarrar 2004].

$$STV = \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1 \quad \ldots\ldots\ldots \quad \textbf{Equation 0.4}$$

Where;

STV: Serial Test Value

n_0: Number of 0's

n_1: Number of 1's

n_{00}: Number of occurrences of the pattern 00

n_{01}: Number of occurrences of the pattern 01

n_{10}: Number of occurrences of the pattern 10

n_{11}: Number of occurrences of the pattern 11

n: The sequence length

Poker Test: The purpose of this test is to determine whether the number of ones and zeros in each of M non-overlapping blocks created from a sequence appear to have a random distribution. This test can be accomplished as given in Equation 0.5 [Jarrar 2004].

$$PTV = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. \quad \textbf{Equation 0.5}$$

Where;

PTV: Poker Test Value

m : Length of each pattern in bits

k : Number of non-overlapping patterns

n_i : Number of occurrences of the ith pattern

Autocorrelation test: The purpose of this test is to check the correlation between the sequence and the shifted versions of it. This correlation value can be computed as given in Equation 0.6 [Jarrar 2004].

$$ATV = 2\left(A(d) - \frac{n-d}{2}\right) / \sqrt{n-d}$$ **Equation 0.6**

………………..……………..

Where;

$$A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$$ **Equation 0.7**

……..……………….………………..

d: any fixed integer, $1 \le d \le \lfloor n/2 \rfloor$

$\oplus$: The XOR operation.

The frequency, serial, poker, and autocorrelation tests were applied on ten thousand keys generated using the proposed method as follows:

- The master key length is 1024 hexadecimal digits (i.e. 4096 bits).
- The key length is varied.
- The significance level ($\alpha = 0.05$).
- In poker test, m is assumed to be 8 bits.

- In autocorrelation test, d is assumed to be (key length / 2).

The tests result is summarized in tables 3-3 and 3-4. The result shows that most of the generated keys passed all the tests, and none of them failed in all tests or passed only one test.

Table 0-3: Randomness statistical tests result

Statistical test	**Number of keys passed the test**	**Percentage**
Frequency	9512	95.12%
Serial	10000	100%
Poker	8041	80.41%
Autocorrelation	10000	100%

Table 0-4: Randomness statistical tests cumulative result

Number of passing tests	**Frequency**	**Percentage**
4	7707	77.07%
3	2139	21.39%
2	154	1.54%
1	0	0%
0	0	0%

In addition to the previous tests we tested the key repetition; the results of this test are summarized in Table 0-5. It is clear from the results that the percentage of repeated keys increases as the call duration increases, and decreases as the number of possible keys increases. To overcome this problem, the following solutions are suggested:

- Tolerate this percentage of key repetition.

- Checking for key repetition and ignoring any repeated key.
- Using a stronger random number generator if available.
- Increasing the number of possible keys that can be generated from the master key.

Table 0-5: Repetition check of the key generation

# of possible keys	Call duration (minutes)	# of keys generated	Average number of repeated keys	Percent of repeated keys	Average distance between repeated keys	Percent of the distance between repeated keys
2621440	5	10000	21	0.21%	3574	36%
	10	20000	88	0.44%	6567	33%
	30	60000	763	1.27%	20084	33%
5242880	5	10000	9	0.09%	3389	34%
	10	20000	46	0.23%	6993	35%
	30	60000	367	0.61%	19530	33%

3.4. Encryption

Encryption is the process of converting original data to a form that is not understandable unless reconverted to its original form by a process known as decryption. As mentioned in the introduction, encryption algorithms execute many rounds of substitutions and transformations on the original data in order to make it impossible for hackers to identify the encrypted data.

In this research, it is aimed to use a simple XOR process for the encryption of the voice data in order to minimize the time needed for this process. This is because each millisecond is significant in real time applications such as VoIP. On the other hand the security level is increased by using a different key for each new packet of the voice data. This means that the disclosing of a packet of this voice data by a hacker is not significant because each packet carries only a few milliseconds of voice data which is not understandable to a hacker who can't identify the other packets as each packet has its own key and there is no relation between different keys.

Algorithm: Encryption

Executed by: Conversation process

Location: Client

Time: During conversations.

Input: 1) The selected encryption key

2) Plain digitized voice data

Output: Ciphered voice data

Comments: None

Algorithm steps:

```
1. Get a buffer for the voice data that has been captured by the voice capturing device.
2. Convert the voice data into a bits string array
3. Convert the selected key for this packet into a bits string array
4. While not end-of-the-voice-bits-array
       i) Get a number of bits equal to the length of the key bits array;
       ii) Do XOR of this segment of the voice bits with the key bits.
5. Pass the ciphered data to the mixing algorithm.
6. END
```

Example 3.4: Assume that a selected key for a packet is 16 hexadecimal digits (i.e. 64 bits) as follows: $\boxed{\text{A236BB8DD70FA35D}}$.

This will be converted to the bit array to be:

```
1010001000110110101110111000110111010111000011111010001101011101
```

Assume also that a segment of the voice bit array that represents this packet is as follows:

```
1110011001010100000110000110101010001010100001111010110110000101

1111111111111111111100110011001100000000111100000000010101000011

1000000010011100011011100000000000011111111111111110111111101001
```

The XORing of this voice data segment with the above key is:

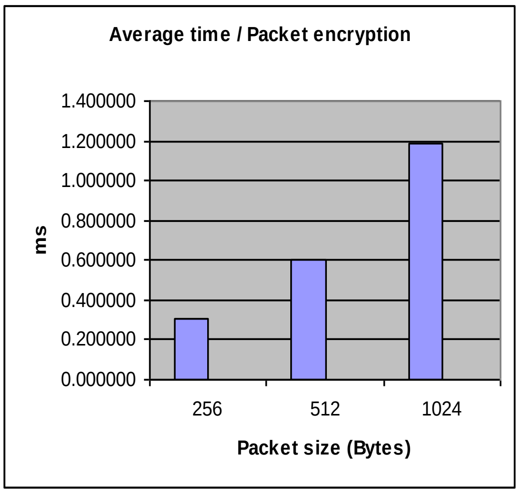
```
0100010001100010101000111110011101011101100010000000111011011000

0101110111001001010010001011111011010111111111111010011000011110

0010001010101010110101011000110111101000111100000100110010110100
```

Time requirements

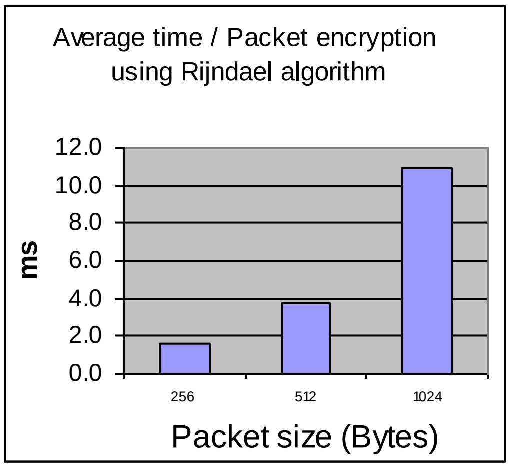
The encryption process in this approach takes a small amount of time to encrypt a packet of voice data compared to other known encryption algorithms. This amount of time will not add much latency to the transmission of the voice packet. Therefore, this approach can be used to secure the voice data that travels over networks and/or Internet.

This algorithm has been implemented and tested to encrypt 1,500,000 packets of different key lengths and different packet lengths. The results of that test are presented in Table 0-6.

The AES_Rijndael algorithm was also implemented♠. This algorithm was used to encrypt 150,000 packets using different key lengths for different packet lengths. The results of that test are presented in Table 0-7. Column charts showing the relationship between the packet size and the encryption time in both cases are presented in Figure 0-2.



(a)



(b)

Figure 0-2 Average time / Packet encryption (a) using the proposed solution and (b) using Rijndael algorithm

♠ The program was customized from a program taken from the internet. [Obviex]

Table 0-6 Time requirments for encryption process using the proposed solution

Encryption Process												
Packet size (Bytes)	Trial No.	Key Length (bits)							Total	Total time (ms)	Average Key Length (bits)	Average time/Packet (ms)
		64	96	128	160	192	224	256				
256	1	14439	14316	14284	14245	13913	14350	14453	100000	30754.22	159.92	0.307542
	2	14346	14439	14149	14139	14285	14356	14286	100000	30694.14	159.93	0.306941
	3	14265	14410	14492	14243	14295	14203	14092	100000	30844.35	159.64	0.308444
	4	14239	14313	14204	14364	14322	14347	14211	100000	30573.96	160.03	0.305740
	5	14354	14119	14360	14326	14203	14224	14414	100000	30754.22	160.07	0.307542
	Total	**71643**	**71597**	**71489**	**71317**	**71018**	**71480**	**71456**	**500000**	**153620.89**	**159.92**	**0.307242**
512	1	14367	14170	14199	14431	14202	14263	14368	100000	59946.20	160.06	0.599462
	2	14321	14217	14226	14382	14263	14053	14538	100000	60186.54	160.12	0.601865
	3	14236	14169	14219	14326	14230	14370	14450	100000	59896.13	160.34	0.598961
	4	14290	14260	14317	14304	14467	14221	14141	100000	60176.53	159.88	0.601765

	5	14200	14350	14417	14305	14455	13984	14289	100000	59805.99	159.86	0.598060
	Total	**71414**	**71166**	**71378**	**71748**	**71617**	**70891**	**71786**	**500000**	**300011.39**	**160.05**	**0.600023**
1024	1	14315	14263	14146	14366	14189	14276	14445	100000	118049.75	160.15	1.180498
	2	14321	14239	14430	14469	14198	14117	14226	100000	118270.06	159.76	1.182701
	3	14295	14405	14100	14302	14253	14253	14392	100000	119041.17	160.04	1.190412
	4	14100	14264	14255	14303	14356	14464	14258	100000	119491.82	160.31	1.194918
	5	14200	14322	14382	14463	14143	14298	14192	100000	118570.50	159.90	1.185705
	Total	**71231**	**71493**	**71313**	**71903**	**71139**	**71408**	**71513**	**500000**	**593423.3**	**160.03**	**1.186847**

Table 0-7 Time requirements for encryption process using Rijndael algorithm

Encryption Process - AES-Rijndael							
Packet size (Bytes)	Trial No.	Key Length (bits)			Total	Total time (ms)	Average time/Packet (ms)
		128	192	256			
256	1	3307	3392	3301	10000	15322	1.532
	2	3379	3346	3275	10000	15442	1.544
	3	3290	3311	3399	10000	15432	1.543
	4	3323	3269	3408	10000	16303	1.630
	5	3338	3365	3297	10000	15472	1.547
	Total	**16637**	**16683**	**16680**	**50000**	**77971**	**1.559**
512	1	3313	3333	3354	10000	38315	3.832
	2	3360	3195	3445	10000	37935	3.794
	3	3354	3374	3272	10000	37925	3.793
	4	3347	3234	3419	10000	37844	3.784
	5	3345	3276	3379	10000	37915	3.792
	Total	**16719**	**16412**	**16869**	**50000**	**189934**	**3.799**
1024	1	3320	3307	3373	10000	108806	10.881
	2	3391	3299	3310	10000	108977	10.898
	3	3382	3322	3296	10000	108636	10.864
	4	3330	3359	3311	10000	108907	10.891
	5	3285	3329	3386	10000	108887	10.889
	Total	**16708**	**16616**	**16676**	**50000**	**544213**	**10.884**

3.4. Conclusions and recommendations

Conclusions:

1. Most of the generated keys using the proposed method passed all the applied statistical tests, and none of the keys failed in all tests or passed only one test.
2. The average time needed to generate a key in this research is very small, around (**0.002604ms**)
3. The length of the master key has no effect on the time needed to generate a key; where the master key length does not add any overhead on the process except a negligible time needed to manage memory locations when the master key length is increased.
4. The average time needed to encrypt a voice packet using the proposed algorithm can be acceptable (relatively small, it is **1.187 ms** for a packet of size **1024 bytes**).
5. The average time needed to encrypt the same size packet in Rijndael algorithm is (**10.884 ms**) which is considered to be big for this type of applications.
6. The average time needed to encrypt a packet using the proposed algorithm is much smaller than that in Rijndael algorithm. ***The ratio is about 1:10 with a packet of size 1024 bytes***.
7. The packet length affects the time needed to encrypt the packet in both cases. In the proposed algorithm, if the packet size is doubled, then the average time needed to encrypt the packet is almost doubled too; this is expected since the process manipulates each bit of the data. In Rijndael algorithm, if the packet size is doubled, then the average time needed to encrypt a packet becomes about three multiples.

3.5. Recommendations:

1. The proposed encryption algorithm works with keys of any length. Therefore, it is recommended to enlarge the range of the key lengths in order to maximize the number of possible keys (NOPK) in Equations 3.1 and 3.2.
2. Since the packet size has an effect on the encryption time as mentioned in point 6 of the above conclusions, then it is recommended to minimize the packet size in order to minimize the time needed for the encryption process; but this is a trade off with the overhead that comes from the headers added to each packet.
3. Since the known encryption algorithms require a significant amount of time as presented in points 4 and 5 of the above conclusions, it is recommended to apply a simple algorithm with minimum computations.

Key management and distribution

4.1. Introduction

The aim of this chapter is to introduce a new technique for key management and distribution. This technique is based on mixing the key with the encrypted data. The receivers of the encrypted data should on the other hand be able to extract the key then use it to decrypt the ciphered data. Following is a number of mixing techniques that can be followed to achieve the mixing process; these techniques are:

1. Mix the key itself with the encrypted data.

2. Mix information about the key which can be used to extract the key.

In this research the second method is selected where the information about a key is much less than the key itself. This results in less overhead onto the data packet. The mixing process itself can be achieved through a lot of approaches such as:

a. Insert the key information as one segment into a pre-defined location.

b. Insert the key information as a number of segments into pre-defined locations.

c. Insert the key information as one segment into a random location.

d. Insert the key information as a number of segments into random locations.

The insertion of key information in a pre-defined location(s) is less secure than distributing the information in a number of locations defined randomly; also, the insertion of key information as a single segment is less secure than distributing the information in a number of segments defined randomly. In this research, the fourth approach is used since it makes it more difficult for a hacker or intruder to separate the key information from the encrypted data. Many methods can be followed to make the segmentation and mixing of the key information with the encrypted data. The following sections describe one of these techniques that can be used to mix the key information with the encrypted data at the sender side, and how it can be separated again at the receiver side. Any subscriber except the authorized receiver of the data will not be able to split the correct key information from the encrypted data of any packet even if he/she uses the same algorithm. If some of the key information altered during the transmission which rarely happens, then the correct receiver will compose a wrong key, and then a wrong data will be produced. In this case, it is suggested that the receiver ignore the packet since the application is tolerance for some packet loose.

4.2.Key information and ciphered data mixing

Once the voice data is encrypted using the relevant key as was described earlier, the key information and the encrypted data are mixed first and then transmitted to their destination. The receiver of the packet on the other side should be able to split key information from the encrypted data and should be the only user who is allowed to do this. As mentioned in the introduction, many methods can be applied to achieve this mixing. The following algorithm describes one of these methods.

Algorithm: Key mixing

Executed by: Conversation process

Location: Client

Time: During conversation.

Input:

1. The selected encryption key information
2. Ciphered voice data packet

Output: Mixed data of ciphered voice data and key information

Comments: None

Algorithm steps:

1. Convert the key information to bits array as follows:

```
number of key segments : 3 bits
key length category: 2 bits
start position : 10 bits
direction of the key selection : 1 bit (0: LTR and 1: RTL)
direction of the key segments selection: 1 bit (0: LTR and 1: RTL)
category of the key segments length : (2 bits)
not used : 5 bits[12]
```

2. Select the start mixing category as follows[13]

```
If [Receiver ID] is odd then
    start mixing category = "K" (i.e. start with key information)
ELSE
    start mixing category = "D" (i.e. start with data)
```

3. Compute number of ciphered data segments (3 –6)

(e.g. [Receiver ID] mod 4 + 3) [14]

4. Compute the length of each data segment

5. Compute number of segments for the key information depending on the output of steps 2 and 3 above.

[12] Those bits can be added to the other parameters to increase the range for any of them.
[13] The same policy should be used by both mixing and splitting algorithms.
[14] The same equation should be used by both mixing and splitting algorithms.

6.

7. Compute the length of each segment of the key information

8. Mix ciphered data segments and the key information segments according to the selected strategy in step 2 above.

9. Pass the mixed data to the transmission algorithm

10. END

Example 4.1: Assuming that G.726 standard (i.e. ADPCM) is used to digitize the voice; then the packet size will be 1200 bits. Assume that the binary stream of Figure 0-1 is the ciphered voice packet. Also assume that the key is the same that was introduced in case 1 of example 3.3 of the previous chapter, then:

```
110000011100111110001111111111000000101000000101011111001101
110000011100111110000000000001100000010100000010101111001101
110001111100111110001111111111000111101000000101011111001101
000000011100111110001111111111111111101000000101011111001101
111111111100111110001111111111000000101000000101011111001101
110000011100111110001111111111000000101000000101011111001101
000000011100111110001111111111111111101000000101011111001101
111111111100111110001111111111000000101000000101011111001101
110000011100111110000000000001100000010100000010101111001101
111111111100111110001111111111000000101000000101011111001101
111111111100111110001111111111000000101000000101011111001101
111111111100111110001111111111000000101000000101011111001101
110000011100111110000000000001100000010100000010101111001101
110001111100111110001111111111000111101000000101011111001101
000000011100111110001111111111111111101000000101011111001101
110000011100111110000000000001100000010100000010101111001101
110001111100111110001111111111000111101000000101011111001101
000000011100111110001111111111111111101000000101011111001101
111111111100111110001111111111000000101000000101011111001101
110000011100111110001111111111000000101000000101011110001101
```

Figure 0-1 Ciphered voice data stream

Receiver ID = 3

Number of key segments = 6 (i.e. 110)

Key length category = 2 (i.e. 10)

Start position = 233 (i.e. 0011101001)

Key selection direction is LTR (i.e. 0)

Key segments selection direction is LTR (i.e. 0)

Key segments length category is 0 (i.e. 00)

Therefore; the key information bit array that will be mixed with the ciphered data is

$\boxed{110100011101001000000000}$

Now:

Number of ciphered data segments = 3 mod 4 + 3 = 6

Start mixing category is "K" since receiver ID is odd

Ciphered data segment length = ceiling (1200/6) = 200

Last ciphered data segment length = 1200 - 200 * 5 = 200

Key information segment length = ceiling (24/6) = 4

Last key information segment length = 24 – 4 * 5 = 4

Therefore; the mixing output will be as shown in Figure 0-2.

1101110000001110011111000111111111100000001010000000101011111001101
110000001110011111000000000000110000000101000000010101111 1001101
110001111100111110001111111111000111101000000010101111 1001101
0000000111001111100**0001**1111111111111111110100000001010111111001101
111111111100111110001111111111000000010100000001010111 11001101
110000001110011111000111111111100000001010000000101011111001101
00000001110011111000111111111111111111010**1101**000000101011111001101
111111111100111110001111111111000000010100000001010111 11001101
110000001110011111000000000000110000000101000000010101111 1001101
111111111100111110001111111111000000010100000001010111 11001101
0010111111111100111110001111111111000000010100000001010111 11001101
111111111100111110001111111111000000010100000001010111 11001101
110000001110011111000000000000110000000101000000010101111 1001101
1100011111001111100**0000**1111111111000111101000000010101111 1001101
000000011100111110001111111111111111110100000001010111 11001101
110000001110011111000000000000110000000101000000010101111 1001101
110001111100111110001111111111000111101**0000**000000101011111001101
000000011100111110001111111111111111110100000001010111 11001101
111111111100111110001111111111000000010100000001010111 11001101
110000001110011111000111111111100000001010000000101011110001101

Figure 0-2 Key information mixed with a ciphered voice data stream

Comment: The bold and enlarged digits represent the key information that has been mixed with the ciphered data.

Time requirements

The algorithm described earlier was implemented and tested to mix the key information with the ciphered data of 1,500,000 packets of different sizes using different keys of different lengths for each packet. The results of that test are presented in Table 0-1. A column chart showing the relationship between the packet size and the time needed to mix the key information with the ciphered data is presented in Figure 0-3. It is clear from the chart that they have a direct relation.

Table 0-1 Time requirments for mixing a voice data packet and a key information

Mixing Data and Key Information				
Packet size (Bytes)	**Trial No.**	**Number of packets**	**Total time (ms)**	**Average time/Packet (ms)**
256	1	100000	67276.74	**0.6727674**
	2	100000	67176.60	0.6717660
	3	100000	67557.14	0.6755714
	4	100000	67286.75	0.6728675
	5	100000	67276.74	0.6727674
	Total	**500000**	**336574**	**0.67315**
512	1	100000	133702.3	1.3370225
	2	100000	133622.1	1.3362214
	3	100000	133722.3	1.3372228
	4	100000	133962.6	1.3396263
	5	100000	133712.3	1.3371227
	Total	**500000**	**668721.6**	**1.33744**
1024	1	100000	266743.6	2.6674356
	2	100000	267164.2	2.6716416
	3	100000	266633.4	2.6663340
	4	100000	267284.3	2.6728434
	5	100000	266623.4	2.6662339
	Total	**500000**	**1334449**	**2.66890**

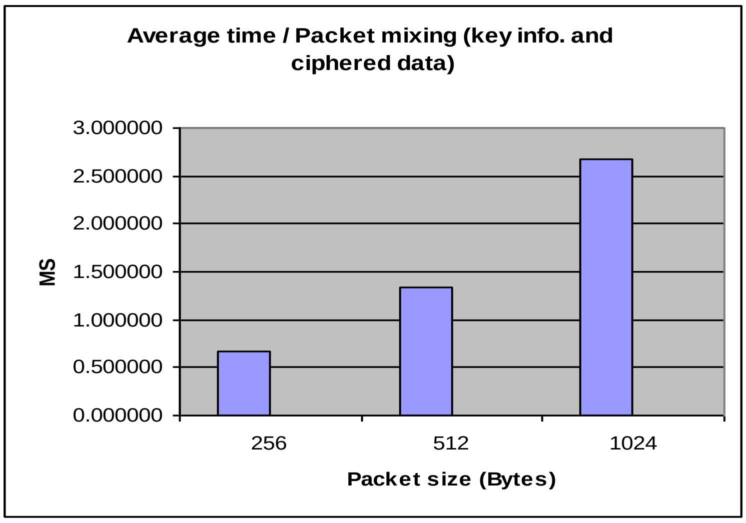


Figure 0-3 Average time / packet mixing (key information and ciphered data)

4.3. Separation of key information from ciphered data

Once the data packet received by the destination user the key information should be separated from the encrypted data. As mentioned earlier, the relevant receiver should be the only one who is able to correctly carryout this separation. The following algorithm describes the inverse process of the mixing algorithm which was presented in the previous section. This algorithm describes how to separate the key information from the encrypted data.

