

# **Improved Performance of Secured VoIP Via Enhanced Blowfish Encryption Algorithm**

by

**Amzari Jihadi Ghazali**

Submitted in accordance with the requirements for the award of the  
degree of Master of Philosophy of the University of Liverpool

JUNE 2019

# ABSTRACT

Both the development and the integration of efficient network, open source technology, and Voice over Internet Protocol (VoIP) applications have been increasingly important and gained quick popularity due to new rapidly emerging IP-based network technology. Nonetheless, security and privacy concerns have emerged as issues that need to be addressed. The privacy process ensures that encryption and decryption methods protect the data from being alternate and intercept, a privacy VoIP call will contribute to private and confidential conversation purposes such as telebanking, telepsychiatry, health, safety issues and many more. Hence, this study had quantified VoIP performance and voice quality under security implementation with the technique of IPSec and the enhancement of the Blowfish encryption algorithm. In fact, the primary objective of this study is to improve the performance of Blowfish encryption algorithm. The proposed algorithm was tested with varying network topologies and a variety of audio codecs, which contributed to the impact upon VoIP network.

A network testbed with seven experiments and network configurations had been set up in two labs to determine its effects on network performance. Besides, an experimental work using OPNET simulations under 54 experiments of network scenarios were compared with the network testbed for validation and verification purposes. Next, an enhanced Blowfish algorithm for VoIP services had been designed and executed throughout this research. From the stance of VoIP session and services performance, the redesign of the Blowfish algorithm displayed several significant effects that improved both the performance of VoIP network and the quality of voice. This finding indicates some available opportunities that could enhance encrypted algorithm, data privacy, and integrity; where the balance between Quality of Services (QoS) and security techniques can be applied to boost network throughput, performance, and voice quality of existing VoIP services. With that, this study had executed and contributed to a threefold aspect, which refers to the redesign of the Blowfish algorithm that could minimize computational resources. In addition, the VoIP network performance was analysed and compared in terms of end-to-end delay, jitter, packet loss, and finally, sought improvement for voice quality in VoIP services, as well as the effect of the designed enhanced Blowfish algorithm upon voice quality, which had been quantified by using a variety of voice codecs.

# ACKNOWLEDGEMENTS

Alhamdulillah, all praise to Allah Subhanahu Wa Ta'ala for his blessings and mercy that gave me the strength to accomplish this research. Besides, I would like to express my gratitude to my sponsor, Majlis Amanah Rakyat (MARA) or the Council of Trust for the People, an agency under the purview of the Ministry of Rural and Regional Development, as well as my employer, MARA Professional College Indera Mahkota, Kuantan, Malaysia, which hosted me and provided the necessary funding and resources required for this research work.

Most importantly, I would like to thank Dr Waleed Al Nuaimy and Dr Ali Al Ataby, who acted sequentially as supervisors, as well as to Professor Majid Al Taei from Kingston University London as a visiting Professor at the Department of Electrical Engineering and Electronics, whose encouragement, guidance, and support had been helpful throughout the whole study period. Moreover, I would never forget to acknowledge the great support given by Professor Asoke Nandi, who guided me at the initial phase of my research, for his vast knowledge in the field and contributions at all levels, as well as being always available for discussion sessions. I also would love to thank all my colleagues in the research group for being there, for their unconditional support, and for the great fun we had together along these years.

Many thanks go to my parents, who made many sacrifices to support me throughout the period of my professional development. Finally, I would want to thank my wife, Dr Hafizan Mat Som, my sons, Aiman Hakeemi, Ameer Amsyar, Akif Ashraf, and Afiq Zareef, who understood very well the ups and downs of research and managed to cheer me up at all times. Their dedication and support has been infinite, and I am forever grateful towards them for being always there for me. Allahuakbar.

# TABLE OF CONTENTS

Abstract .....	i
Acknowledgements .....	ii
Table of Contents .....	iii
List of Abbreviations and Symbols.....	vii
List of Figures .....	xii
List of Tables .....	xv
Chapter 1 .....	16
Introduction.....	16
1.1    Background .....	16
1.2    Problem Statement .....	19
1.3    Aim and Objectives.....	20
1.4    Contribution .....	21
1.5    Thesis Outline .....	21
1.6    Published Work And Publication Plan .....	23
Chapter 2 .....	24
Background .....	24
2.1    Voice Over Internet Protocol (VoIP).....	24
2.1.1    VoIP Protocols .....	25
2.2    VoIP Traffic Metrics .....	31
2.3    VoIP Security Issues .....	34
2.3.1    VoIP Security Threats.....	34
2.3.2    Internet Protocol Security (IPSec) .....	36
2.3.3    VoIP Encryption and algorithm .....	41
2.3.4    Blowfish Encryption Algorithm .....	44
2.4    VoIP Audio Codecs .....	48
2.5    VoIP Voice Quality Assessment.....	51

2.5.1	Mean Opinion Score (MOS) .....	52
2.5.2	E-Model .....	53
2.6	Revisiting Previous Work .....	59
2.6.1	VoIP QoS and VoIP Security .....	59
2.6.2	Voice Codecs and Voice Quality Measurement .....	61
2.6.3	Enhances Blowfish Encryption Algorithm .....	62
2.7	Summary .....	63
Chapter 3 .....		65
Methodology .....		65
3.1	Experimental and Preparation .....	65
3.2	Encryption Algorithm Enhancement .....	69
3.3	Network Testbed Design .....	70
3.4	Network Simulation Tools .....	75
3.4.1	OPNET/Riverbed Simulation .....	75
3.4	Summary .....	76
Chapter 4 .....		78
Blowfish Encryption Algorithm Enhancement and Implementations .....		78
4.1	First Stage: Algorithm Experiment & Simulation .....	79
4.1.1	Algorithm design and test by using C++ simulation .....	79
4.1.2	S-Box Design and Enhancement: .....	86
4.1.3	Performance metrics .....	87
4.1.4	Execution time (encryption and decryption) .....	87
4.1.5	Throughput .....	87
4.1.6	CPU Process time and memory usage .....	88
4.1.7	Cryptanalysis/ Security measurement .....	88
4.1.8	Avalanche Effect .....	88
4.1.9	Correlation Coefficient .....	89
4.2	Second Stage : Network Design and Testbed : .....	89
4.3	Third Stage : VoIP clients and Monitoring Tools .....	92
4.3.1	SIP Call setup .....	93

4.4	Fourth Stage: OPNET, Codecs and Network performance. ....	94
4.4.1	OPNET Network design. ....	96
4.4.2	Application Configuration .....	97
4.4.3	Application Specification.....	97
4.4.4	Voice Encoder Schemes.....	98
4.4.5	Profile Configuration .....	98
4.5	Fifth Stage: Data Collection and Processing .....	98
4.6	Sixth Stage: Validation and Verification .....	99
4.7	Summary .....	99
Chapter 5 .....		100
Results, Analysis and Comparison .....		100
5.1	Simulation Results .....	100
5.1.1	Algorithm performance (Execution Time & Throughput