Algorithm: key separation

Executed by: Conversation process

Location: Client

Time: During conversation.

Input:

1. Receiver user name.
2. Mixed data of ciphered voice data and key information.

Output:

1. Bits array of the key information.
2. Bits array of ciphered voice data.

Algorithm steps:

1. Compute number of ciphered data segments (3 – 6)

   (e.g. Collie ID mod 4 + 3) [15]

2. Select the start mixing category as follows [16]

```
If Collie-ID is odd then
    start mixing category = "K" (i.e. start with key)
ELSE
    start mixing category = "D" (i.e. start with data)
```

3. Compute the length of each data segment

4. Compute the number of segments for the key information depending on the output of steps 2 and 3 above.

5. Compute the length of each segment of the key information

6. Separate ciphered data from key information according to the selected strategy in step 2 above.

[15] The same equation should be used by both mixing and splitting algorithms.
[16] The same policy should be used by both mixing and splitting algorithms.

7.

8. Pass the key information bit array to the key extraction algorithm

9. Pass the ciphered data bit array to the decryption algorithm

10. END

Example 4.2:

Case 1: Assuming that the mixed data of example 4.1 above received by the user of ID 3 (*i.e. the correct receiver*), then the following calculations will take place:

Number of ciphered data segments = 3 mod 4 + 3 = 6

Start mixing category is "K" since collie ID is odd

Ciphered data segment length = ceiling (1200/6) = 200

Last ciphered data segment length = 1200 - 200 * 5 = 200

Key information segment length = ceiling (24/6) = 4

Last key information segment length = 24 – 4 * 5 = 4

Therefore; the algorithm will correctly separate the key information from the ciphered data as follows:

Key information = $\boxed{\textbf{110100011101001000000000}}$

Ciphered data as it is in Figure 0-2.

Case 2: Assuming that the mixed data of the previous example received by the user of ID 4 (*i.e. an incorrect receiver*), then the following calculations will take place:

Number of ciphered data segments = 4 mod 4 + 3 = 3

Start mixing category is "D" since collie ID is even

Ciphered data segment length = ceiling (1200/3) = 400

Last ciphered data segment length = 1200 - 400* 2 = 400

Key information segment length = ceiling (24/3) = 8

Last key information segment length = 24 – 8 * 2 = 8

Therefore; the algorithm will not separate the correct key information from the ciphered data as follows:

- The extracted key information will be: $\boxed{\textbf{111110101111100010001101}}$
- The Ciphered data is shown in Figure 0-4 excluding the bold digits which they represent the key information.

1101110000011100111110001111111111000000101000000101011111001101
11000001110011111000000000000110000001010000000101011111001101
11000111110011111000111111111100011110100000001010111110011 01
0000000111001111100000011111111111111111010000000101011111001101
11111111110011111000111111111100000010100000001010111110 01101
11000001110011111000111111111100000010100000001010111110 01101
00000001110011111000111111111111**11111010**110100000010101111100 1101
11111111110011111000111111111100000010100000001010111110 01101
11000001110011111000000000000110000001010000000101011111001101
11111111110011111000111111111100000010100000001010111110 01101
0010111111111110011111000111111111100000010100000001010111110 01101
11111111110011111000111111111100000010100000001010111110 01101
11000001110011111000000000000110000001010000000101011111001101
110001111100**11111000**0000111111111100011110100000001010111110 01101
000000011100111110001111111111111111101000000010101111100 1101
11000001110011111000000000000110000001010000000101011111001101
1100011111001111100011111111110001111010000000000101011111001101
000000011100111110001111111111111111101000000010101111100 1101
11111111110011111000111111111100000010100000001010111110 01101
110000011100111110001111111111000000101000000010101111**10001101**

Figure 0-4 Key information mixed with a ciphered voice data stream (copy of Figure 0-2)

It is clear that wrong key information and wrong ciphered data will be passed to the key extraction and decryption algorithms, and therefore; wrong voice data will be produced.

Time requirements

The algorithm was implemented and tested; the test was carried out on 1,500,000 ciphered data packets. The test extracted key information from the ciphered data for each of the received packets. Ciphered data packets were of different sizes. The results of the test are presented in Table 0-2. A column chart showing the relationship between the packet size and the time needed to split key information from the ciphered data is presented in Figure 0-5.

Table 0-2 Time requirments for key information separtion from ciphered data

Separation of Ciphered Data and Key Information				
Packet size (Bytes)	**Trial No.**	**Number of packets**	**Total time (ms)**	**Average time/Packet (ms)**
256	1	100000	18246.24	0.182462
	2	100000	18306.32	0.183063
	3	100000	18246.24	0.182462
	4	100000	18456.54	0.184565
	5	100000	18306.56	0.183066
	Total	**500000**	**91561.9**	**0.183124**
512	1	100000	36442.54	0.364425
	2	100000	36482.46	0.364825
	3	100000	36452.42	0.364524
	4	100000	36322.23	0.363222
	5	100000	36452.42	0.364524
	Total	**500000**	**182152.07**	**0.364304**
1024	1	100000	72514.27	0.725143
	2	100000	72674.51	0.726745
	3	100000	72524.28	0.725243
	4	100000	72814.7	0.728147
	5	100000	73105.12	0.731051
	Total	**500000**	**363632.88**	**0.727266**

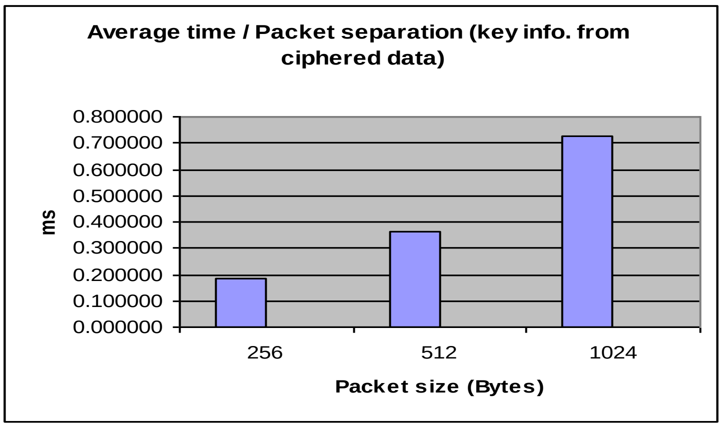


Figure 0-5 Average time / packet separation (key information from ciphered data)

4.4 Conclusions and recommendations

Conclusions

1. The average time needed to mix key information with the ciphered data of one packet is very small. It is **0.67315 ms** for a packet of size **256 bytes**. Also, the average time needed for the reverse process (i.e. splitting key information from the ciphered data of one packet) is very small too. It is **0.183124 ms** for a packet of size **256 bytes**.
2. The packet size affects this time in both processes, i.e. key mixing and separation. If the packet size is doubled, then the average time needed to mix/split key information into/from the ciphered data is almost doubled too; this is expected since these processes manipulate each bit of the data.

Recommendations

1. Because of the fact that was presented in point 2 of the above conclusion, it is recommended to minimize the packet size in order to minimize the time needed for the mixing process; but again this is a trade off with the overhead that comes from the headers added to each packet.

2. The average time needed to split key information from the ciphered data of a packet in the separation algorithm is much less than the time needed to mix the key information into the ciphered data in the mixing algorithm. This is expected since the separation process makes few calculations to determine the locations of the key information and then extracts them from the mixed data. However this difference would not affect the overall process; since it is the average time for the total operation that is of concern to us, and should be kept to a minimum value.

Key extraction and Decryption

5.1. Introduction

Decryption is the inverse of the encryption process. It converts a ciphered data into its original form. In order to decrypt any ciphered data, the encryption key and the algorithm should be known. Therefore, the correct key should be extracted using the mixing information before the decryption process can be carried on the encrypted data.

As it was mentioned earlier in chapter 4, encryption algorithms execute many rounds of substitutions and transformations on the original data in order to make it impossible for a hacker or intruder to understand data. The same number of substitutions and transformations must be performed in reverse order to decrypt the ciphered data.

In the proposed technique, a simple XOR process is used for the encryption of the voice data, this is to minimize the time needed for this process; this simple encryption process produces a highly complex ciphered result since the data is split into different packets and no one packet has enough information to recover the encrypted data packets. Decryption process is also a simple XOR process on the ciphered data with the relevant key. This method for encryption/decryption minimizes the overhead time produced from the process of securing the voice data that travels over the network. At the same time it is strongly enough to secure the data.

Also in this chapter, the key extraction and data decryption algorithms are described in details.

5.2. Key extraction

At this stage, the bit array of the key information is received from the key separation algorithm which was presented in the previous chapter. This algorithm extracts the key information embedded with the encrypted data, the key information is then used to obtain the key from the receiver master key . If the user is the correct receiver of the data, correct information will be received, and then the algorithm will compose the correct key (i.e. the same key that was generated in the key selection algorithm). If the user is not the intended receiver wrong information will be extracted, and then a wrong key will be composed. In both cases the composed key will be passed to the decryption algorithm which produces a correct data in the former case and wrong data in the latter case.

Algorithm: Key extraction

Executed by: Conversation process

Location: Client

Time: During conversation.

Input: 1) User name;

2) The bits array of the key information.

Output: The encryption key.

Comments: None

Algorithm steps:

1. Extract the decimal equivalent value for each of the key information parameters from the bits array of the key information as follows:

number of key segments : bits 1-3

key length category: bits 4-5

start position : bits 6-15

direction of the key selection : bit 16 (0: LTR and 1: RTL)

direction of the key segments selection: bit 17 (0: LTR and 1: RTL)

category of the key segments length : bits 18-19

2. **not used : bits 20-24**
3. Compose the key from the master key of the user according to the extracted information of step 1 above.
4. Pass the composed key to the decryption algorithm.
5. END

Example 5.1:

Case 1: The extracted key information in case 1 of example 4.2 was (**110100011101001000000000**), and the data has been received by a user with ID 3. Therefore:

Received parameters

Number of key segments = 110_2 = 6

Key length category = 10_2 = 2 (i.e. 128 bits = 32 hexadecimal digits)

Start position = 0011101001_2 = 233

Start position[17] = (Start position mod 256) = 233

Key selection direction = 0_2 = 0 (i.e. LTR)

Key segments selection direction = 0_2 = 0 (i.e. LTR)

Key segments length category = 00_2 = 0 (i.e. segments of the same length)

Calculated parameters

Segment length = ceiling (32/6) = 6 hexadecimal digits

Last segment length = 32 – 5*6 = 2 hexadecimal digits

Distance between key segments = ceiling ((256-32)/6) = 38 hexadecimal digits

Last distance between key segments = (256-32) – 5 * 38 = 34 hexadecimal digits

[17] The mod operation is necessary because a wrong key information that has been extracted from a wrong user may yield in a start position that exceeds the master key length.

Key composition

According to the above parameters; the key segments is the bold segments as shown below ***where the underlined segment is the starting segment***.

0ACDFFBFF1D4C6E5E16D**259AF2**F26CE20146EC2882C566DBFF

40A40A12ED0993**2B4A17**72FE40833854267D7B0A5CE57E993B

2297821C**F01DE5**862B438A72B3A656F0863ABE5CBCF87D8DB8

6D**E7A724**CE76A91D5F51B3678CAFBDBCADF210BB862644**15**6A

798AC366EED29E3ABD9FB43C9851033A**580614**F0CBB2DE5C33

ABC77B

Therefore; the extracted key is:

580614259AF22B4A17F01DE5 E7A72415

Observe that it is the same key that was produced by case 1 of example 3.3. This correct key and correct ciphered data will be passed to the decryption algorithm which means that it will produce the original data.

Case 2: The extracted key information in case 2 of example 4.2 was (**111110101111100010001101**), and the data has been received by a user with ID 4. Therefore:

Received parameters

Number of key segments = 111_2 = 7

Key length category = 11_2 = 3 (i.e. 160 bits = 40 hexadecimal digits)

Start position = 0101111100_2 = 380 mod 256 = 124

Key selection direction = 0_2 = 0 (i.e. LTR)

Key segments selection direction = 1_2 = 1 (i.e. RTL)

Key segments length category = 00_2 = 0 (i.e. segments of the same length)

Calculated parameters

Segment length = ceiling (40/7) = 6 hexadecimal digits

Last segment length = 40 – 6*6 = 4 hexadecimal digits

Distance between key segments = ceiling ((256-40)/7) = 31 hexadecimal digits

Last distance between key segments = (256-40) – 6 * 31 = 30 hexadecimal digits

Key composition

According to the above parameters; the key segments will be the bold segments as shown below ***where the underlined segment is the starting segment***.

0ACDFFBFF1D4C6E**5E16D2**59AF2F26CE20146EC2882C566DBFF

40**A40A12**ED09932B4A1772FE40833854267D7B0**A5CE**57E993B

2297821CF01DE5862B438A7**2B3A65**6F0863ABE5CBCF87D8DB8

6DE7A724CE**76A91D**5F51B3678CAFBDBCADF210BB8626441**56A**

798AC366EED29E3ABD9FB43C9851033A58**0614F0**CBB2DE5C33

ABC77B

Therefore; the extracted key is:

56A3B2D19A67897A650F41602D61E521A04AEC5A

Observe that it is not the same key that was produced by the key selection algorithm. This incorrect key and incorrect ciphered data will be passed to the decryption algorithm which means that it will produce an incorrect data too.

Time requirements

Key extraction in this approach does not require a significant amount of time with the assumption that the master key is available, as it is retrieved at the beginning of the call. The time needed therefore is the same as the time needed to compose the key from the master key by selecting the key segments according to the received key information.

This algorithm was implemented and tested 1,500,000 times on different key lengths and different master key lengths. The results of that test are presented in Table 0-1. A column chart that showing the relationship between length of the master key and the time needed to extract a key is presented in Figure 0-1. It is clear that the time is almost the same in all cases.

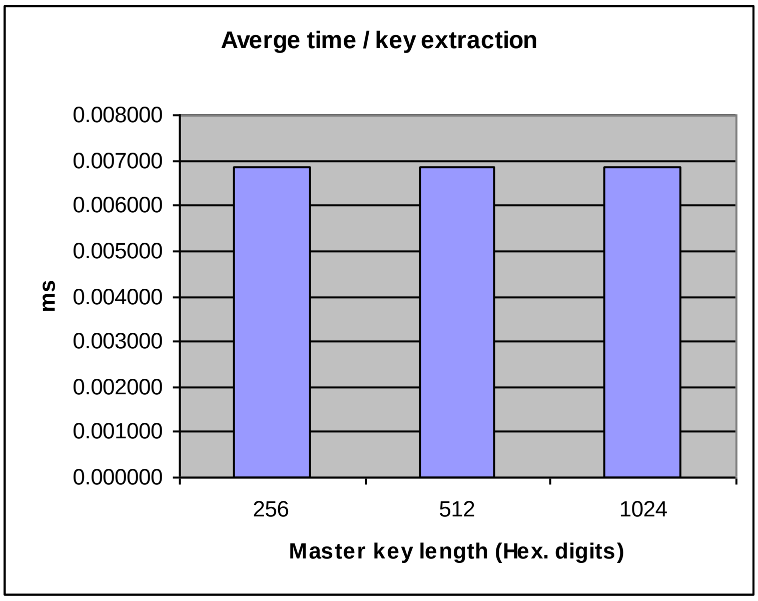


Figure 0-1 Average time / key extraction

Table 0-1 Time requirements for a key extraction

Key Extraction Process (at Receiver Side)												
Master Key Length (Hex. Digits)	**Trial No.**	**Key Length (bits)**							**Total**	**Total time (ms)**	**Average Key Length (bits)**	**Average time/Key (ms)**
		64	**96**	**128**	**160**	**192**	**224**	**256**				
256	1	14125	14435	14308	14219	14440	14290	14183	100000	690.994	160.01	0.006910
	2	14374	14018	14163	14285	14389	14363	14408	100000	690.984	160.33	0.006910
	3	14133	14488	14381	14282	14174	14207	14335	100000	680.022	159.95	0.006800
	4	14234	14279	14407	14428	14284	14276	14092	100000	681.006	159.82	0.006810
	5	14332	14244	14193	14416	14370	14195	14250	100000	688.025	159.95	0.006880
	Sub total	**71198**	**71464**	**71452**	**71630**	**71657**	**71331**	**71268**	**500000**	**3431.030**	**160.01**	**0.006862**
512	1	14416	14242	14323	14155	14225	14225	14414	100000	682.408	159.96	0.006824
	2	14411	14295	14307	14230	14247	14224	14286	100000	691.025	159.82	0.006910
	3	14172	14396	14260	14461	14109	14172	14430	100000	680.976	160.06	0.006810

	4	14419	14427	14210	14181	14389	14099	14275	100000	697.829	159.71	0.006978
	5	14324	14273	14072	14422	14257	14348	14304	100000	678.792	160.09	0.006788
	Sub total	**71742**	**71633**	**71172**	**71449**	**71227**	**71068**	**71709**	**500000**	**3431.031**	**159.92**	**0.006862**
1024	1	14377	14302	14400	14226	14246	14270	14179	100000	686.954	159.74	0.006870
	2	14493	14108	14334	14278	14229	14237	14321	100000	685.367	159.88	0.006854
	3	14447	14171	14325	14382	14155	14251	14269	100000	690.965	159.83	0.006910
	4	14447	14212	14101	14314	14627	14157	14142	100000	684.979	159.84	0.006850
	5	14263	14244	14523	14322	14055	14174	14419	100000	682.767	159.96	0.006828
	Sub total	**72027**	**71037**	**71683**	**71522**	**71312**	**71089**	**71330**	**500000**	**3431.032**	**159.85**	**0.006862**
Totals		**214967**	**214134**	**214307**	**214601**	**214196**	**213488**	**214307**	**1500000**	**10293.093**	**159.93**	**0.006862**

5.3. Decryption

As it was mentioned in the introduction of this chapter, decryption is the inverse of the encryption process. It is aimed to recover the original data. Also, it was mentioned that this decryption is a simple XOR process using the key that is received from the key extraction algorithm which was presented earlier in this chapter. Therefore, the accuracy of this algorithm depends on the received key where a correct key produces a correct results and vise versa.

Algorithm: Decryption

Executed by: Conversation process

Location: Client.

Time: During conversation.

Input: 1) the encryption key;

2) the ciphered voice data.

Output: Original voice data.

Comments: None

Algorithm steps:

```
1. Convert the selected key for this packet to a bits array
2. While not end-of-the-ciphered bits array
     i) get a number of bits equal to the length of the key bits array
     ii) do XOR of this segment of the ciphered bits with the key bits
3. Convert the XORed data to a bytes array
4. Copy the bytes array to a sound buffer
5. Pass the sound buffer to the playback device
```

6. END

Example 5.2: Assuming that the encryption key and the segments of the ciphered data are the same as that used in example 3.4, then, the key is $\boxed{\text{A236BB8DD70FA35D}}$ and the ciphered data is:

```
0100010001100010101000111110011101011101100010000000111011011000

0101110111001001010010001011111011010111111111111010011000011110

0010001010101010110101011000110111101000111100000100110010110100
```

The bits array of the above key is:

```
1010001000110110101110111000110111010111000011111010001101011101
```

And the XOR process of this ciphered data segment with the above key produces the following data:

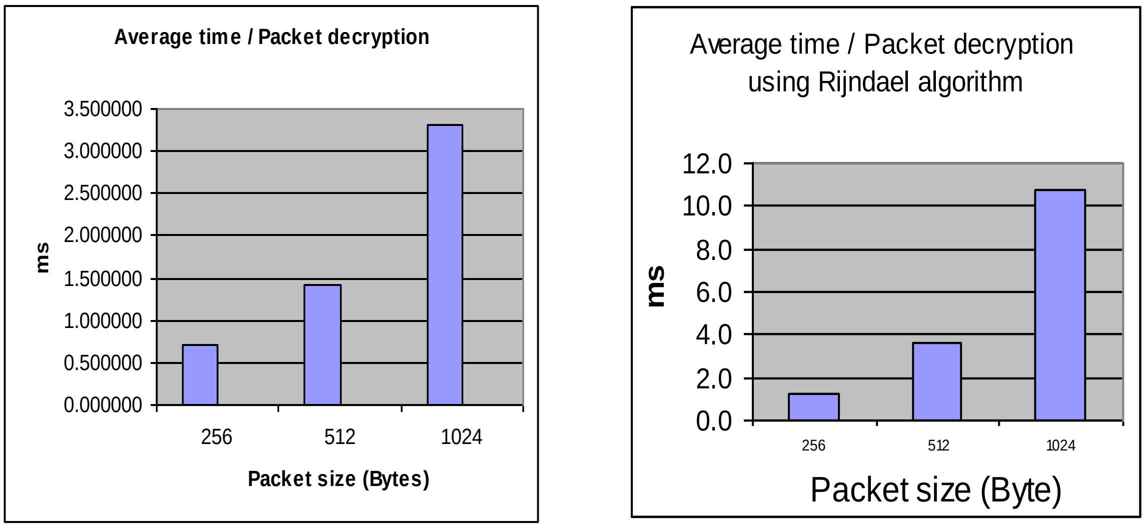
```
1110011001010100000110000110101010001010100001111010110110000101

1111111111111111111100110011001100000000111100000000010101000011

1000000010011100011011100000000000011111111111111110111111101001
```

This data is the same as the original data of example 3.4.

Time requirements

The presented decryption process required less time to decrypt a packet of voice data compared to known decryption algorithms such as Rijndael; so this small amount of added time to the latency is considered to be acceptable in this type of applications. Therefore, the proposed approach improved the security of voice data that travels over the networks and/or Internet.

This algorithm has been implemented and tested to decrypt 1,500,000 packets of different key lengths and different packet sizes. The results of the test are presented in Table 0-2. The AES_Rijndael algorithm was also implemented[18]. This algorithm was used to decrypt 150,000 packets using different key lengths and different packet sizes. The results of that test are presented in Table 0-3. Column charts showing the relationship between the packet size and the decryption time in both cases are presented in Figure 0-2 (a) and (b).



(a) (b)

Figure 0-2 Average time for decryption process

[18] The program was customized from a program taken from the internet. [Obviex]

Table 0-2 Time requirements for decryption process using the proposed solution

Decryption Process												
Packet size (Bytes)	**Trial No.**	**Key Length (bits)**							**Total**	**Total time (ms)**	**Average Key Length (bits)**	**Average time/Packet (ms)**
		64	**96**	**128**	**160**	**192**	**224**	**256**				
256	1	14082	14557	14333	14309	14212	14259	14248	100000	72213.84	159.93	0.722138
	2	14301	14485	14182	14134	14399	14074	14425	100000	71933.44	159.93	0.719334
	3	14295	14320	14161	14187	14175	14413	14449	100000	71903.39	160.21	0.719034
	4	14313	14318	14236	14131	14448	14398	14156	100000	71913.41	159.97	0.719134
	5	14324	14431	14338	14205	14215	14403	14084	100000	71953.46	159.71	0.719535
	Total	**71315**	**72111**	**71250**	**70966**	**71449**	**71547**	**71362**	**500000**	**359917.54**	**159.95**	**0.719835**
512	1	14238	14241	14293	14342	14178	14239	14469	100000	142004.19	160.18	1.420042
	2	14185	14540	14170	14185	14199	14235	14486	100000	142034.24	160.10	1.420342
	3	14467	14274	14221	14096	14090	14349	14503	100000	142404.77	160.04	1.424048

	4	14314	14563	14179	14282	14177	14262	14223	100000	142863.99	159.72	1.428640
	5	14257	14139	14509	14274	14224	14218	14379	100000	141974.15	160.08	1.419742
	Total	**71461**	**71757**	**71372**	**71179**	**70868**	**71303**	**72060**	**500000**	**711281.34**	**160.02**	**1.422563**
1024	1	14080	14241	14394	14309	14394	14141	14441	100000	331676.93	160.28	3.316769
	2	14384	14312	14363	14301	14205	14245	14190	100000	331787.52	159.72	3.317875
	3	14314	14431	14150	14245	14578	14200	14082	100000	331787.09	159.77	3.317871
	4	14386	14309	14259	14327	14128	14300	14291	100000	331857.19	159.86	3.318572
	5	14192	14300	14150	14342	14361	14324	14331	100000	331937.30	160.22	3.319373
	Total	**71356**	**71593**	**71316**	**71524**	**71666**	**71210**	**71335**	**500000**	**1659046**	**159.97**	**3.318092**

Table 0-3 Time requirements for decryption process using Rijndael algorithm

Decryption Process - AES-Rijndael							
Packet size (Bytes)	Trial No.	Key Length (bits)			Total	Total time (ms)	Average time/Packet (ms)
		128	192	256			
256	1	3289	3339	3372	10000	12237	1.224
	2	3345	3345	3310	10000	12148	1.215
	3	3310	3331	3359	10000	11917	1.192
	4	3277	3352	3371	10000	12378	1.238
	5	3379	3298	3323	10000	11807	1.181
	Total	**16600**	**16665**	**16735**	**50000**	**60487**	**1.210**
512	1	3275	3325	3400	10000	36112	3.611
	2	3424	3196	3380	10000	35912	3.591
	3	3234	3421	3345	10000	36162	3.616
	4	3361	3263	3376	10000	36072	3.607
	5	3279	3383	3338	10000	35862	3.586
	Total	**16573**	**16588**	**16839**	**50000**	**180120**	**3.602**
1024	1	3387	3334	3279	10000	106904	10.690
	2	3310	3341	3349	10000	107725	10.773
	3	3309	3367	3324	10000	107174	10.717
	4	3398	3299	3303	10000	107114	10.711
	5	3305	3361	3334	10000	107004	10.700
	Total	**16709**	**16702**	**16589**	**50000**	**535921**	**10.718**

5.4. Conclusions and recommendations

Conclusions

1. The average time needed to extract/compose a key using the received key information is very small (**0.006862 ms**).
2. The length of the master key does not affect the key extraction time; where the master key length does not add an overhead on the process except a negligible amount of time that is due to the management of more memory locations when the master key length is increased.
3. The time needed to extract/compose a key is greater than the time needed to generate the key in the key selection algorithm. The extra overhead comes from converting the

4. bits array of key information to their decimal equivalent values. Again this difference would not affect the overall process; since it is the average time for the total operation that is of concern to us, and should be kept to a minimum value.
5. The average time needed to decrypt a packet in the proposed algorithm is much less than that of Rijndael algorithm. ***The ratio is about 1:10 for a packet of size 1024 bytes.***
6. The average time needed to decrypt a packet in the proposed algorithm is relatively small and hence it is acceptable; It is around 3.2252 ms for a packet of size 1024 bytes. While it is a round 10.718 ms in Rijndael algorithm for the same packet size which is considered to be big for this type of applications.
7. The packet size affects the decryption time in both cases. In the proposed algorithm, if the packet size is doubled, then the average time needed to decrypt a packet is almost doubled too; this is expected since the process manipulates each bit of the data. In Rijndael algorithm, if the packet size is doubled, then the average time needed to decrypt a packet becomes three multiples.

Recommendations

1. Because of the conclusion in point 6 above, it is recommended to minimize the packet size in order to minimize the time needed for the decryption process; keeping in mind a trade off with the overhead that comes from the headers added to each packet.
2. Since the known encryption algorithms require a significant amount of time as presented in points 4 and 5 of the above conclusions, it is recommended to apply a simple algorithm with minimum computations.

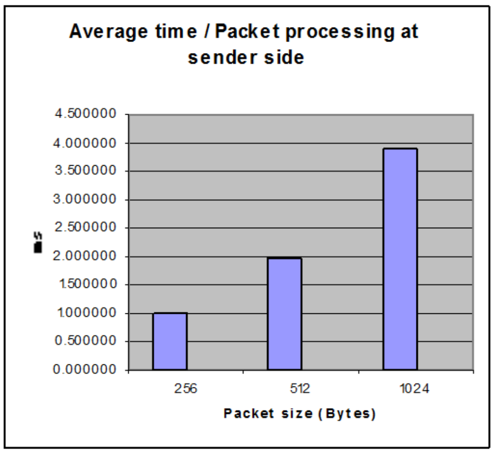
Evaluation of the security process

6.1. Introduction

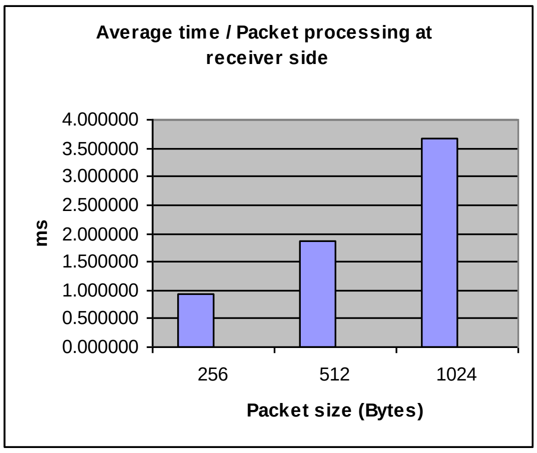
Chapters 3, 4, and 5 introduced the required analysis of the implementation and the tests carried out for the different stages/steps of the overall security process: sender side (key selection, voice data encryption, and mixing key information into the encrypted data) and receiver side (separation of key information from the ciphered data, key extraction, and decryption of the ciphered data). As the goal of this research is an overall process, the average time needed for the overall process is of concern to us, and should be kept to a minimum value. In this chapter all steps of the security process are set together, the average time of the overall process in this approach is present, a comparison between the average time needed for total steps at the sender side and the average time needed for total steps at the receiver side is present too. The success of any security process depends on its strength, key management and distribution, and performance; therefore an in-depth analysis and evaluation of the preceding algorithms is presented in the third section of this chapter. The analysis shows that this approach is simple, strong, and need small amount of time; therefore it is strongly recommended to be used with VoIP.

6.2. Time Requirements for the Overall Process

Table 0-1 summaries the average time for the individual algorithms, subtotal time at sender and receiver sides, and the total time of the overall process; Table 0-2 shows the time needed for all operations at the sender side when executed as one algorithm; Table 0-3 shows the time needed for all operations at the receiver side when executed as one algorithm. Column charts in Figure 0-1 showing the relationship between the packet size and the average time for the total operations at both sender and receiver sides.



(a)



(b)

Figure 0-1 Average time /packet process at sender and receiver sides

Table 0-1 Summary of time requirments for all individual security processes

Time needed by variety processes for a voice packet					**Average time (ms) /Byte**
Process ⇓	**Packet Size (Bytes) ⇒**	**Average time (ms)**			
		256	**512**	**1024**	
Key Selection / Key		0.002604	0.002604	0.002604	5.93E-06
Encryption / Packet		0.307242	0.6000228	1.1868466	0.001177
Mixing Data and Key Information / Packet		0.673148	1.3374431	2.6688977	0.002616
Splitting Data and Key Information / Packet		0.183124	0.3643041	0.7272658	0.000712
Key Extraction (Key)		0.006862	0.006862	0.006862	1.56E-05
Decryption (Packet)		0.719835	1.4225627	3.3180921	0.002944
Sub total (Sender Side)		**0.982993**	**1.940070**	**3.858348**	**0.003799**
Sub total (Receiver Side)		**0.909821**	**1.793729**	**4.052220**	**0.003672**
Total		**1.892814**	**3.733798**	**7.910568**	**0.007471**

Table 0-2 Time requirments for all processes at the sender side as one unit

Processes at Sender Side[19]												
Packet Length (Bytes)	**Trial No.**	**Key Length (bits)**							**Total**	**Total time (ms)**	**Average Key Length (bits)**	**Average time/Packet (ms)**
		64	**96**	**128**	**160**	**192**	**224**	**256**				
256	1	14264	14368	14174	14148	14346	14240	14460	100000	99233	160.16	0.992330
	2	14258	14104	14436	14308	14270	14290	14334	100000	99303	160.14	0.993030
	3	14221	14292	14231	14119	14368	14366	14403	100000	98973	160.27	0.989730
	4	14402	14339	14264	14308	14257	14128	14302	100000	99223	159.77	0.992230
	5	14157	14219	14338	14428	14226	14301	14331	100000	99193	160.18	0.991930
	Total	**71302**	**71322**	**71443**	**71311**	**71467**	**71325**	**71830**	**500000**	**495925**	**160.10**	**0.991850**
512	1	14544	14320	14129	14253	14053	14276	14425	100000	195731	159.83	1.957310
	2	14259	14272	14312	14349	14359	14258	14191	100000	195350	159.94	1.953500
	3	14404	14070	14265	14154	14425	14291	14391	100000	196572	160.18	1.965720
	4	14140	14335	14292	14210	14120	14195	14708	100000	195631	160.40	1.956310

[19] The test has been applied using a master key of size 256 hexadecimal digits.

	5	14262	14385	14338	14162	14286	14103	14464	100000	195671	160.00	1.956710
	Total	**71609**	**71382**	**71336**	**71128**	**71243**	**71123**	**72179**	**500000**	**978955**	**160.07**	**1.957910**
1024	1	14197	14070	14480	14364	14322	14205	14362	100000	387998	160.19	3.879980
	2	14291	14052	14334	14203	14606	14201	14313	100000	388419	160.20	3.884190
	3	14327	14481	14121	14142	14258	14332	14339	100000	387998	159.96	3.879980
	4	14305	14460	14150	14145	14392	14165	14383	100000	388468	159.96	3.884680
	5	14261	14284	14374	14109	14252	14458	14262	100000	388229	160.07	3.882290
	Total	**71381**	**71347**	**71459**	**70963**	**71830**	**71361**	**71659**	**500000**	**1941112**	**160.08**	**3.882224**

Table 0-3 Time requirements for all processes at the receiver side as one unit

Processes at Receiver Side[20]												
Packet Length (Bytes(	**Trial No.**	**Key Length (bits)**							**Total**	**Total time (ms)**	**Average Key Length (bits)**	**Average time/Packet (ms)**
		64	**96**	**128**	**160**	**192**	**224**	**256**				
256	1	14357	14357	14255	14200	14131	14421	14279	100000	93344	159.93	0.933440
	2	14305	14378	14227	14023	14284	14345	14438	100000	93344	160.12	0.933440
	3	14202	14332	13940	14433	14522	14384	14187	100000	92934	160.21	0.929340
	4	14228	13997	14242	14430	14266	14506	14331	100000	93214	160.43	0.932140
	5	14265	14194	14183	14394	14410	14280	14274	100000	93434	160.14	0.934340
	Total	**71357**	**71258**	**70847**	**71480**	**71613**	**71936**	**71509**	**500000**	**466270**	**160.16**	**0.932540**
512	1	14289	14328	14297	14204	14406	14217	14259	100000	184475	159.94	1.844750
	2	14431	14278	14163	14273	14090	14351	14414	100000	184055	160.01	1.840550
	3	14337	14152	14536	14262	14270	14255	14188	100000	184045	159.84	1.840450
	4	14232	14171	14219	14329	14365	14379	14305	100000	187740	160.25	1.877400
	5	14306	14186	14540	14278	14200	14128	14362	100000	186428	159.91	1.864280

[20] The test has been applied using a master key of size 256 hexadecimal digits.

	Total	**71595**	**71115**	**71755**	**71346**	**71331**	**71330**	**71528**	**500000**	**926743**	**15999.**	**1.853486**
1024	1	14257	14217	14313	14194	14179	14408	14432	100000	365625	16025.	3.656250
	2	14377	14144	14292	14309	14263	14474	14141	100000	365656	15998.	3.656560
	3	14458	14373	14193	14283	14298	14338	14057	100000	365586	15963.	3.655860
	4	14190	14423	14260	14441	14076	14446	14164	100000	365926	15993.	3.659260
	5	14360	14233	14260	14241	14288	14372	14246	100000	365576	15999.	3.655760
	Total	**71642**	**71390**	**71318**	**71468**	**71104**	**72038**	**71040**	**500000**	**1828369**	**15995.**	**3.656738**

6.3. Evaluation of the Security Process

Assuming that:

Number of users on the system is **NU**

Length of master key for each user is **MKL** hexadecimal digits

Number of key length categories is **NOKLC**

Number of key segments categories is **NOKSC**

Number of directions for key selection is **NODFKS** = 2 (LTR and RTL).

Number of directions for key segment selection is **NODFKSS** = 2 (LTR and RTL).

Number of key segment lengths categories is **NOKSLC**[21].

Number of possible keys that can be generated from each master key is **NOPK**.

Total number of possible keys that can be generated from all master keys is **TNOPK**.

Two possible scenarios for the location of master keys are:

Scenario 1: they are stored on each client.

Scenario 2: they are stored on the server only.

For the first scenario and according to the above parameters, the number of possible keys

[21] See Table 0-1.

that can be generated from any master key is computed by Equation 0.1.

$$NOPK = MKL \times NOKLC \times NOKSC \times NODFKS \times NODFKSS \times NOKSLC$$

$$= MKL \times NOKLC \times NOKSC \times 2 \times 2 \times NOKSLC \quad \ldots\ldots\ldots\ldots.. \quad \textbf{Equation 0.1}$$

While for the second scenario, the total number of possible keys that can be generated from all master keys should be computed; and hence, the number of users should be added to the parameters of Equation 0.1, so it becomes as given in Equation 0.2.

$$TNOPK = NU \times MKL \times NOKLC \times NOKSC \times NODFKS \times NODFKSS \times NOKSLC$$

$$= NU \times MKL \times NOKLC \times NOKSC \times 2 \times 2 \times NOKSLC \quad \ldots\ldots\ldots\ldots \quad \textbf{Equation 0.2}$$

This means that a hacker should try NOPK for the first scenario and TNOPK for the second scenario in order to identify a key of one packet. *This is with the assumption that he has the key extracted correctly out of the transmitted data*.

Example 6.1: Assume that the values of the above parameters are as follows:

NU = 100 users

MKL = 256 hexadecimal digits

NOKLC = 7

NOKSC = 4

NOKSLC = 4

Then, Equation 0.1 deduces the following:

$NOPK = 256 \times 7 \times 4 \times 2 \times 2 \times 4 = 114{,}688$ keys

And from Equation 0.2:

$TNOPK = 100 \times 256 \times 7 \times 4 \times 2 \times 2 \times 4 = 11{,}468{,}800$ keys

The above number can be increased by increasing any of the above parameters. As was mentioned earlier in the key selection section, the length of the master key doesn't add any significant amount of time on the key generation and the key extraction processes;

therefore it can be increased enough to increase the number of possible keys. It was also mentioned in the encryption process that the encryption and decryption algorithms work on keys of any size; therefore it is possible to increase the range of key size too in order to increase the number of possible keys. The same thing is also true for NU, NOKSC, and NOKSLC too.

Example 6.2: Assuming that the values of the above parameters are as follows:

NU = 1000 users

MKL = 2048 hexadecimal digits

NOKLC = 32

NOKSC = 6

NOKSLC = 4

Then, from Equation 0.1 gives:

NOPK = $2048 \times 32 \times 6 \times 2 \times 2 \times 4 = 6{,}291{,}456$ keys

And from Equation 0.2 produces:

TNOPK = $1000 \times 2048 \times 32 \times 6 \times 2 \times 2 \times 4 = 6{,}291{,}456{,}000$ keys

Voice packets carry few milliseconds of voice data (a maximum of 30 ms in the case of PCM). This means that if a single packet of voice data is decrypted by any mean by an unauthorized user it will be useless. For a hacker to get an understandable segment of speech, he needs at least ***t seconds*** of voice data. The minimum number of consecutive voice packets that should be decrypted correctly in order to get an understandable or useful data (NOPFUV) is given in Equation 0.3.

NOPFUV = $t \times 1000 / 30$ **Equation 0.3**

…………………………………………...

Therefore; an unauthorized user must try a huge number of permutations (NOPER) to get any useful data. Equations 6.4 and 6.5 show this for the two given scenarios.

Scenario 1:

$\text{NOPER} = (\text{NOPK})^{\text{CEILING(NOPFUV)}}$ **Equation 0.4**

……………………………………

Scenario 2:

$\text{NOPER} = (\text{TNOPK})^{\text{CEILING(NOPFUV)}}$ **Equation 0.5**

………………………………..

The total time (**T**) needed to try the NOPER given in Equations 6.4 and 6.5 above depends on the computer speed; if the computer system manipulates **n** permutations per unit of time; then the total time (**T**) is given by Equation 0.6.

$T = (\text{NOPER}/n)$ units of time **Equation 0.6**

…………………………………………

If it is assumed that 0.25 second creates an understandable segment of voice data, then Equations 6.3, 6.4, and 6.5 produce:

$\text{NOPFUV} = 0.25\times 1000 / 30 = 8.33$ packets; ceiling(8.33) = 9.

Scenario 1:

$\text{NOPER} = (\text{NOPK})^{\text{ceiling}(8.33)} = (\text{NOPK})^9$

Scenario 2:

$\text{NOPER} = (\text{TNOPK})^{\text{ceiling}(8.33)} = (\text{TNOPK})^9$

Table 0-4 illustrates the number of possible keys, number of permutations needed to identify a voice segment using brute-force attack, and the time needed in years to identify a voice segment by an unauthorized user for variety of factor combinations. Table 0-5 illustrates the same for the second scenario. These facts are shown in Figures 6-2, 6-3, and 6-4; each figure has two column charts: one for each of the two scenarios. A direct relationship is very clear between the number of possible keys and different factors as shown in Figure 0-2; also a direct relationship is very clear too between the number of permutations and different factors as shown in Figure 0-3; the same is also true for the required time needed to identify a voice message using brute-force attack as shown in Figure 0-4.

Table 0-4 Time complexity of the security system for scenario 1

System security - Time complexity							
						Time (years)	
MKL	NOKLC	NOKSC	NOKSLC	NOPK[22]	NOPER[23]	1 Permut./ µs	10,000,000 Permut./µs
256	1	1	1	1024	1.24E+27	3.93E+13	3.93E+06
			4	4096	3.25E+32	1.03E+19	1.03E+12
		5	1	5120	2.42E+33	7.67E+19	7.67E+12
			4	20480	6.34E+38	2.01E+25	2.01E+18
	8	1	1	8192	1.66E+35	5.27E+21	5.27E+14
			4	32768	4.36E+40	1.38E+27	1.38E+20
		5	1	40960	3.25E+41	1.03E+28	1.03E+21
			4	163840	8.51E+46	2.70E+33	2.70E+26
	32	1	1	32768	4.36E+40	1.38E+27	1.38E+20
			4	131072	1.14E+46	3.62E+32	3.62E+25
		5	1	163840	8.51E+46	2.70E+33	2.70E+26
			4	655360	2.23E+52	7.07E+38	7.07E+31
512	1	1	1	2048	6.34E+29	2.01E+16	2.01E+09
			4	8192	1.66E+35	5.27E+21	5.27E+14
		5	1	10240	1.24E+36	3.93E+22	3.93E+15
			4	40960	3.25E+41	1.03E+28	1.03E+21
	8	1	1	16384	8.51E+37	2.70E+24	2.70E+17
			4	65536	2.23E+43	7.07E+29	7.07E+22
		5	1	81920	1.66E+44	5.27E+30	5.27E+23
			4	327680	4.36E+49	1.38E+36	1.38E+29
	32	1	1	65536	2.23E+43	7.07E+29	7.07E+22
			4	262144	5.85E+48	1.85E+35	1.85E+28
		5	1	327680	4.36E+49	1.38E+36	1.38E+29
			4	1E+06	1.14E+55	3.62E+41	3.62E+34
1024	1	1	1	4096	3.25E+32	1.03E+19	1.03E+12
			4	16384	8.51E+37	2.70E+24	2.70E+17
		5	1	20480	6.34E+38	2.01E+25	2.01E+18
			4	81920	1.66E+44	5.27E+30	5.27E+23

[22] See Equation 0.1
[23] See Equation 0.4

	8	1	1	32768	4.36E+40	1.38E+27	1.38E+20
			4	131072	1.14E+46	3.62E+32	3.62E+25
		5	1	163840	8.51E+46	2.70E+33	2.70E+26
			4	655360	2.23E+52	7.07E+38	7.07E+31
	32	1	1	131072	1.14E+46	3.62E+32	3.62E+25
			4	524288	2.99E+51	9.49E+37	9.49E+30
		5	1	655360	2.23E+52	7.07E+38	7.07E+31
			4	3E+06	5.85E+57	1.85E+44	1.85E+37

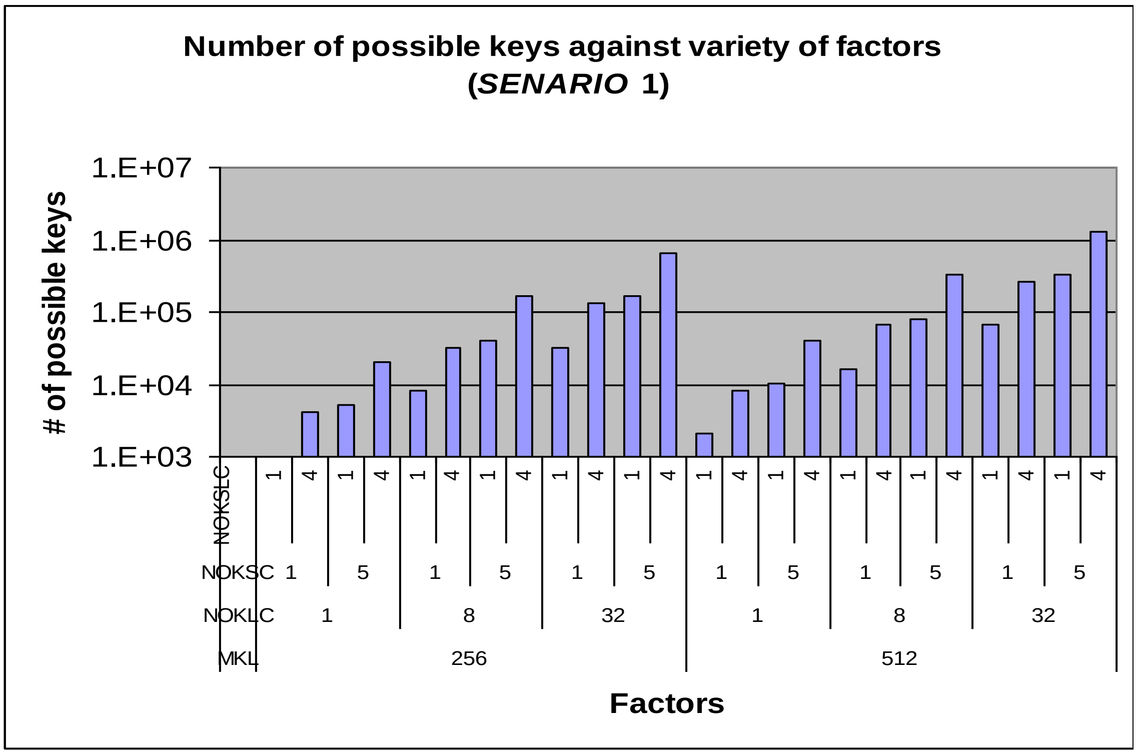
Table 0-5 Time complexity of the security system for scenario 2

System security - Time complexity								
NU	MKL	NOKLC	NOKSC	NOKSLC	TNOPK[24]	NOPER[25]	Time (years)	
							1 Permut./μs	10,000,000 Permut./μs
100	256	1	1	1	1.02E+05	1.24E+45	3.93E+31	3.93E+24
				4	4.10E+05	3.25E+50	1.03E+37	1.03E+30
			5	1	5.12E+05	2.42E+51	7.67E+37	7.67E+30
				4	2.05E+06	6.34E+56	2.01E+43	2.01E+36
		8	1	1	8.19E+05	1.66E+53	5.27E+39	5.27E+32
				4	3.28E+06	4.36E+58	1.38E+45	1.38E+38
			5	1	4.10E+06	3.25E+59	1.03E+46	1.03E+39
				4	1.64E+07	8.51E+64	2.70E+51	2.70E+44
		32	1	1	3.28E+06	4.36E+58	1.38E+45	1.38E+38
				4	1.31E+07	1.14E+64	3.62E+50	3.62E+43
			5	1	1.64E+07	8.51E+64	2.70E+51	2.70E+44
				4	6.55E+07	2.23E+70	7.07E+56	7.07E+49
	512	1	1	1	2.05E+05	6.34E+47	2.01E+34	2.01E+27
				4	8.19E+05	1.66E+53	5.27E+39	5.27E+32
			5	1	1.02E+06	1.24E+54	3.93E+40	3.93E+33
				4	4.10E+06	3.25E+59	1.03E+46	1.03E+39
		8	1	1	1.64E+06	8.51E+55	2.70E+42	2.70E+35
				4	6.55E+06	2.23E+61	7.07E+47	7.07E+40
			5	1	8.19E+06	1.66E+62	5.27E+48	5.27E+41
				4	3.28E+07	4.36E+67	1.38E+54	1.38E+47
		32	1	1	6.55E+06	2.23E+61	7.07E+47	7.07E+40
				4	2.62E+07	5.85E+66	1.85E+53	1.85E+46
			5	1	3.28E+07	4.36E+67	1.38E+54	1.38E+47
				4	1.31E+08	1.14E+73	3.62E+59	3.62E+52
	1024	1	1	1	4.10E+05	3.25E+50	1.03E+37	1.03E+30
				4	1.64E+06	8.51E+55	2.70E+42	2.70E+35
			5	1	2.05E+06	6.34E+56	2.01E+43	2.01E+36
				4	8.19E+06	1.66E+62	5.27E+48	5.27E+41
		8	1	1	3.28E+06	4.36E+58	1.38E+45	1.38E+38
				4	1.31E+07	1.14E+64	3.62E+50	3.62E+43

[24] See Equation 0.2
[25] See Equation 0.5

			5	1	1.64E+07	8.51E+64	2.70E+51	2.70E+44
				4	6.55E+07	2.23E+70	7.07E+56	7.07E+49
		32	1	1	1.31E+07	1.14E+64	3.62E+50	3.62E+43
				4	5.24E+07	2.99E+69	9.49E+55	9.49E+48
			5	1	6.55E+07	2.23E+70	7.07E+56	7.07E+49
				4	2.62E+08	5.85E+75	1.85E+62	1.85E+55

NU	MKL	NOKLC	NOKSC	NOKSLC	NOPK	NOPER	Time (years)	
							1 Permut./μs	10,000,000 Permut./μs
1000	256	1	1	1	1.02E+06	1.24E+54	3.93E+40	3.93E+33
				4	4.10E+06	3.25E+59	1.03E+46	1.03E+39
			5	1	5.12E+06	2.42E+60	7.67E+46	7.67E+39
				4	2.05E+07	6.34E+65	2.01E+52	2.01E+45
		8	1	1	8.19E+06	1.66E+62	5.27E+48	5.27E+41
				4	3.28E+07	4.36E+67	1.38E+54	1.38E+47
			5	1	4.10E+07	3.25E+68	1.03E+55	1.03E+48
				4	1.64E+08	8.51E+73	2.70E+60	2.70E+53
		32	1	1	3.28E+07	4.36E+67	1.38E+54	1.38E+47
				4	1.31E+08	1.14E+73	3.62E+59	3.62E+52
			5	1	1.64E+08	8.51E+73	2.70E+60	2.70E+53
				4	6.55E+08	2.23E+79	7.07E+65	7.07E+58
	512	1	1	1	2.05E+06	6.34E+56	2.01E+43	2.01E+36
				4	8.19E+06	1.66E+62	5.27E+48	5.27E+41
			5	1	1.02E+07	1.24E+63	3.93E+49	3.93E+42
				4	4.10E+07	3.25E+68	1.03E+55	1.03E+48
		8	1	1	1.64E+07	8.51E+64	2.70E+51	2.70E+44
				4	6.55E+07	2.23E+70	7.07E+56	7.07E+49
			5	1	8.19E+07	1.66E+71	5.27E+57	5.27E+50
				4	3.28E+08	4.36E+76	1.38E+63	1.38E+56
		32	1	1	6.55E+07	2.23E+70	7.07E+56	7.07E+49
				4	2.62E+08	5.85E+75	1.85E+62	1.85E+55
			5	1	3.28E+08	4.36E+76	1.38E+63	1.38E+56
				4	1.31E+09	1.14E+82	3.62E+68	3.62E+61
	1024	1	1	1	4.10E+06	3.25E+59	1.03E+46	1.03E+39
				4	1.64E+07	8.51E+64	2.70E+51	2.70E+44
			5	1	2.05E+07	6.34E+65	2.01E+52	2.01E+45
				4	8.19E+07	1.66E+71	5.27E+57	5.27E+50
		8	1	1	3.28E+07	4.36E+67	1.38E+54	1.38E+47
				4	1.31E+08	1.14E+73	3.62E+59	3.62E+52
			5	1	1.64E+08	8.51E+73	2.70E+60	2.70E+53
				4	6.55E+08	2.23E+79	7.07E+65	7.07E+58
		32	1	1	1.31E+08	1.14E+73	3.62E+59	3.62E+52
				4	5.24E+08	2.99E+78	9.49E+64	9.49E+57
			5	1	6.55E+08	2.23E+79	7.07E+65	7.07E+58
				4	2.62E+09	5.85E+84	1.85E+71	1.85E+64



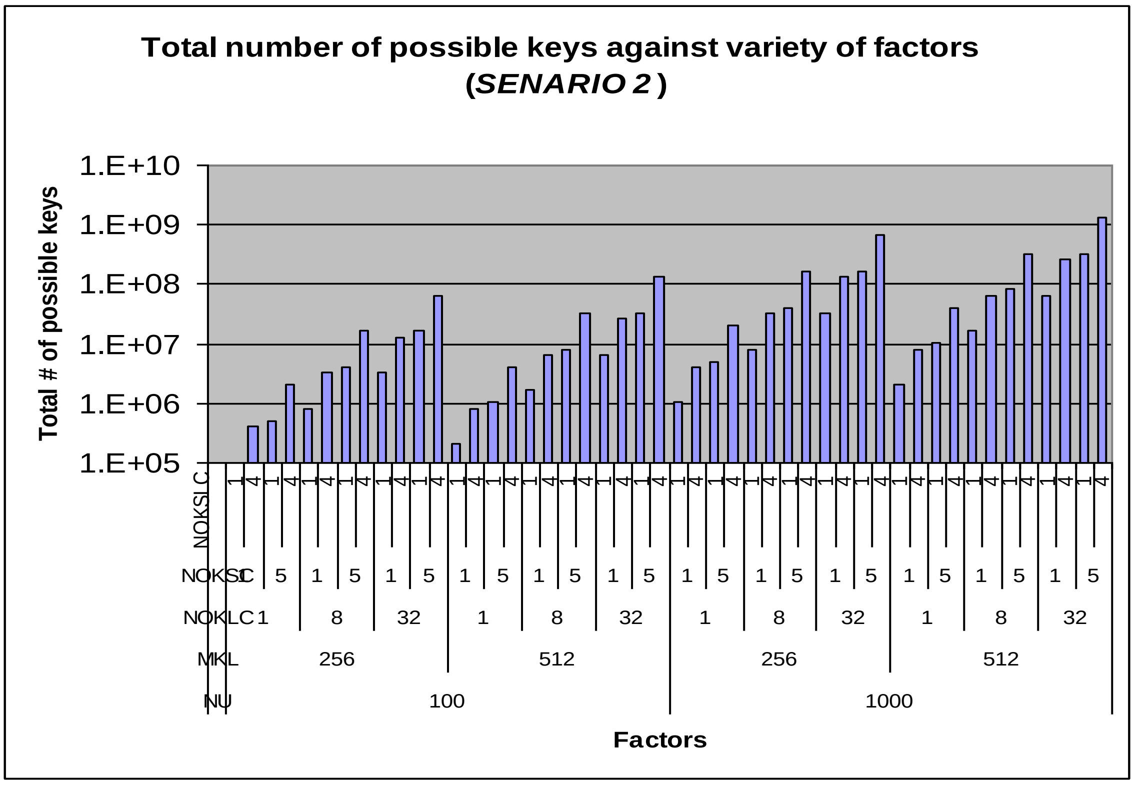
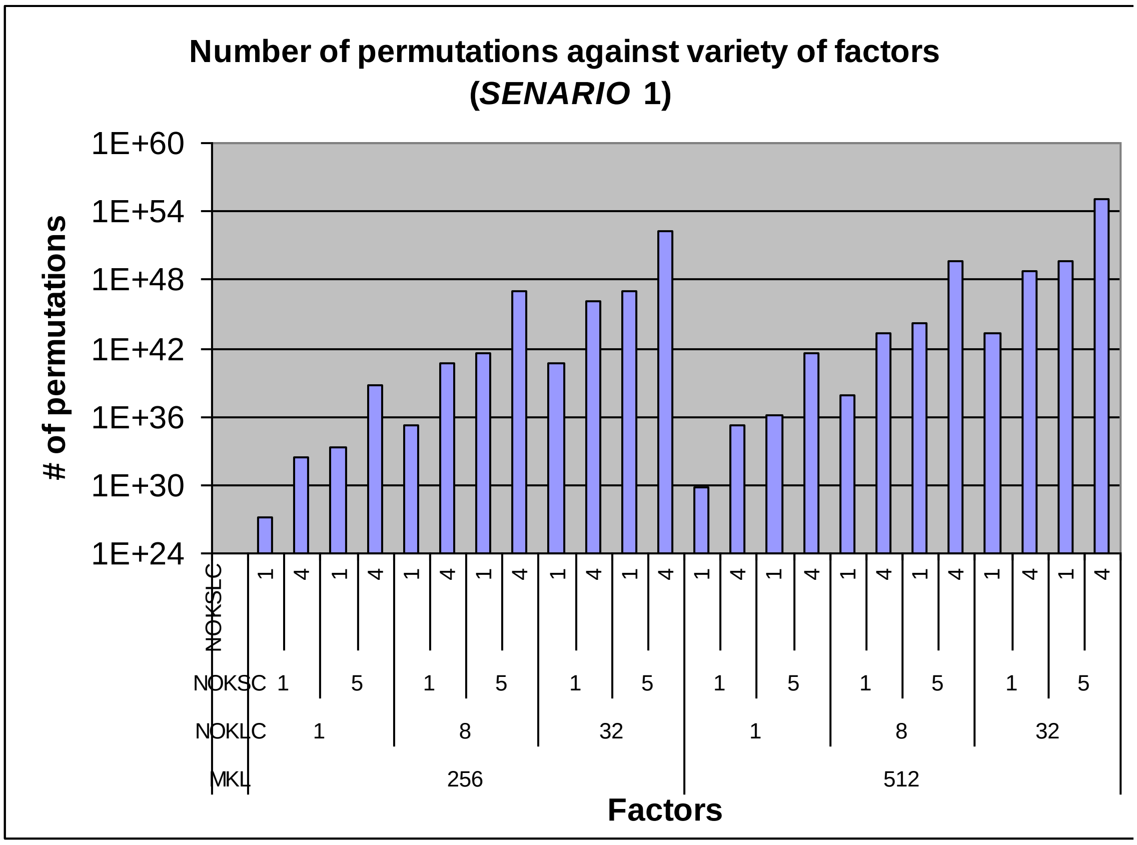


Figure 0-2 Number of possible keys that can be generated from master key(s) for two scenarios



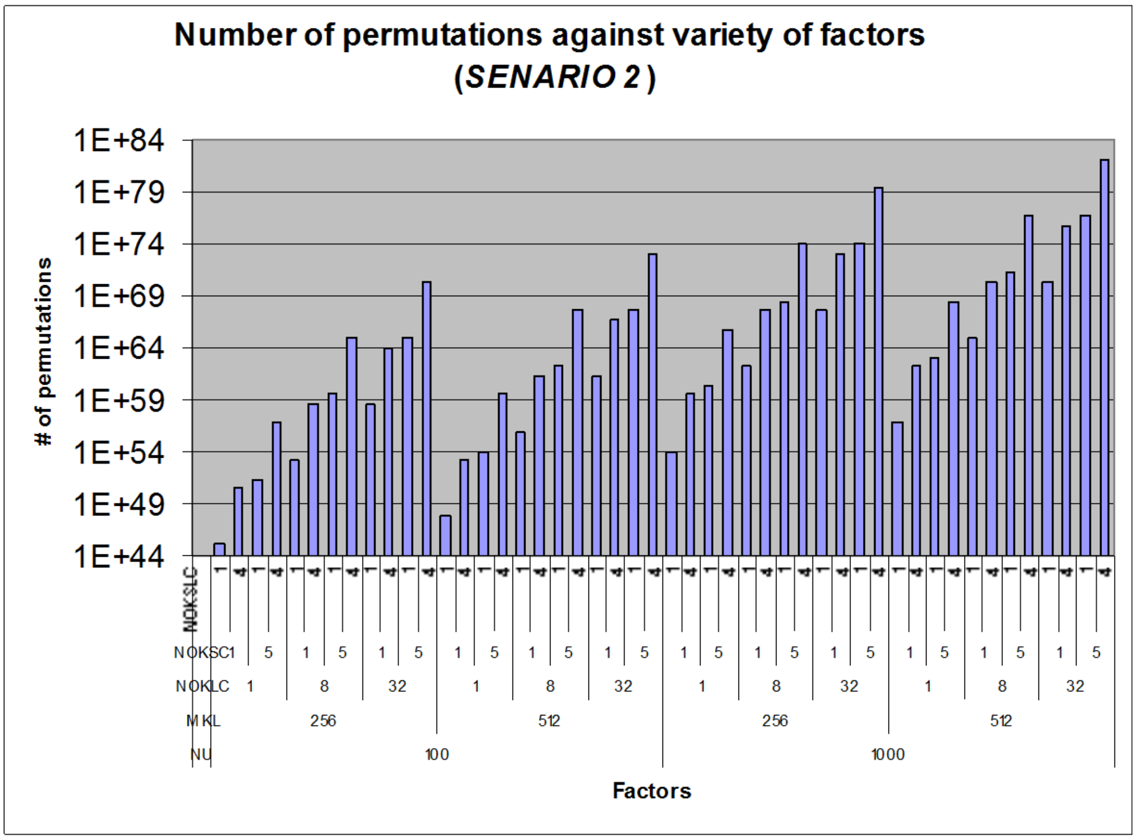
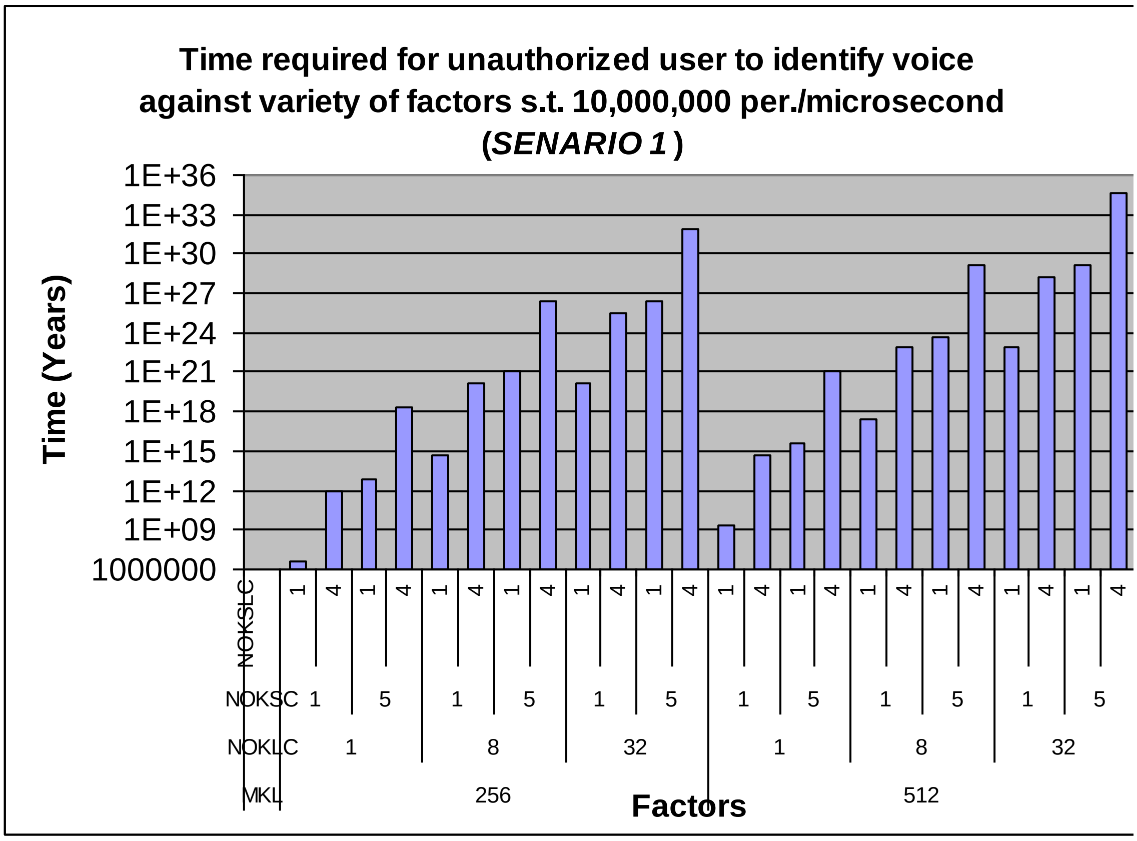


Figure 0-3 Number of permutations required to identify a voice segment using brute-force attack



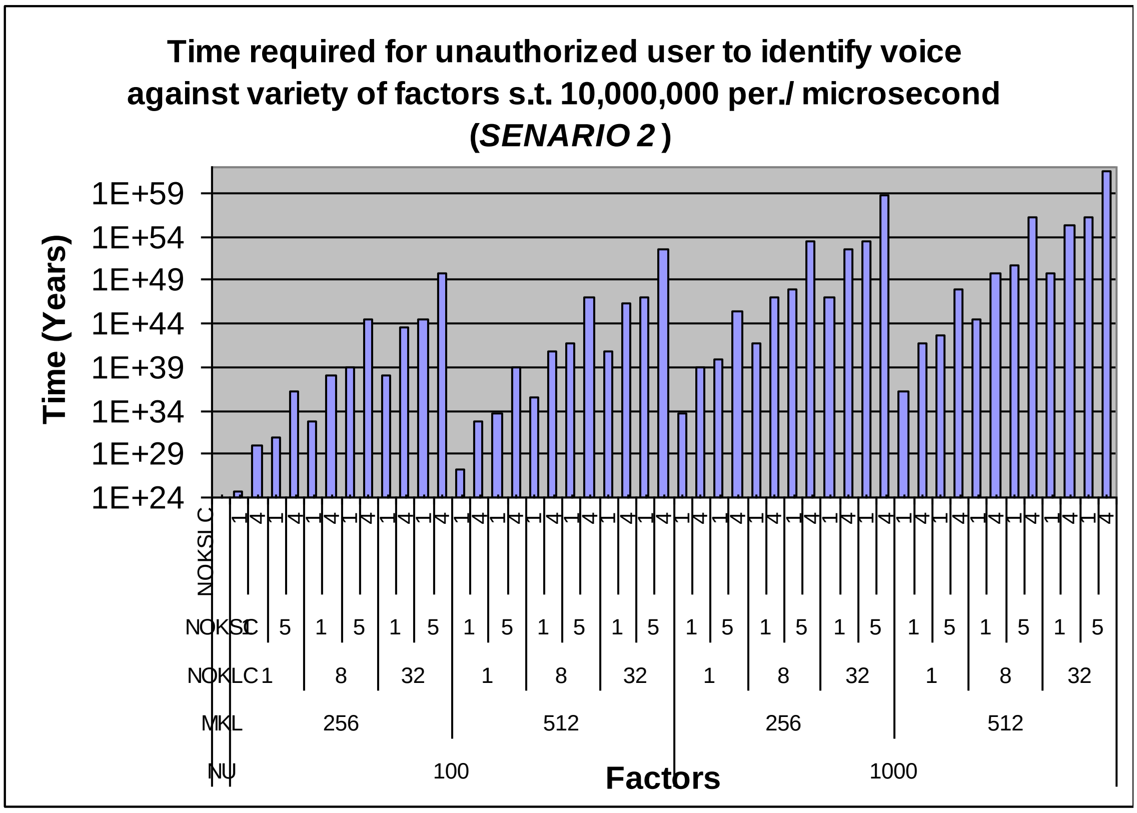


Figure 0-4 Time needed from an unauthorized user to identify a voice segment in two scenarios s.t. a computer processes 10,000,000 permutations/microsecond

6.4 VoIP Delay Budget

This section includes some examples that illustrate the end-to-end delay budget of voice application for various network configurations. The end-to-end delay is the summation of the fixed and variable delays that were discussed in chapter two. The network designer should take these delays into account. Security process delay that was computed in the preceding chapters should be added to the delay budget. In this research, it is found – as shown in Table 0-1 - that the average time needed for the security processes per byte of the voice data is around **0.00747 ms**; this value will be used in the coming calculations of the security delay.

Vocoder		Data rate (kbps)	30 ms voice data	
Coder	Reference		Frame Size (bits)	Security process delay (ms)
PCM	G.711	64	1920	1.7928
ADPCM	G.726	40	1200	1.1205
ADPCM	G.726	32	960	0.8964
ADPCM	G.726	24	720	0.6723
ADPCM	G.726	16	480	0.4482
CS-ACELP	G.729AB	8	240	0.2241
MP-MLQ	G.723.1m	6.3	189	0.176479
ACELP	G.723.1a	5.3	159	0.148466

Table 0-6 illustrates security processes delay for a variety of voice codecs with the assumption that 30 ms of voice data is processed. There exists a direct relationship between the codec data rate and the time required to carry out the security process on its data as shown in Figure 0-5. It is clear from

Vocoder		**Data rate (kbps)**	**30 ms voice data**	
			Frame Size (bits)	**Security process delay (ms)**
Coder	**Reference**			
PCM	G.711	64	1920	1.7928
ADPCM	G.726	40	1200	1.1205
ADPCM	G.726	32	960	0.8964
ADPCM	G.726	24	720	0.6723
ADPCM	G.726	16	480	0.4482
CS-ACELP	G.729AB	8	240	0.2241
MP-MLQ	G.723.1m	6.3	189	0.176479
ACELP	G.723.1a	5.3	159	0.148466

Table 0-6 that the time required to carry out the security process is small even with the PCM coder which does not do compression. Other codecs have less delay values but they have much greater values for codec delays as illustrated in Table 0-5. In order to decide which codec is better to be used, all delay sources should be considered together. There is no single choice that is suitable for all applications/environments.

Vocoder		Data rate (kbps)	30 ms voice data	
			Frame Size (bits)[26]	Security process delay (ms)[27]
Coder	**Reference**			
PCM	G.711	64	1920	1.7928
ADPCM	G.726	40	1200	1.1205
ADPCM	G.726	32	960	0.8964
ADPCM	G.726	24	720	0.6723
ADPCM	G.726	16	480	0.4482
CS-ACELP	G.729AB	8	240	0.2241
MP-MLQ	G.723.1m	6.3	189	0.176479
ACELP	G.723.1a	5.3	159	0.148466

Table 0-6 Security process delay of 30 ms voice data for variety codecs

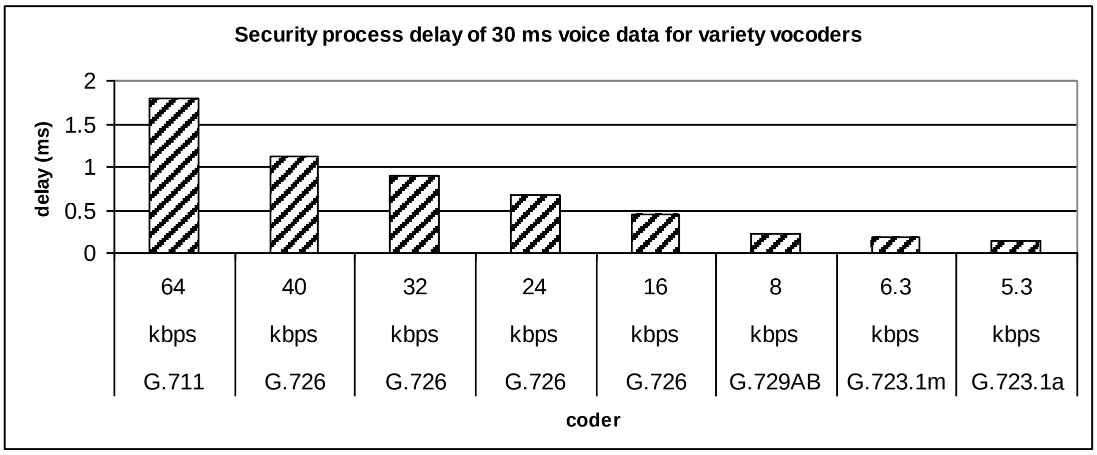


Figure 0-5 Delay caused by security process for variety of codecs

It is necessary to add headers overhead to the packet size for any delay computation; this is true whenever the packet size is a factor of the computational formula. Table 0-7 illustrates packet size and its overhead for a variety voice codecs. An inverse relationship exists between voice data size and the headers overhead as shown in Figure 0-6. Headers overhead exceeds 200% of the data size as coder data rate becomes 8 kbps or less. The column chart in Figure 0-7 shows a comparison between packet size and its header overhead for a variety of codecs.

[26] Data Rate(kbps) × 30
[27] Size(bits) / 8 × 0.00747 ms

Table 0-7 Voice packet size and header overhead

Vocoder			30 ms voice packet			
Coder	Reference	Data rate (kbps)	Frame Size (bits)	Header overhead (bits)[28]	Total packet size (bits)[29]	header tax[30]
PCM	G.711	64	1920	488	2408	25.4%
ADPCM	G.726	40	1200	488	1688	40.7%
ADPCM	G.726	32	960	488	1448	50.8%
ADPCM	G.726	24	720	488	1208	67.8%
ADPCM	G.726	16	480	488	968	101.7%
CS-ACELP	G.729AB	8	240	488	728	203.3%
MP-MLQ	G.723.1m	6.3	189	488	677	258.2%
ACELP	G.723.1a	5.3	159	488	647	306.9%

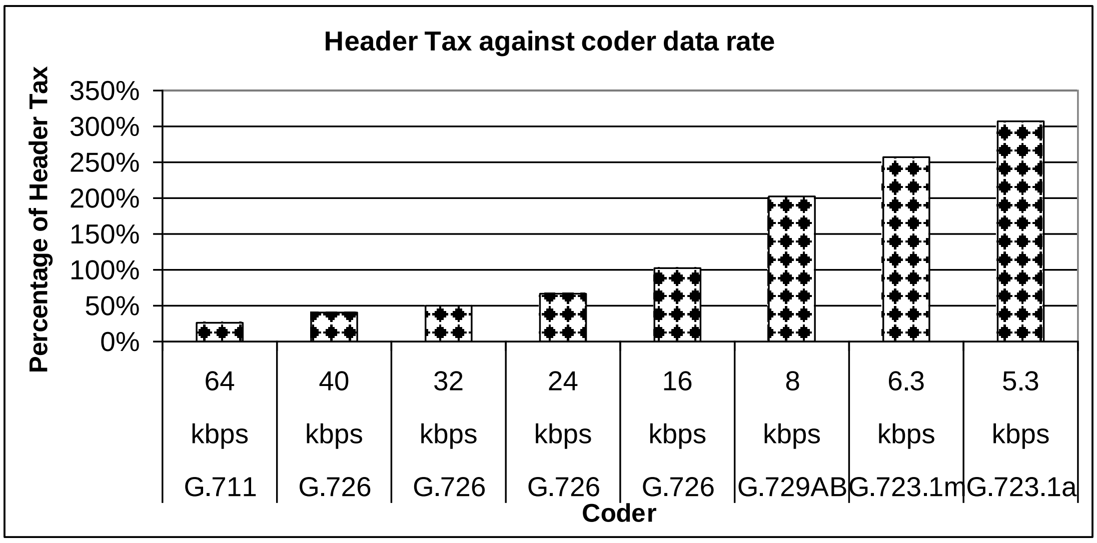


Figure 0-6 Header tax against data rate for variety of codecs

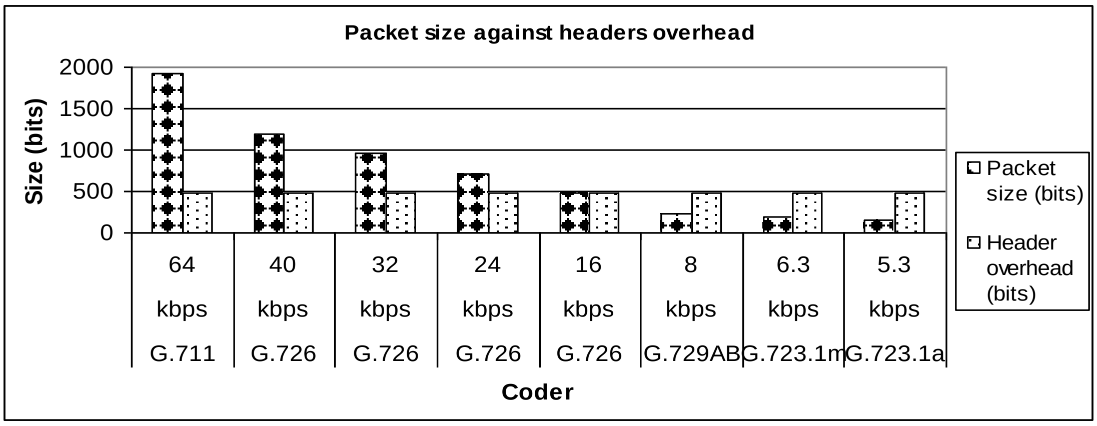


Figure 0-7 Comparison between voice packet size and headers overhead for variety of codecs

[28] Header (RTP/UDP/IP: 40 bytes, Ethernet: 18 bytes, Key management: 3 bytes).
[29] Frame size + Header overhead.
[30] Header overhead / Frame size × 100

The following examples illustrate the delay budget for different network configurations:

Example 6.3: LAN of building 2 at Amman Arab University for Graduate Studies.

Type: Fast Ethernet.

Speed: 100 Mbps.

In the case of LAN, there is no network delay since there is no router in the path of the packet; instead there is a collision that may reduce the actual bandwidth to about 33% of its full capacity; this may happen if the maximum number of terminals and repeaters is installed; in practice this rarely exists and therefore in most of the cases the bandwidth is about 70 – 80% of its full capacity. Table 0-8 illustrates the delay budget of variety codecs on this network for both the best and worst bandwidth utilization. It is clear from the estimated values in the table that any codec can be used on this configuration since none of them exceeds the end-to-end delay threshold; it is also clear that G.711 and G.726 introduced the minimum delay; therefore G.711 which has no compression is recommended to be used in this case.

Example 6.4: LAN-to-LAN, a link between the two campuses of Amman Arab University for Graduate Studies.

Type: Fast Ethernet at each campus (100 Mbps).

Link speed: 512 kbps.

In this case there are two routers on the path (one at each campus). The UDP transport layer of the TCP/IP suite is suitable to be used for messages transfer between the two campuses. The UDP is selected rather than the TCP because there is no need to retransmit any lost packets since an individual packet carries only few voice data (almost 30 ms of voice data) and therefore there is no effect of loosing one or two packets; also the retransmission of the lost packets affects the synchronization between the two participants. To be fair, if burst packets are lost, then the quality of voice will be affected. Table 0-9 illustrates the delay budget of variety codecs on this network. Again the delay of the codecs PCM, ADPCM (all its variations), and CS-ACELP are within the toll quality area of Figure 0-1; while the delay of the codecs MP-MLQ and ACELP are within the good area of the same figure. Therefore all codecs can be used on this configuration since none of them exceeds the end-to-end delay standard recommendations.

Example 6.5: National network.

Number of routers: 3.

Link speed: 2 Mbps.

Distance: up to 1000 km

This example illustrates the end-to-end delay of voice messenger on WAN with three routers as expected for most of the national WAN configurations. The estimation delays of this configuration are shown in Table 0-10. In fact all mentioned codecs achieve toll or good quality over national WANs as it is shown in Table 0-11 and Figure 0-8. None of these delays exceed the end-to-end delay of the standard recommendations; therefore any of them can be used in this situation.

Table 0-8: End-to-end delay of voice packet on fast Ethernet for a variety of codecs.

BW	Vocoder		Packet size (bits)	Fixed Delay (ms)						Variable Delay (ms)			Total Delay (ms)
	Coder	Reference		Coder	Decoder[31]	Security	Serialization	Propagation	Sub total	Queuing	Network	Sub total	
100 Mbps	PCM	G.711	2408	30.125	15.0625	1.7928	0.0241	0.006	47.010	0.0753	0	0.07528	47.0857
	ADPCM	G.726	1688	30.125	15.0625	1.7928	0.0169	0.006	47.003	0.0681	0	0.06808	47.0713
	ADPCM	G.726	1448	30.125	15.0625	1.7928	0.0145	0.006	47.001	0.0657	0	0.06568	47.0665
	ADPCM	G.726	1208	30.125	15.0625	1.7928	0.0121	0.006	46.998	0.0633	0	0.06328	47.0617
	ADPCM	G.726	968	30.125	15.0625	1.7928	0.0097	0.006	46.996	0.0609	0	0.06088	47.0569
	CS-ACELP	G.729AB	728	45.000	22.5000	1.7928	0.0073	0.006	69.306	0.0585	0	0.05848	69.3646

[31] Decoder delay is assumed to be 1/2 of the coder delay [Kostas et al, 1998].

	MP-MLQ	G.723.1m	677	67.500	33.7500	1.7928	0.0068	0.006	103.056	0.0580	0	0.05797	103.1135
	ACELP	G.723.1a	647	67.500	33.7500	1.7928	0.0065	0.006	103.055	0.0577	0	0.05767	103.1129
0.33 * 100 Mbps	PCM	G.711	2408	30.125	15.0625	1.7928	0.0730	0.006	47.059	0.2281	0	0.22812	47.2874
	ADPCM	G.726	1688	30.125	15.0625	1.7928	0.0512	0.006	47.037	0.2063	0	0.20630	47.2438
	ADPCM	G.726	1448	30.125	15.0625	1.7928	0.0439	0.006	47.030	0.1990	0	0.19903	47.2292
	ADPCM	G.726	1208	30.125	15.0625	1.7928	0.0366	0.006	47.023	0.1918	0	0.19176	47.2147
	ADPCM	G.726	968	30.125	15.0625	1.7928	0.0293	0.006	47.016	0.1845	0	0.18448	47.2001
	CS-ACELP	G.729AB	728	45.000	22.5000	1.7928	0.0221	0.006	69.321	0.1772	0	0.17721	69.4981
	MP-MLQ	G.723.1m	677	67.500	33.7500	1.7928	0.0205	0.006	103.069	0.1757	0	0.17567	103.2450

	ACELP	G.723.1a	647	67.500	33.7500	1.7928	0.0196	0.006	103.068	0.1748	0	0.17476	103.2432

Table 0-9 LAN-to-LAN end-to-end delay of voice packet for a variety of codecs

Voice end-to-end delay (ms) on LAN-to-LAN with 512 kbps link and <5 km distance												
Vocoder		Packet size (bits)	Fixed Delay (ms)						Variable Delay (ms)			Total Delay (ms)
Coder	Reference		Coder	Decoder	Security	Serialization	Propagation	Sub total	Queuing	Network	Sub total	
PCM	G.711	2408	30.125	15.0625	1.7928	4.703	0.03	51.713	14.7031	19.406	34.1094	85.8228
ADPCM	G.726	1688	30.125	15.0625	1.7928	3.297	0.03	50.307	13.2969	16.594	29.8906	80.1978
ADPCM	G.726	1448	30.125	15.0625	1.7928	2.828	0.03	49.838	12.8281	15.656	28.4844	78.3228
ADPCM	G.726	1208	30.125	15.0625	1.7928	2.359	0.03	49.370	12.3594	14.719	27.0781	76.4478
ADPCM	G.726	968	30.125	15.0625	1.7928	1.891	0.03	48.901	11.8906	13.781	25.6719	74.5728
CS-ACELP	G.729AB	728	45.000	22.5000	1.7928	1.422	0.03	70.745	11.4219	12.844	24.2656	95.0103
MP-MLQ	G.723.1m	677	67.500	33.7500	1.7928	1.322	0.03	104.395	11.3223	12.645	23.9668	128.3619
ACELP	G.723.1a	647	67.500	33.7500	1.7928	1.264	0.03	104.336	11.2637	12.527	23.7910	128.1275

Table 0-10 National WAN end-to-end delay of voice packet for variety codecs

Voice end-to-end delay (ms) on WAN with 3 routers, 2 Mbps link, and <1000 km distance												
Vocoder		Packet size (bits)	Fixed Delay (ms)						Variable Delay (ms)			Total Delay (ms)
Coder	Reference		Coder	Decoder	Security	Serialization	Propagation	Sub total	Queuing	Network	Sub total	
PCM	G.711	2408	30.125	15.0625	1.7928	1.204	6.0	54.184	3.764	9.936	13.700	67.884
ADPCM	G.726	1688	30.125	15.0625	1.7928	0.844	6.0	53.824	3.404	8.496	11.900	65.724
ADPCM	G.726	1448	30.125	15.0625	1.7928	0.724	6.0	53.704	3.284	8.016	11.300	65.004
ADPCM	G.726	1208	30.125	15.0625	1.7928	0.604	6.0	53.584	3.164	7.536	10.700	64.284
ADPCM	G.726	968	30.125	15.0625	1.7928	0.484	6.0	53.464	3.044	7.056	10.100	63.564
CS-ACELP	G.729AB	728	45.000	22.5000	1.7928	0.364	6.0	75.657	2.924	6.576	9.500	85.157
MP-MLQ	G.723.1m	677	67.500	33.7500	1.7928	0.339	6.0	109.381	2.899	6.474	9.373	118.754
ACELP	G.723.1a	647	67.500	33.7500	1.7928	0.324	6.0	109.366	2.884	6.414	9.298	118.664

Table 0-11 End-to-end delay (ms) over national WANs against number of routers for variety codecs

Vocoder		# of routers			
Coder	**Reference**	**3**	**5**	**7**	**9**
PCM	G.711	67.88	77.82	87.76	97.69
ADPCM	G.726	65.72	74.22	82.72	91.21
ADPCM	G.726	65.00	73.02	81.04	89.05
ADPCM	G.726	64.28	71.82	79.36	86.89
ADPCM	G.726	63.56	70.62	77.68	84.73
CS-ACELP	G.729AB	85.16	91.73	98.31	104.88
MP-MLQ	G.723.1m	118.75	125.23	131.70	138.18
ACELP	G.723.1a	118.66	125.08	131.49	137.91

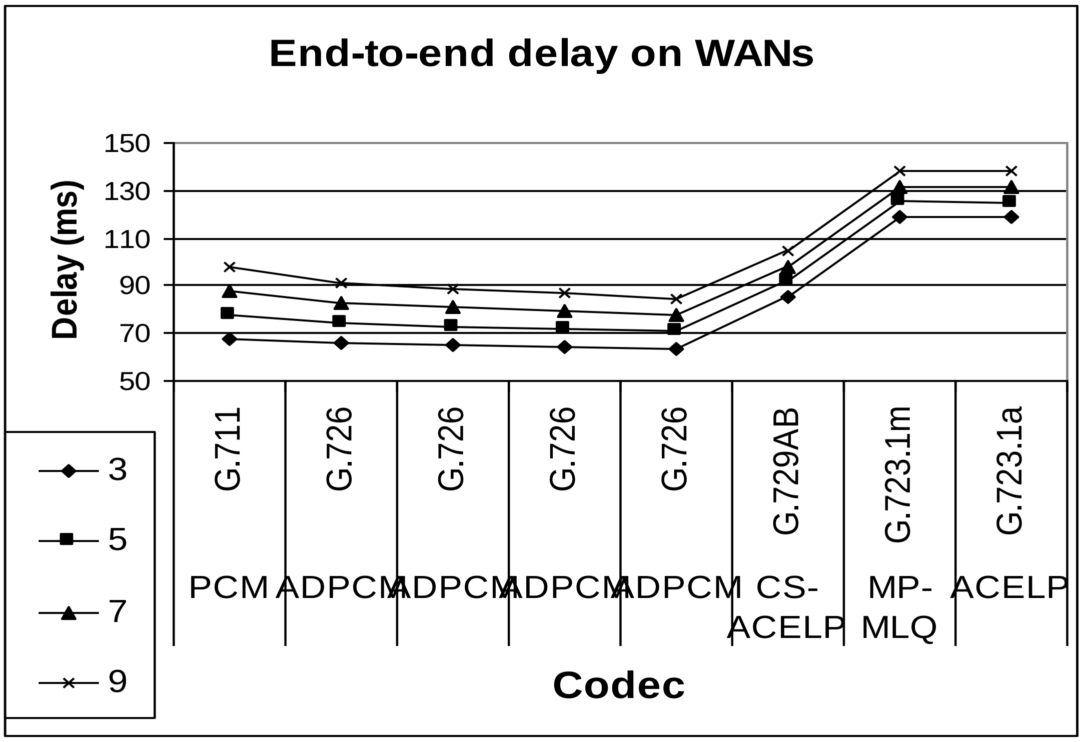


Figure 0-8 End-to-end delay over national WANs against number of routers for variety codecs

Example 6.6: Internet.

Number of routers: 15.

Link speed: 2 Mbps.

Distance: up to half of the earth circumference (13000 km)

This example illustrates the end-to-end delay of voice messenger on the internet with fifteen routers as the maximum allowed number of routers to be in this configuration. The estimation delays of this configuration are shown in Table 0-12. It is clear that none of these codec delays is within the toll or good quality; but it is potentially useful. Therefore they can be used but with less quality especially the PCM, ADPCM, and CS-ACELP where their delays are less than 200 ms. Keep in mind that this is the worst case where it is assumed that the distance between the two participants are half of the earth circumference (i.e. 78 ms propagation time); also it is assumed that the maximum possible number of routers of any path (i.e. 15 routers) are setting on the path of the voice packet and this is rare to happen in the real life.

Table 0-13 and Figure 0-9 summarize the delay budget of voice messenger for a variety codecs on the globe with different number of routers on the message path. The shaded delays in the table are less than the standard limit (150 ms); all delays are bigger than this limit when the number of routers becomes seven or more. The delays in the table are subject to be changed –increased or decreased- depending on the queuing delays across the path since the given delays are computed for specific assumptions. It is clear that the delay increases as the number of routers increases. The delays of MP-MLQ and ACELP are always bigger than the delays of the other codecs; this is due to the high processing delays of these codecs. Table 0-14 illustrates the same data in more details where the fixed, queuing, network, and total delays are shown.

The problem can be solved or the delay can be minimized by keeping the number of routers as minimum as possible, increasing the link speed, and employing a queuing delay that minimizes the queuing time of the voice packets.

Table 0-12 Internet end-to-end delay of voice packet for variety codecs

Voice end-to-end delay (ms) on WAN with 15 routers and 2 Mbps link through the global												
Vocoder		Packet size (bits)	Fixed Delay (ms)						Variable Delay (ms)			Total Delay (ms)
Coder	Reference		Coder	Decoder	Security	Serialization	Propagation	Sub total	Queuing	Network	Sub total	
PCM	G.711	2408	30.125	15.0625	1.7928	1.204	78.0	126.184	3.764	69.552	73.316	199.500
ADPCM	G.726	1688	30.125	15.0625	1.7928	0.844	78.0	125.824	3.404	59.472	62.876	188.700
ADPCM	G.726	1448	30.125	15.0625	1.7928	0.724	78.0	125.704	3.284	56.112	59.396	185.100
ADPCM	G.726	1208	30.125	15.0625	1.7928	0.604	78.0	125.584	3.164	52.752	55.916	181.500
ADPCM	G.726	968	30.125	15.0625	1.7928	0.484	78.0	125.464	3.044	49.392	52.436	177.900
CS-ACELP	G.729AB	728	45.000	22.5000	1.7928	0.364	78.0	147.657	2.924	46.032	48.956	196.613
MP-MLQ	G.723.1m	677	67.500	33.7500	1.7928	0.339	78.0	181.381	2.899	45.318	48.217	229.598
ACELP	G.723.1a	647	67.500	33.7500	1.7928	0.324	78.0	181.366	2.884	44.898	47.782	229.148

Table 0-13 End-to-end delay over the globe against number of routers for variety codecs

End-to-end delay (ms)						
Vocoder		**# of routers**				
Coder	**Reference**	**3**	**5**	**6**	**11**	**15**
PCM	G.711	139.88	149.82	154.79	179.63	199.50
ADPCM	G.726	137.72	146.22	150.47	171.71	188.70
ADPCM	G.726	137.00	145.02	149.03	169.07	185.10
ADPCM	G.726	136.28	143.82	147.59	166.43	181.50
ADPCM	G.726	135.56	142.62	146.15	163.79	177.90
CS-ACELP	G.729AB	157.16	163.73	167.02	183.46	196.61
MP-MLQ	G.723.1m	190.75	197.23	200.46	216.65	229.60
ACELP	G.723.1a	190.66	197.08	200.28	216.32	229.15

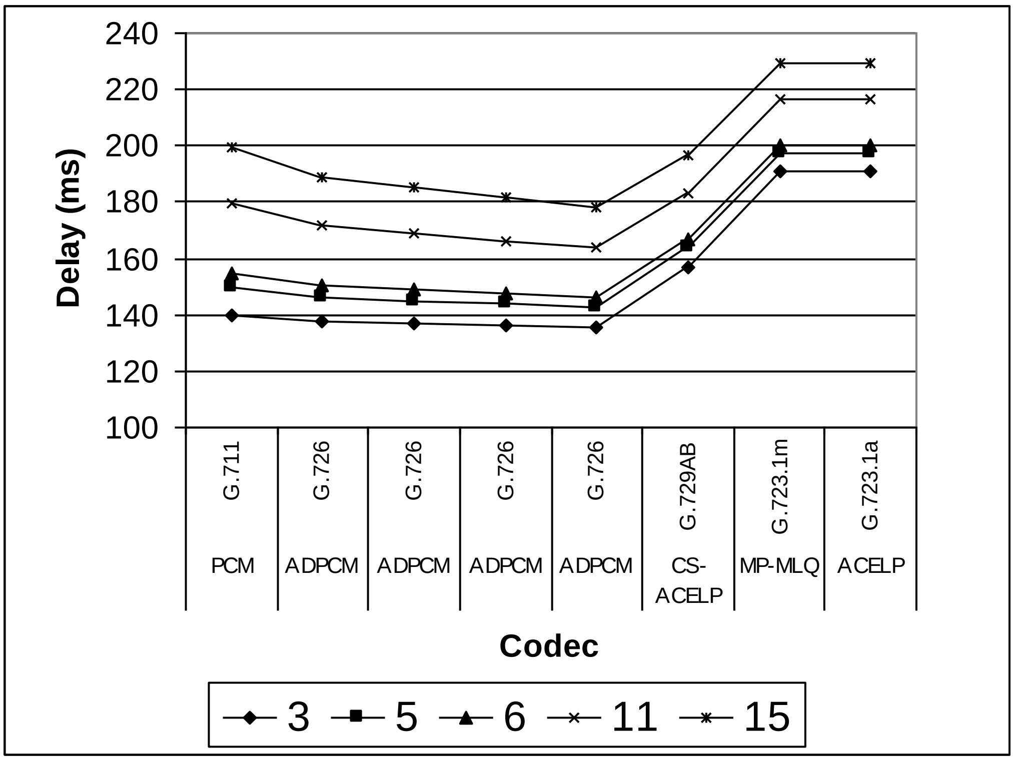


Figure 0-9 End-to-end delay against number of routers for variety codecs

Table 0-14 End-to-end delay of voice packet for variety codecs against # of routers

Voice end-to-end delay (ms) on WAN with variety number of routers and 2 Mbps link through the global for variety codecs						
				Variable delays (ms)		
Coder	**Reference**	**# of routers**	**Fixed delays (ms)**	**Queuing delay**	**Network delay**	**Total Delay (ms)**
PCM	G.711	3	126.18	3.76	9.94	139.88
		5			19.87	149.82
		6			24.84	154.79
		11			49.68	179.63
		15			69.55	199.50
ADPCM	G.726	3	125.82	3.40	8.50	137.72
		5			16.99	146.22
		6			21.24	150.47
		11			42.48	171.71
		15			59.47	188.70
ADPCM	G.726	3	125.70	3.28	8.02	137.00
		5			16.03	145.02
		6			20.04	149.03
		11			40.08	169.07
		15			56.11	185.10
ADPCM	G.726	3	125.58	3.16	7.54	136.28
		5			15.07	143.82
		6			18.84	147.59
		11			37.68	166.43
		15			52.75	181.50
ADPCM	G.726	3	125.46	3.04	7.06	135.56
		5			14.11	142.62
		6			17.64	146.15
		11			35.28	163.79
		15			49.39	177.90
CS-ACELP	G.729AB	3	147.66	2.92	6.58	157.16
		5			13.15	163.73
		6			16.44	167.02
		11			32.88	183.46
		15			46.03	196.61
MP-MLQ	G.723.1m	3	181.38	2.90	6.47	190.75
		5			12.95	197.23
		6			16.19	200.46
		11			32.37	216.65
		15			45.32	229.60
ACELP	G.723.1a	3	181.37	2.88	6.41	190.66
		5			12.83	197.08
		6			16.04	200.28
		11			32.07	216.32
		15			44.90	229.15

6.5. Conclusions and recommendations

Conclusions:

1. There exists a direct relationship between the packet size and the average time required to carry out the overall security process at both sides (sender and receiver). If the packet size is doubled, then the time is almost doubled too.
2. The average time required at the sender side is a little bit larger than the average time required at the receiver side. The relationship between this difference in time and the packet size is a direct relationship. This difference in time is negligible because it is small (about .06 ms with 256 bytes packet). This difference exists because the algorithms are tested separately, and the randomness selection of keys is an important factor that affects the results. Even, if there exist a small amount of time as a difference between the time used at the sender side and the time used at the receiver side, it would not affect the voice data since the receiver processes each packet as it is arrive; If there still exists a difference in time between the sender and the receiver, the de-jitter buffers at the receiver side solve this problem.
3. The average time used at both sides in this approach is very small compared to the average time used by known algorithms (see Tables 6-2 and 6-3).
4. The proposed algorithm produces a huge number of possible keys; this number grows quickly as values of the different factors increased; e.g. there are 3×10^6 different possible keys with the following parameters: (MKL: 1024, NOKL: 32, NOKSC: 5, and NOKSLC: 4); In the second scenario this number becomes 3×10^8 if there are 100 users and 3×10^9 with 1000 users.
5. The time required for any unauthorized user to identify a voice message using brute-force attack is very big as this illustrated in Tables 6-4 and 6-5 even if it is assumed

6. that half of the permutations is enough on average to reach the goal.
7. Any of the mentioned codecs in this chapter can be used in PC-to-PC telephony application over fast Ethernet LAN with toll quality service.
8. Any of the mentioned codecs with the given assumptions in this chapter can be used in PC-to-PC telephony application over national WANs (i.e. up to 5 km distance and up to 9 routers between the two parties) with toll or good quality service.
9. PCM and ADPCM keep the end-to-end delay of voice messages below the standard recommendations over the globe if the number of routers in the path is less than seven routers. At the same time all of the mentioned codecs achieve a potentially useful quality service (less than 250 ms delay) over the globe even with the maximum number of routers on the path (i.e. 15 routers). This is true with the given assumptions for the link speed and the queuing delays.

Recommendations:

1. Because of the facts presented in points 1 and 2 above, it is recommended to keep the packet size small enough in order to minimize the required time for the overall process, keeping in mind that reducing the packet size adds overhead on the transmission process which is due to the packet headers and that there is a trade off between processing time and headers overhead. A final recommendation on the best/suitable packet size is introduced in chapter seven where all latencies/delays including data transmission and network queuing were put together.
2. It is recommended to enlarge the master key length as it does not affect the amount of time needed to carry out the security processes.

3. It is also recommended to use a large range of key length since the proposed algorithm runs using keys of any length.
4. It is also recommended to increase the values of the rest factors since they do not add any significant amount of time to the time required by the security processes.
5. The proposed algorithm can be used to secure VoIP since it does not add much time to the delays that VoIP suffers.
6. It is recommended to use the PCM coder over the LANs, LAN-to-LAN, and national WANs since it hasn't any kind of compression and gives the best voice quality. Other codecs can be used too.
7. It is recommended to use the variations of the ADPCM codecs over the international calls since they achieve the minimal delays and at the same time produce a good voice quality.
8. It is recommended to avoid using the MP-MLQ and ACLEP codecs in this kind of applications because of their high processing delays and high overhead penalty; and hence high delays.

System Prototype "Design and Implementation"

7.1. Introduction

This chapter introduces and describes the practical aspect of this research. Figure 0-1 shows a block diagram of the proposed technique. Even though most of the algorithms have been described in the preceding chapters; the system architecture including database design, user interface, and some other algorithms are presented here.

The system consists of a server and a set of clients. The server is responsible for organizing and managing the users' information, and distributing the updated information to the clients whenever it is updated; it keeps track of the active users and update the database whenever a user logs in or logs out. Each client maintains a database that includes information about the users which is necessary to be used during the conversations. This architecture is shown in Figure 0-2.

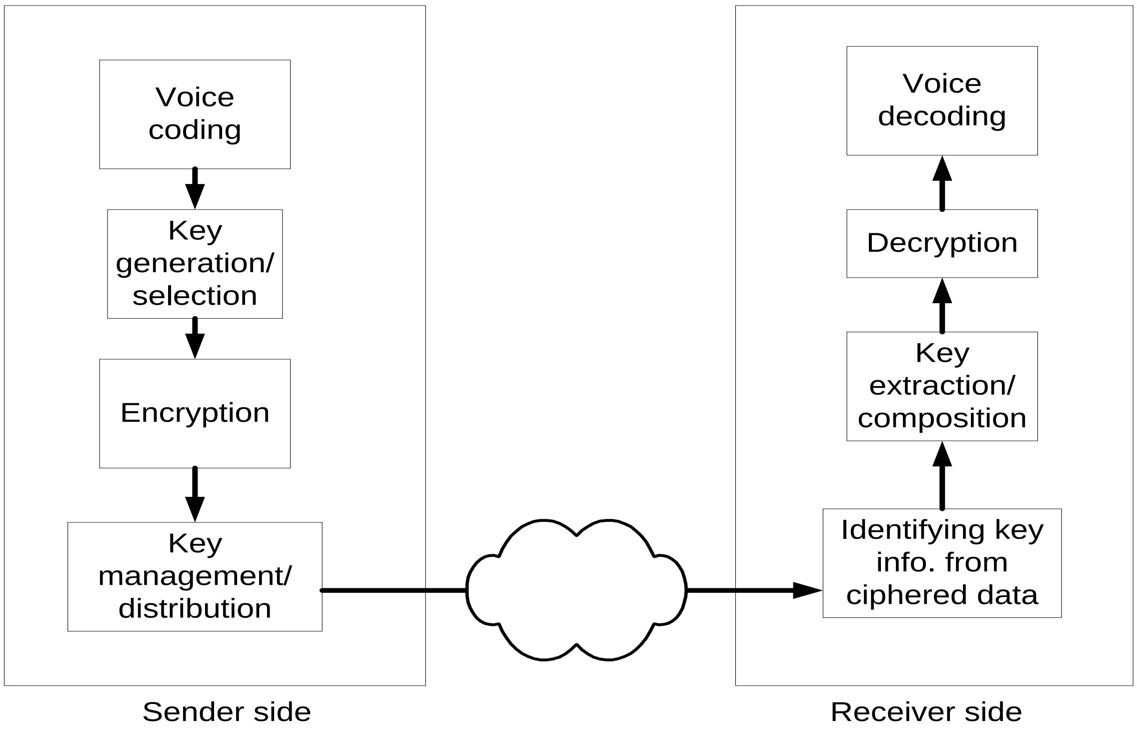

Voice coding
Key generation/ selection
Encryption
Key management/ distribution
Voice decoding
Decryption
Key extraction/ composition
Identifying key info. from ciphered data
Sender side
Receiver side

Figure 0-1: A block diagram of the proposed technique

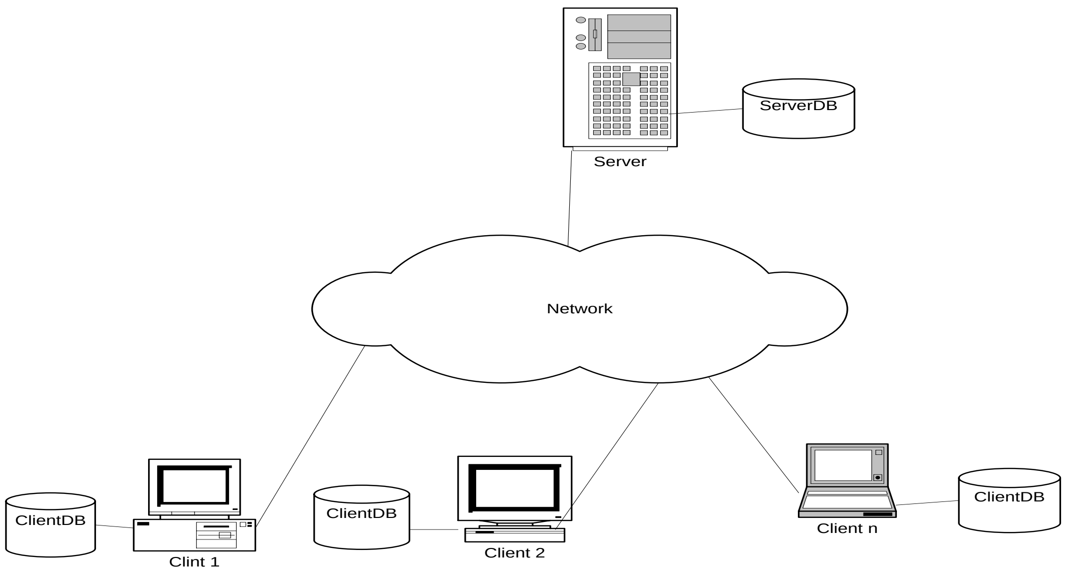


Figure 0-2 System architecture

7.2. Databases

The system requires building of two databases: a server database, which is to be located on the server, stores all information needed to manage and control the system and a client database, which is to be located on the client machine, stores the users' master keys and temporary information about the user hot list. Details of both databases are available in the coming sections.

7.2.1. Server DB

This database is to be located on the server and managed by the system administrator. It is accessible by all users.

7.2.1.1. ERM

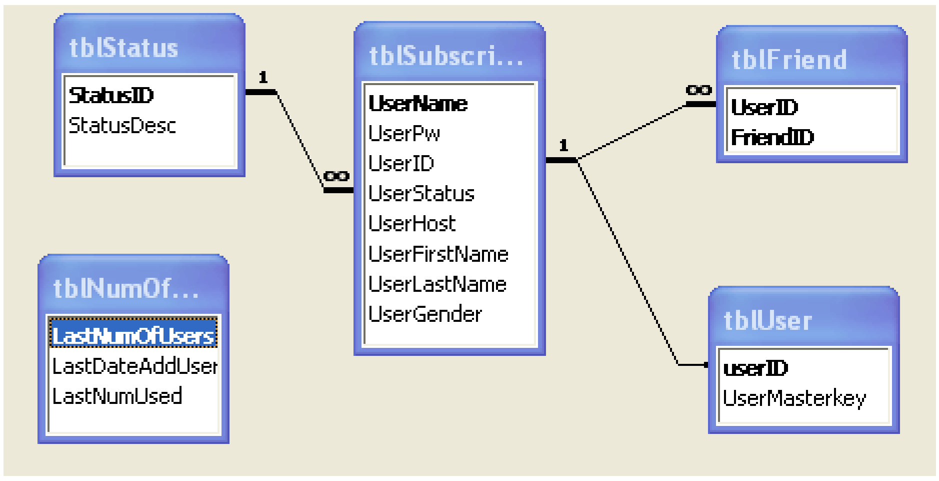


Figure 0-3 : ERM of the ServerDB

7.2.1.2. Database Tables

Table 0-1 Users database table description in serverDB

tblUser: Contains the information needed for the encryption/decryption processes. Its records are generated through the algorithm "Create users" in the basis of one record per user.

Field Name	Data type	Size	Description
UserID	Numeric	6	A sequential number used as identifier for a user.
UserMasterKey[32]	String	256	Randomly selected Hexadecimal digits that form the master key of this user.

[32] The size of this field can be increased to achieve more possible keys as mentioned in chapter 7.

Table 0-2: Status database table description in serverDB

tblStatus: contains a description for different status codes of the user.			
Field Name	Data type	Size	Description
StatusID	Numeric	1	Status code.
StatusDesc	String	30	String describes the status (e.g. Active, Inactive, …).

Table 0-3 : Subscribers database table description in serverDB

tblSubscriber: Contains subscribers' information; It has one record for each user; This information is needed to validate login to the system, identify users' locations, and their statuses[33].			
Field Name	Data type	Size	Description
UserName	String	50	Subscriber name.
UserPw	Numeric	12	A hashing value represents a user password.
UserID	Numeric	6	Foreign key links this table with the table "tblUser".
UserStatus	Numeric	1	A code describes the status of the user; A foreign key links this table with the table "tblStatus".
UserHost	String	50	A string describes the computer name or IP address of the user.
UserFirstName	String	15	The user's First Name.
UserLastName	String	15	The user's Last Name.
UserGender	String	1	The user's Gender (M or F).

[33] Other information can be added as required.

Table 0-4 : Friends database table description in serverDB

tblFriend: contains records that identify friends of the users.			
Field Name	Data type	Size	Description
UserID	Numeric	6	See tblUser
FriendID	Numeric	6	A UserID who is a friend to this user.

Table 0-5 : Number of users database table description in serverDB

tblNumOfUsers: contains the number of users who can be registered on the system as well as the registered number of users.			
Field Name	Data type	Size	Description
LastNumOfUsers	Numeric	6	The number of users who can be registered on the system.
LastDateAddUsers	Date	8	The last date users were created.
LastNumUsed	Numeric	6	The number of registered users on the system.

7.2.2.ClientDB

This database is based on each client machine. It contains two tables; one is permanent and identical for all clients and the other is temporary and differs from one client to another.

7.2.2.1. ERM

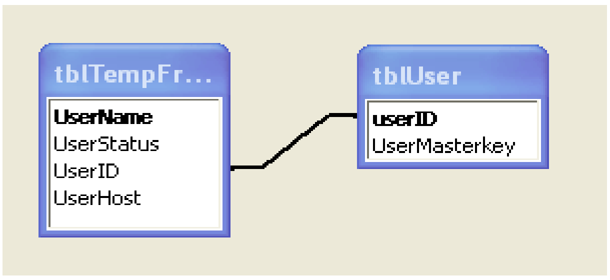


Figure 0-4 ERM of the ClientDB

7.2.2.2. Database Tables

a. tblUser[34]: a copy of the tblUser from the ServerDB on each client machine; this table will be used during the encryption/decryption processes.

b. tblTempFriend: a temporary table is to be generated whenever a user is logged in. It contains information - needed to carry out a call - about friends (active and inactive) whom a person would like to be in contact with and share some information with. The records of this table differ from client to another depending on the user friends list.

Table 0-6 : Temporary friends database table description in clientDB

tblTempFriend: contains friends' information of the current user. It is needed to communicate with friends.

Field Name	Data type	Size	Description
UserName	String	50	Subscriber name.
UserStatus	Numeric	1	A code describes the status of the user.
UserID	Numeric	6	Foreign key links this table with the table "tblUser".
UserHost	String	50	A string describes the computer name or IP address of the user.

7.3. Algorithms and Interface

7.3.1. Create Users

Procedure: Create users

Executed by: System administrator

Location: Server

Time: Through the installation process[35].

Input:

[34] For the table description see Table 0-1

[35] It can be executed later on at any time, but in this case the modified table "tblUser" should be sent to all clients.

1. Number of users to be created; say **N**.
2. Network speed.

Output:

1) N records on tblUser of the ServerDB will be inserted; and then they will be copied to the ClientDB on all clients.

2) The table "tblNumOfUsers" will be updated.

Comments: It can be executed later on at any time, but in this case the updated table "tblUser" should be sent to all clients. It is advisable that the administrator creates enough number of users during the system installation.

Algorithm steps:

```
1. Input number of users to be created (say N).
2. Select the network speed[36].
3. Read number of created users from table "tblNumOfUsers", say (NCU).
4. Read number of registered users (i.e. subscribers) from table "tblNumOfUsers", say
   (NRU).
5. If N is invalid depending on the network speed then
      5.1. Display an invalid message.
      5.2. Go to 1.
   ELSE
      Continue.
6. For I = (NCU + 1) to (NCU+N)
   6.1. Set UserID ← I.
```

[36] It can be programmed to be captured by the system.

```
7.  6.2. Generate a randomly 256 hexadecimal digits as "UserMasterKey".
     6.3. Insert a record/row in the table "tblUser".
8.  Set LDAU[37] (LastDateAddUsers) ← System date.
9.  If NCU > 0 then
    8.1. Broadcast the modified table "tblUser" to all installed clients' machines.
    8.2. Update NCU and LDAU in the table "tblNumOfUsers".
10. END
```

Interface form:

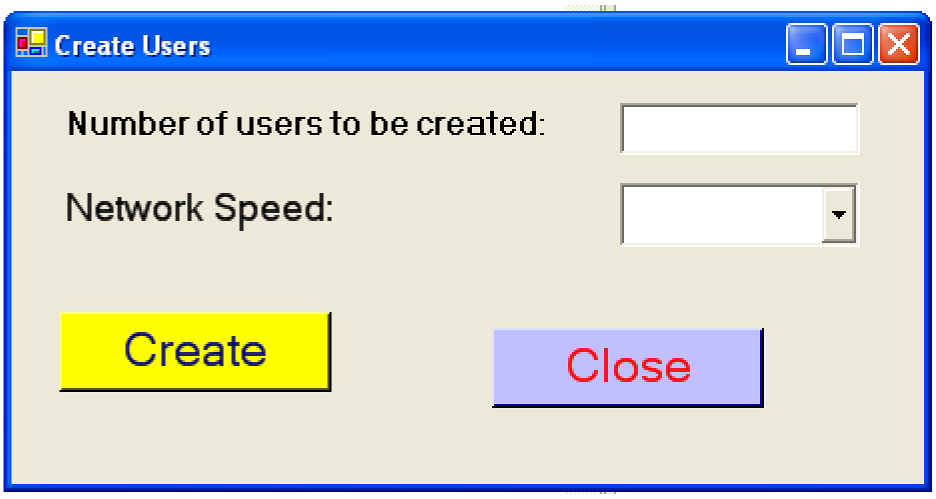


Figure 0-5 : Interface form of create users algorithm

Example 7.1

The table "tblNumOfUsers" initially has the following record:

LastNumOfUsers	LastDateAddUsers	LastNumUsed
0		0

The table "tblUser" initially has no records.

The procedure is executed on 28/5/2005 with the following input:

Number of users to be created: 20

Network speed: 10 Mbps

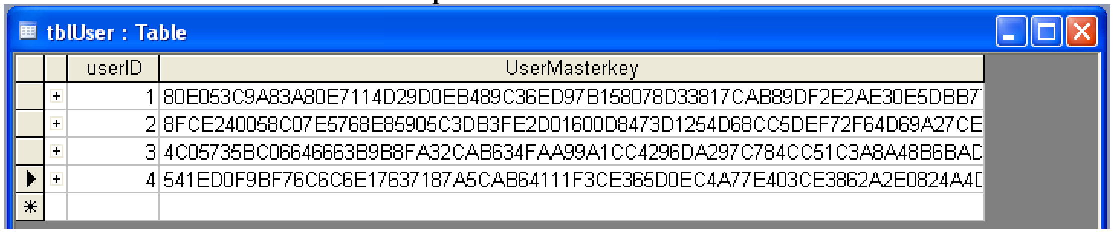
[37] This will be useful in the case of adding users rather than the first time.

Since the number of users is valid; the table "tblNumOfUsers" is updated and becomes:

LastNumOfUsers	LastDateAddUsers	LastNumUsed
20	5/28/2005	0

And 20 records are appended to the table "tblUser". The first four rows of this table are shown in Table 0-7. Only part of the "UserMasterKey" is shown for each record.

Table 0-7 : Sample rows of the database table "tblUser"


tblUser : Table

userID	UserMasterkey
1	80E053C9A83A80E7114D29D0EB489C36ED97B158078D33817CAB89DF2E2AE30E5DBB7
2	8FCE240058C07E5768E85905C3DB3FE2D01600D8473D1254D68CC5DEF72F64D69A27CE
3	4C05735BC06646663B9B8FA32CAB634FAA99A1CC4296DA297C784CC51C3A8A48B6BAD
4	541ED0F9BF76C6C6E17637187A5CAB64111F3CE365D0EC4A77E403CE3862A2E0824A4D

7.3.2. User Registration

Procedure: User registration

Executed by: Subscribers

Location: Client

Time: Any time.

Input: Subscriber information (user name, password, confirm password, first name, last name, and gender).

Output:

1. A record will be appended on the table "tblSubscriber".
2. Number of registered users in table "tblNumOfUsers" will be updated (i.e. increased by 1).

3.

Algorithm steps:

```
1.  Input user name, password, and retype password.
2.  Check password with the retyped password.
3.  If passwords are not matching then
        3.1. Display an error message.
        3.2. Go to 1.
    ELSE
        Continue.
4.  Input first name, last name, and gender.
5.  If any field is (null OR space) then
        5.1. Display an invalid message.
        5.2. Go to 4.
    ELSE
        Continue.
6.  Get the computer name.
7.  Hash the password.
8.  Set UserStatus ← 2 (i.e. inactive).
9.  Connect to the ServerDB.
10. Send the inputted data to the ServerDB.
11. Validate the user name for duplicates.
12. If user name is valid then
```

```
13.
14.    12.1. Append a record to "tblSubscriber".
       12.2. Add 1 to "LastNumUsed" in the table "tblLastNumOfUsers"
       12.3. Display a congratulation message.
   ELSE
        12.4. Display a rejection message.
        12.5. Go to 1.
15. END
```

Interface form:

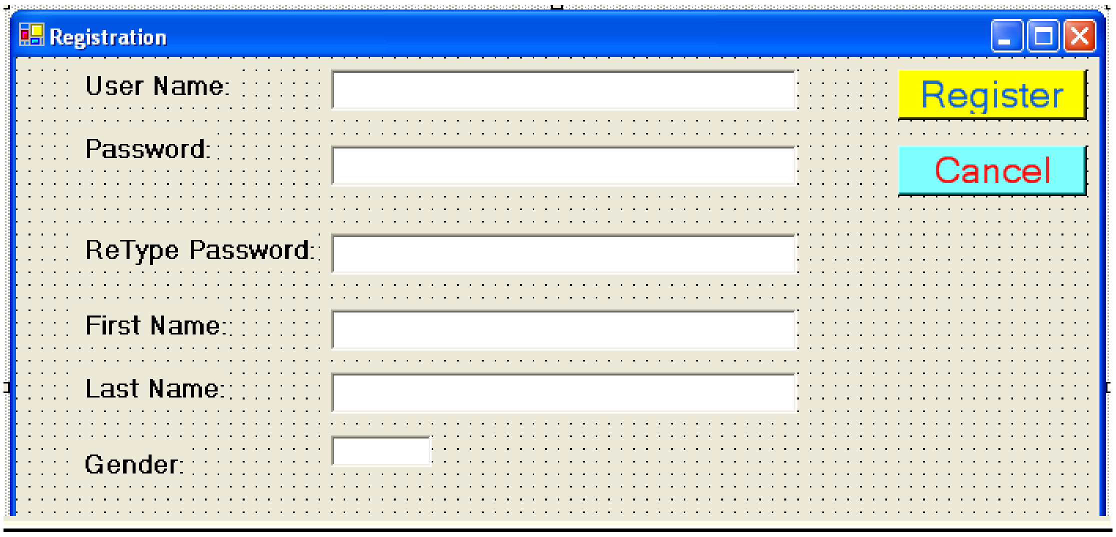


Figure 0-6 : Interface form of user registration algorithm

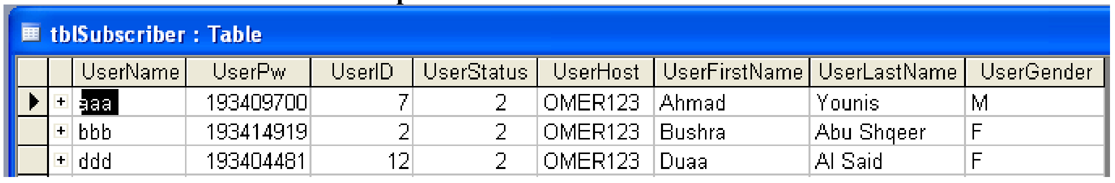
Example 7.2

After the registration of three users; the table "tblNumOfUsers" will be updated and becomes:

LastNumOfUsers	LastDateAddUsers	LastNumUsed
20	5/28/2005	3

And three records are appended to the table "tblSubscriber". These records are shown in Table 0-8.

Table 0-8 : Sample rows of the database table "tblSubscriber"



tblSubscriber : Table

UserName	UserPw	UserID	UserStatus	UserHost	UserFirstName	UserLastName	UserGender
aaa	193409700	7	2	OMER123	Ahmad	Younis	M
bbb	193414919	2	2	OMER123	Bushra	Abu Shqeer	F
ddd	193404481	12	2	OMER123	Duaa	Al Said	F

7.3.3. User Login

Procedure: User Login

Executed by: Subscriber

Location: Client

Time: Any time.

Input: User name and password.

Output:

1. In the table "tblSubscriber" of the serverDB, the value of the field "UserStatus" becomes "active" and the value of the field "UserHost" becomes the name of the host where user logged in.
2. Controls properties of the login form will be modified.
3. Friends' lists (active and inactive) of this user will be filled.
4. The table "tblTempFriend" of this user will be created and filled.
5. A socket is created and opened to listen for messages from others.
6. A login message should be sent to all active friends of this user.
7. All active friends who receive the login message should update their friends' lists as well as their "tblTempFriend" tables.

Algorithm steps:

```
1.  Enter user name and password.
2.  If (user name OR password) is (null or space) then
        2.1. Display an error message.
        2.2. Go to 1.
    ELSE
        Continue.
3.  Get the computer name.
4.  Hash the password.
5.  Connect to the server.
6.  Send user name, hashed password, and computer name to the server.
7.  Read the user record from the table "tblSubscriber".
8.  Validate user name and password.
9.  If (user name OR password) is invalid then
        2.1. Display an invalid message.
        2.2. Go to 1.
    ELSE
        Continue.
10. Modify controls' properties on the interface form.
11. Show the user full name.
12. Update the user status and host name on the table " tblSubscriber".
13. Start listening to messages from others.
14. Read table "tblFriend" to get friends of this user.
```

15. Create a DB table named "tblTempFriend"&UserID on the ClientDB of this user.

16. Prepare friends' information for this user; this includes:

  16.1. Fill the active friends' list.

  16.2. Fill the inactive friends' list.

  16.3. Append a record/row for each of the user friends in the table "tblTempFriend"&UserID (Including both active and inactive friends).

  16.4. Inform by messages all active friends of this user with his/her new status and host.

17. Any person who is a friend of the user receives the login message should update his/her friends' lists and his/her "tblTempFriend" table (i.e. shifts the logged in user from the inactive to the active list).

18. END

Interface form:

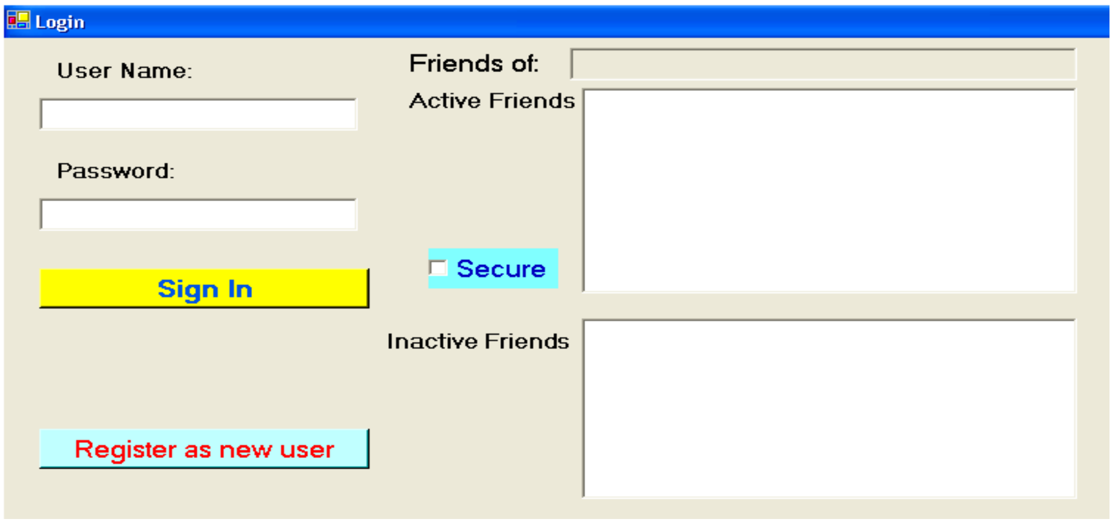
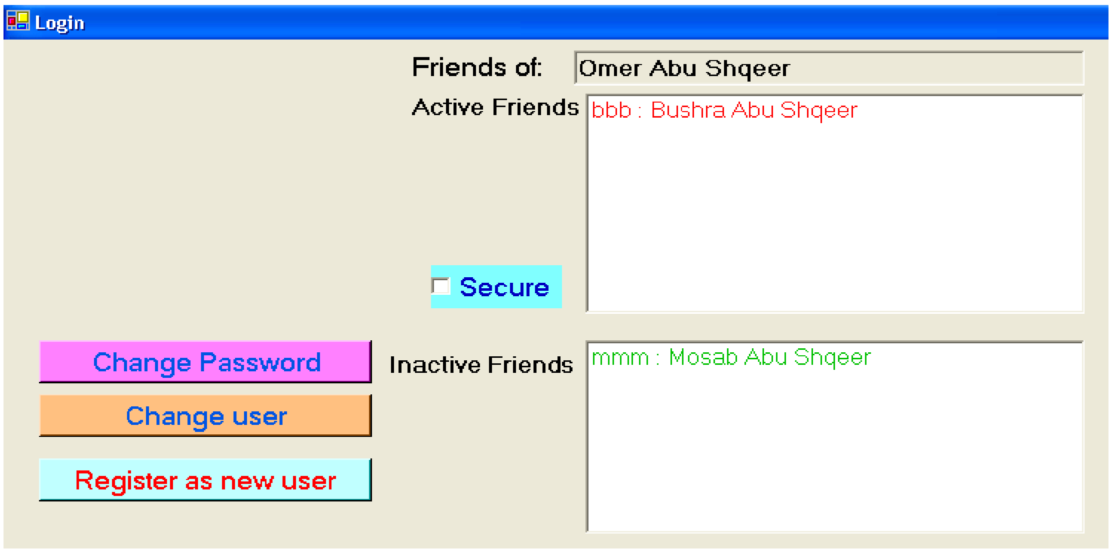


Figure 0-7 : Interface form of user login algorithm before a user login



Friends of: Omer Abu Shqeer

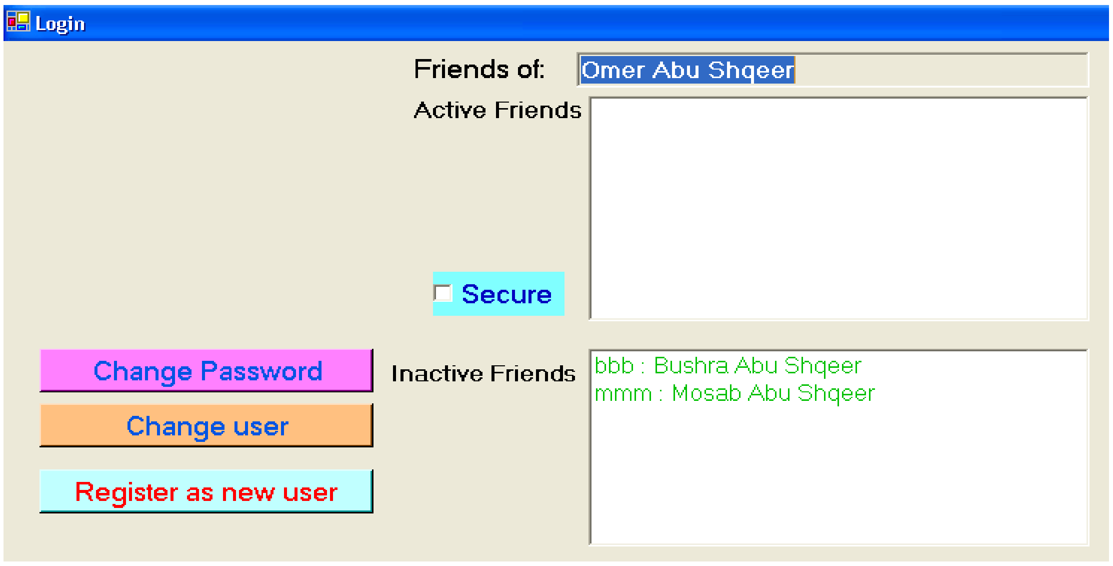
Active Friends: bbb : Bushra Abu Shqeer

Inactive Friends: mmm : Mosab Abu Shqeer

Figure 0-8 : Interface form of user login algorithm after a user logged in

Example 7.3

Assume that the user Bushra Abu Shqeer has the friends Omar Abu Shqeer, Mosab Abu Shqeer, and Naim Ajlooni. Assume also that Omar Abu Shqeer and Naim Ajlooni are active (i.e. currently logged in as the user status in the DB) while Mosab Abu Shqeer is inactive. Figures 7-9 and 7-10 show the interface forms of both Omar Abu Shqeer and Naim Ajlooni. Observe that Bushra Abu Shqeer appears on their inactive lists.

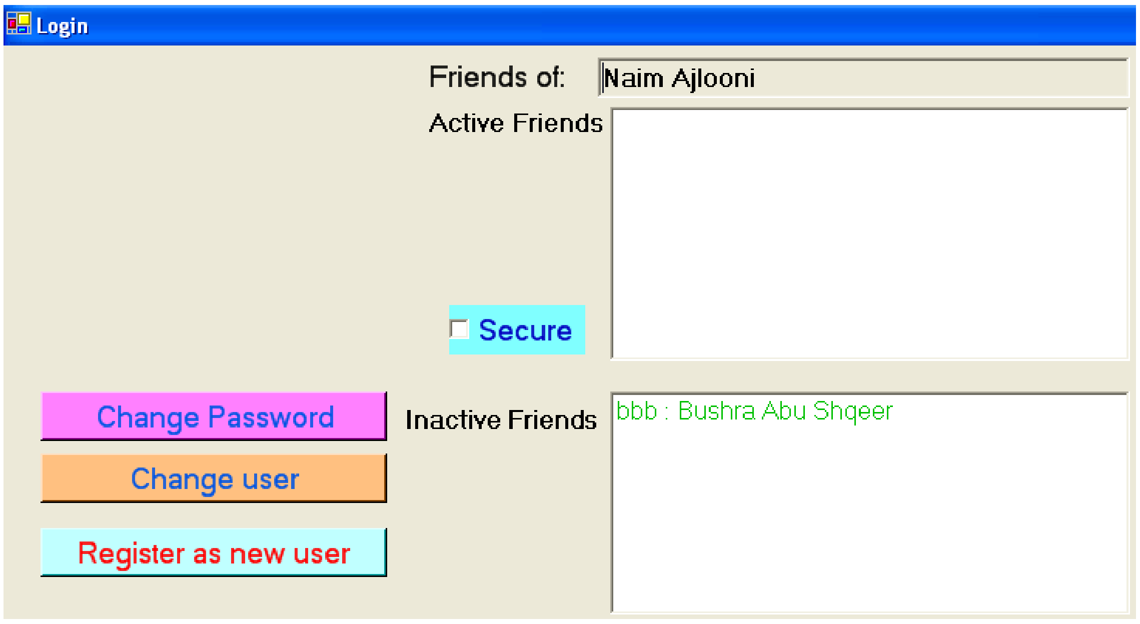


Friends of: Omer Abu Shqeer

Active Friends

Inactive Friends
bbb : Bushra Abu Shqeer
mmm : Mosab Abu Shqeer

Figure 0-9 : Interface form of the user "Omer" logged in and none of his friends is active


Login

Friends of: Naim Ajlooni

Active Friends

☐ Secure

Change Password

Change user

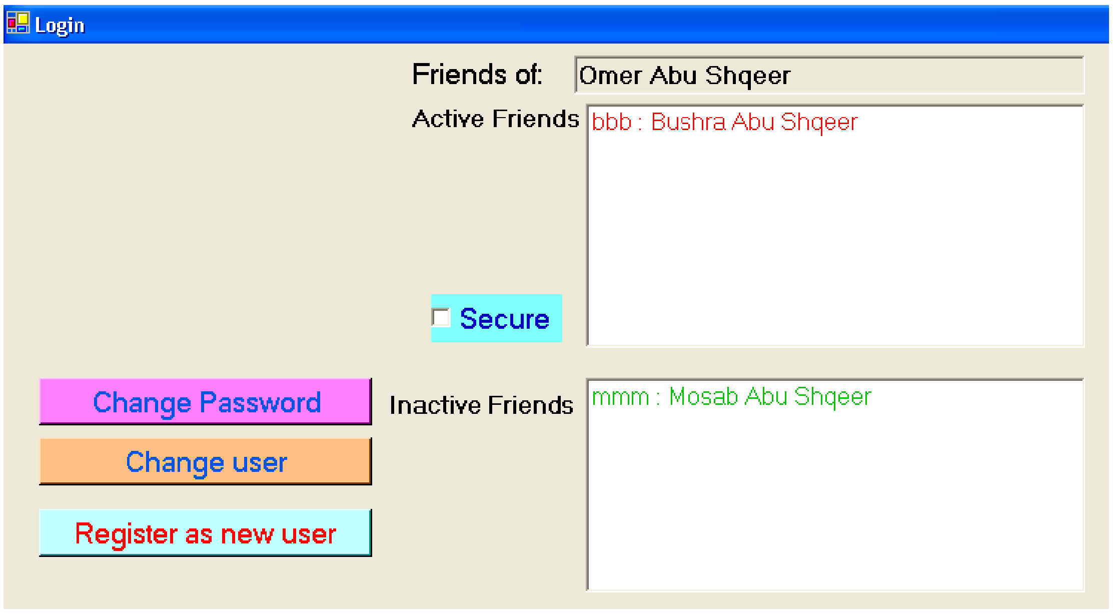
Register as new user

Inactive Friends

bbb : Bushra Abu Shqeer

Figure 0-10 : Interface form of the user "Naim" logged in and none of his friends is active

If Bushra Abu Shqeer logs-in then the interface forms of Omar Abu Shqeer, Naim Ajlooni, and Bushra Abu Shqeer look like Figures 7-11, 7-12, and 7-13 respectively. Observe that Bushra Abu Shqeer has been shifted from the inactive lists to the active lists in both Omar and Naim forms.

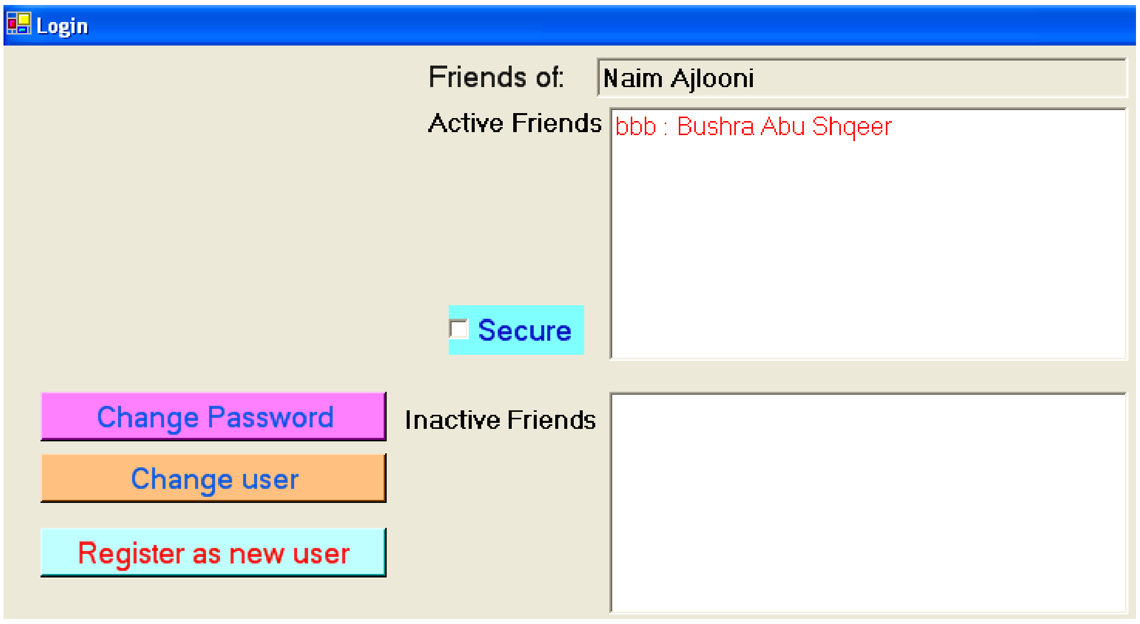


Friends of: Omer Abu Shqeer

Active Friends bbb : Bushra Abu Shqeer

Inactive Friends mmm : Mosab Abu Shqeer

Figure 0-11 : Interface form of the user "Omer" logged in and one of his friends is active

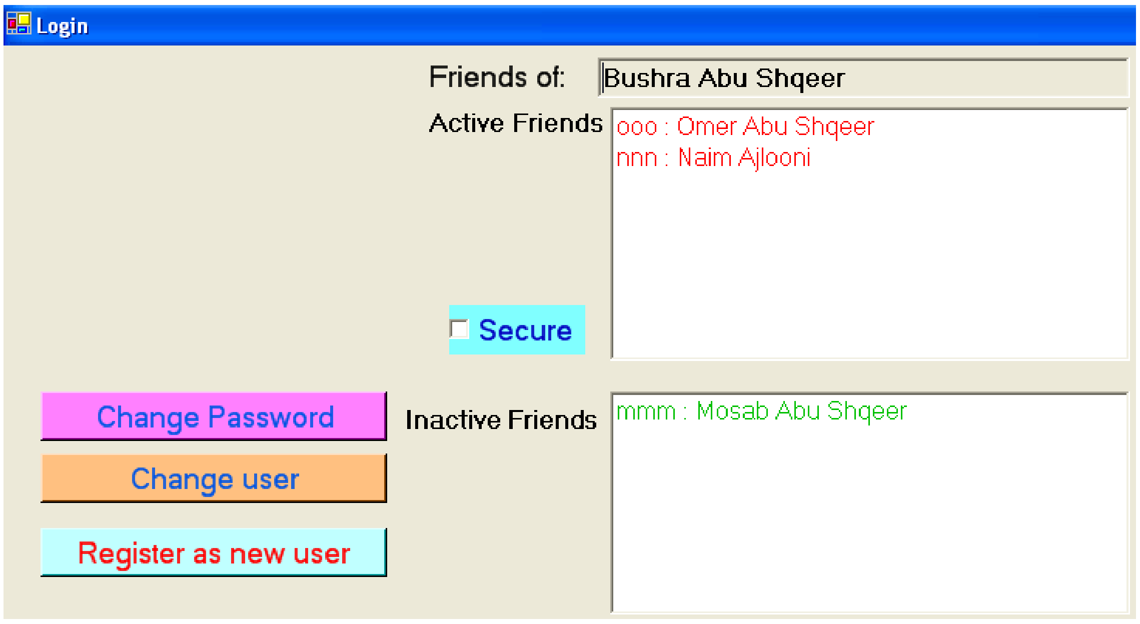


Friends of: Naim Ajlooni

Active Friends bbb : Bushra Abu Shqeer

Inactive Friends

Figure 0-12 : Interface form of the user "Naim" logged in and one of his friends is active

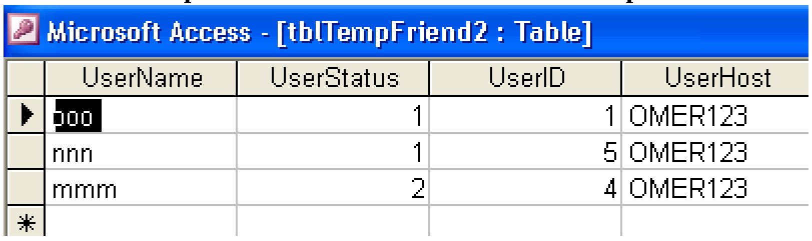


Friends of:	Bushra Abu Shqeer
Active Friends	ooo : Omer Abu Shqeer nnn : Naim Ajlooni
Inactive Friends	mmm : Mosab Abu Shqeer

Figure 0-13 : Interface form of the user "Bushra" logged in and two of her friends are active

Table 0-9 illustrates the table "tblTempFriend" of the user "Bushra" after she has logged in. Observe that it contains information for the user friends (both active and inactive). This information is necessary to make communication between this user and his/her friends.

Table 0-9 : Sample rows of the database table "tblTempFriend2"


Microsoft Access - [tblTempFriend2 : Table]

UserName	UserStatus	UserID	UserHost
ooo	1	1	OMER123
nnn	1	5	OMER123
mmm	2	4	OMER123

7.3.4. User Logout

Procedure: User Logout

Executed by: Subscriber

Location: Client

Time: Any time while a user is already logged in.

Input: Click on the close button ☒

Output:

1. User status on the table "tblSubscriber" will be updated.
2. The table "tblTempFriend" of this user will be dropped.
3. The socket will be closed.
4. A logout message should be send to all active friends of this user.
5. All active friends who receive the logout message should update their friends' lists as well as their "tblTempFriend" tables.

Algorithm steps:

1. Click on the close button ☒

2. Update user status on the table " tblSubscriber" to be inactive.

3. Send logout message to all active users of this user.

4. Drop "tblTempFriend" of this user from his ClientDB.

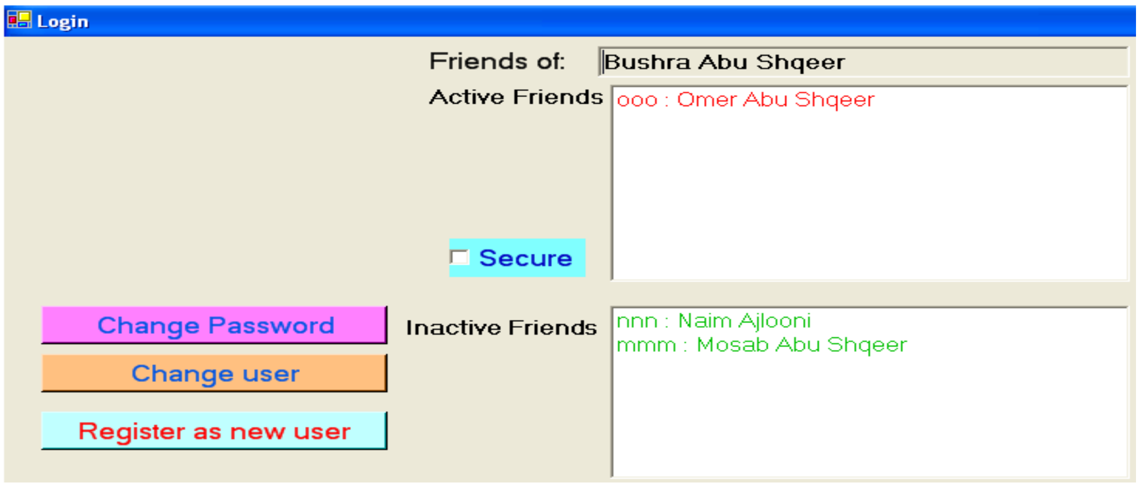
5. Any friend receives the logout message should update his/her friends' lists and his/her "tblTempFriend" table.

6. END

7.

Example 7.4

Assume that Figures 7-11, 7-12, and 7-13 represent the current state. If the user "Naim Ajlooni" logout then the interface form of the user "Bushra Abu Shqeer" becomes as shown in Figure 0-14. Observe that the user "Naim Ajlooni" has been shifted from the active to the inactive list.



Friends of: Bushra Abu Shqeer

Active Friends: ooo : Omer Abu Shqeer

Inactive Friends: nnn : Naim Ajlooni
mmm : Mosab Abu Shqeer

Figure 0-14 : Interface form of the user "Bushra" after one of her active friends logged out

7.3.5.Do conversation

Procedure: Do conversation

Executed by: Subscriber

Location: Client

Time: Any time both users are logged in.

Input:

1. The initiator user double-clicks on a friend's user name from the friends' active list; this action sends a request message to the second user.
2. The second user responses to the request.

3. During the conversation, a user can activate/inactivate the security process via the secure check box on the form.
4. A user can finish the conversation by clicking on the stop button.

Output:

1. A request message from the initiator user to the second user.
2. A response message from the second user to the first one.
3. Bi-directional conversation messages flow between the two users.
4. A termination message from the user who clicks on the stop button to the other user.

Comments: The conversation takes place only if the response of the target user is OK.

Algorithm steps:

1. A user double-clicks on a user name from his friends' active list.

2. Reads the friend's record from his "tblTempFriend" table.

3. A request message includes the user name is sent to the target user.

4. The target user sends a response message.

5. If the response message is OK then repeat until one of the users clicks on the stop or close button; each of the two users[38]:

  5.1. Changes his/her channel status to be "BUZY".

  5.2. Prepares a speech buffer for recording/capturing.

  5.3. Speaks and records the speech to the buffer.

  5.4. If the secure check box is ON then

[38] Steps 5.1 through 5.5 are done at the sender side while steps 5.6 through 5.9 are done at the receiver side.

```
6.      5.4.1. Selects an encryption key randomly from the master key as
               described in chapter 4.
        5.4.2. Encrypts the speech data as described in chapter 4 using the
               above selected key.
        5.4.3. Mixes the ciphered data with the key information as described
               in chapter 5.
    5.5. Sends the voice/mixed data to the other user.
    5.6. The voice/mixed data is received by the other user.
    5.7. If the received data was encrypted then
        5.7.1. Splits the encryption key information from the ciphered data as
               described in chapter 5.
        5.7.2. Extracts the encryption key as described in chapter 6.
        5.7.3. Decrypts the ciphered data as described in chapter 6.
    5.8. Prepares a speech buffer for playing and copies the received voice
         data to it.
    5.9. Plays the buffer and listen.
    ELSE
       The conversation is cancelled.
7.  If a user clicks on the close button then
    6.1.1. He/she Changes his/her channel status to be "FREE".
    6.1.2. Sends a close message to the other user.
    6.1.3. Stops talking.
```

8. **6.1.4.** When the other user receives the close message he/she changes
his/her channel status to "FREE" and stops talking.

9. END

The algorithms presented in this chapter and the preceding chapters form the core of the practical part of this research. These algorithms are considered enough to test, validate, and evaluate the proposed approach. A number of vital details of these algorithms were not mentioned. Also, some necessary algorithms have been implemented but not presented in this document. Some of the algorithms required the production of a complete comprehensive system; examples are:

- Search algorithm is needed to find out a user out of the friends list in order to call him.
- Add a friend algorithm is needed to add a user to the friends list of a user.
- Delete a friend algorithm is needed to delete a user from the friends list of a user.
- Chatting rooms algorithm(s) is/are needed to create chatting rooms to allow many users to communicate with each other at the same time.

The implementation of the mentioned algorithms and some others are out of the scope of this research. The goal of the practical part of this research was achieved by the implementing a prototype.

7.4.System Requirements

To install the implemented prototype, you need:

Hardware

- A PC to be used as server; preferable 1.5+ GHz, 256+ MB RAM, 40+ GB HDD, sound card, and network card.
- Two PC's to be used as clients with a similar specification as the server above; you can use the server to be one of them.
- Fast Ethernet network (IEEE 802.3u).
- Speakers and microphones / headphones.

Software

- Windows 2000 or XP.
- Visual Studio.Net [39]
- MS-ACCESS
- Microsoft DirectX 9.0 SDK [40]

[39] A lot of references have been used for programming in VB.Net such as [Reid 2004], [Gefen & Govindarajulu 2004], [Deitel, Deitel, Nieto 2002], and [Duncan & Kent]; some web sites have been explored too such as http://www.vbfreecode.com

[40] This includes direct sound; "Direct sound is specifically designed for shorter pieces of audio data. Direct sound's main advantage is its speed and its 3D audio effects." [Sink 2002, p.250].

Conclusions and Future work

8.1.Conclusions

Most of the available encryption/decryption techniques are not suitable to be used to secure voice data over an open network since they were originally built for text data, and due to their extensive computations which result in an unacceptable delay. This research attempts to develop a new encryption approach which adds a minimum delay time to the voice packets end-to-end delay. The distribution of the encryption keys is usually carried out through a trusted agency; this results in a significant delay before the conversation starts. This research attempts also to provide a new method of key exchange without an intermediate party.

A new approach for key selection and distribution has been developed; a key is selected in a random fashion from a pool of master keys; enough information about this key is mixed with the voice data; the target receiver –and the only one - is able to extract this key. Simple substitutions and transformations are used to encrypt the voice data in order to minimize the computation cost; the security level is increased by using a new key with every new packet.

The proposed solution has been implemented and tested; the conclusions and recommendations of the researchers are as follows:

1. Most of the generated keys using the proposed method passed the frequency, serial, poker, and autocorrelation statistical tests, and none of the keys failed in all tests or passed only one test.

2. The average time required to carry out the overall security process at both sides (sender and receiver) is affected by the voice packet size; larger packet size requires more processing time. If the packet size is doubled, then the security processes time is almost doubled too. Therefore, it is recommended to keep the packet size small enough in order to minimize the required time for the overall process keeping in mind the overload that comes from packet headers.
3. The length of the master key does not affect the amount of time needed to carry out the security processes; therefore it is recommended to enlarge the master key length as this increases the possible number of keys that can be generated from it.
4. The average time required to carry out the overall security process at both sides (sender and receiver) in the proposed solution as shown in Table 0-1 is very small compared to the average encryption and decryption time required by known algorithms such as AES_Rijndael as shown in Tables 3-5 and 5-3).
5. Keys of any length can be used in the proposed solution; therefore it is recommended to use a large range of key length since this strengths the security level.
6. The proposed solution produces a huge number of possible keys for each user; this number depends on different factors[41], and it grows quickly as values of the different factors increase (see Tables 6-4 and 6-5). Increasing the values of these factors does not add any significant amount of time to the overall process; therefore it is recommended to increase these values.
7. It is impossible for an unauthorized user to identify a voice message using brute-force attack assuming that half of the permutations are to be tried to reach the goal as presented in Table 0-4 and Table 0-5.

[41] See Equation 0.1 and Equation 0.2

8. It is possible to apply the proposed solution for VoIP on variety network configurations as presented in chapter 7; (see examples 6.3 to 6.6).

8.2.Future work

Even though, the research in this dissertation proved that the proposed solution enhanced the security of VoIP and can be applied on variety network configurations; there is further research that can be done to enhance or support the presented solution such as:

1. A comparison between the security timing cost of the proposed solution and a known cryptosystems.
2. Extra factors can be added to the key selection algorithm in order to increase the possible number of keys and hence increase the security level.
3. New method(s) of mixing the key information with the voice data can be used.
4. Test if the proposed solution can be used to secure video over internet protocol.
5. Analyze the effects of applying the proposed solution in IPv6 environment.

References

1 Angus, Ma. (2001). Voice over IP (VoIP), (on-line). Available: http://www.spirentcom.com/documents/100.pdf

2 Ahuja, S. and Ensor, R. (2004). VoIP, What is it good for? (on-line). Available: http://www.acmqueue.com. ACM Queue, vol.2; no. 6; Sep 2004. PP: 49-55.

3 Arcomano, R. (2002). VoIP How to (on-line). Available: http://www.tldp.org/HOWTO/VoIP-HOWTO.html#toc1

4 Balliache, L. (2003). Voice over IP (on-line). Available: http://opalsoft.net/qos

5 Barlow, G. (ND). (Cited on Feb, 2006). Traffic Delay in Ethernet over SONET/SDH (on-line). Available: http://www.innocor.com/pdf_files/delay_in_EoS.pdf

6 Boger, Y. (ND). Fine-tuning Voice over Packet services. (on-line). Available: http://www.protocols.com/pbook/pdf/voip.pdf

7 Cisco, Document ID: 5125. (2006).Understanding delay in packet voice networks. (on-line) available: http://www.cisco.com/warp/public/788/voip/delay-details.html.

8 Collier, M. (ND). VoIP Vulnerabilities and Solutions (on-line). DATA COMMUNICATION MANAGEMENT, 51-40-2. PP: 1-15.

9 comdial.com (ND). VoIP technical training – IP networking installation, (on-line). Available: http://www.comdial.com/portal/training/voip/voip_networkinstall.pdf

10 Davidson, J. and Peters, J. (2004). Voice over IP fundamentals – a systematic approach to understanding the basics of Voice over IP. Eighth print. Indianapolis, USA. Cisco Press.

11 Deitel, H., Deitel, P., and Nieto, T. (2002). Visual Basic.NET How to Program, 2nd ed. US. Prentice Hall.

12 Downey, J.J. (2005). Understanding VoIP Packet Sizing and Traffic Engineering. SCTE CABLE-TEC EXPO 2005. (on-line) Available: http://www.cisco.com/application/pdf/en/us/guest/netsol/ns4/c654/cdccont_0900aed803956d3.pdf

13 Duncan, M. and Kent, S. (2002). Teach yourself Visual Basic.NET in 21 Days. US, SAMS Publishing.

14 FIPS PUB 140-2, NIST (2001). Security Requirements for Cryptographic Modules. (on-line) available: http://www.us.design-euse.com/exit?url=http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

15 Gefen, D. and Govindarajulu, C. (2004). Advanced Visual Basic.NET Prgramming Web and Desktop Applications in ADO.NET and ASP.NET, 1st ed. New Jersy, USA. Prentice Hall.

16 Jarrar, H. J. (2004). Image encryption and decryption with residual intelligibility measurements, doctoral dissertation. The Arab Academy for Banking and Financial Sciences, Amman-Jordan.

17 ITU-T Recommendation G.114 (on-line). Available: http://ftp.tiaonline.org

18 Keagy, S. (2000). Integrating voice and data networks – Practical solutions for the new world of packetized voice over data networks. Indianapolis, USA. Cisco Press.

19 Kostas, J; Borella, M.; Sidhu, I.; Schuster, M.; Grabiec, J; and Mahler, J. (1998). Real-Time voice over packet-switched networks (on-line).Available: http://www.borella.net/mike/academics/voip.pdf. IEEE network Jan/Feb 98. PP:18-27.

20 Kuhn, D.; Thomas, J.; Walsh; and Fries, S. (2005). Security Considerations for Voice over IP Systems, Recommendations of the National Institute of Standards and Technology, (on-line). Available: http://csrc.nist.gov/publications/nistpubs/800-58/SP800-58-final.pdf NIST, Special Publication 800-58, Jan 2005.

21 Newport networks. (2005). VoIP Bandwidth Calculation (on-line). Available: http://www.newport-networks.com

22 Nortel networks. (2001). Voice over packet, an assessment of voice performance on packet networks (on-line). Available: http://nortel.com/products/library/collateral/74007.25-09-01.pdf

23 obviex.com (2002). How To: Encrypt and Decrypt Data Using a Symmetric (Rijndael) Key (C#/VB.NET), (on-line). Available: http://www.obviex.com/samples/Encryption.aspx

24 Pfleeger, C. P. and Pfleeger, S. L. (2003). Security in Computing, 3rd ed. New Jersy, USA. Prentice Hall.

25 Pietrosemoli, E. (2004). Voice over IP (on-line).Available: www.eslared.org.ve

26 Al Rashed, A.A. (2004). Intelligent encryption decryption system using limited genetic algorithm and Rijndael algorithm. Unpublished doctoral dissertation. The Arab Academy for Banking and Financial Sciences, Amman-Jordan.

27 Reid, F. (2004). Network Programming in .NET With C# and Visual Basic.NET, 1st ed. USA. Elsevier digital press.

28 Sharif, M. (2004). Voice privacy service over the public telephone network. Published doctoral dissertation. George Mason University, Fairfax, Virginia.

29 Sherburne, P. and FitzGerald, C. (2004). You don't know Jack about VoIP (on-line). Available: http://www.acmqueue.com. ACM Queue, vol.2; no. 6; Sep 2004. PP: 31-38.

30 Shuman, J. (2003). Multiple packet-streams in encrypted voice over IP. Published master dissertation. Carleton University, Canada.

31 Sink, K. (2002). DirectX 8and Visual Basic Development. US. SAMS Publishing.

32 Stallings, W. (2003). Cryptography and network security – principles and practices, 3rd ed. New Jersy, USA. Prentice Hall.

33 Stallings, W. (2004). Computer Networking with Internet Protocols and Technology. New Jersy, USA. Prentice Hall.

34 SUNYIT. (2003). Performance consideration (on-line). Available: http://www.tele.sunyit.edu/0130224634.pdf

35 Tanenbaum, A.S. (2003). Computer Networks, 4th ed. New Jersy, USA. Prentice Hall.

36 Uday O. and Pabrai, A. (2004). The challenge of VoIP Security (on-line).available: http://www.certmag.com/articles/templates/cmag_techniques.asp?articleid=957&zoneid=95. Certification Magazine Nov, 2004.

37 Walker, J.Q. (2001). A handbook for successful VoIP deployment: Network Testing QoS, and more (on-line). Available: http://whitepapers.zdnet.co.uk NetIQ corporation.

38 Zeadally, S.; Siddiqui, F.; and Kubher, F. (2004). Voice over IP in intranet and internet environments (on-line). Available: http://www.nortelnetworks.com. IEE Proceedings communications, vol. 151, No. 3, June 2004, pp: 262-269.

39 Zimmerman, P. (1999). An Introduction to Cryptography (on-line). Available:citeseer.ist.psu.edu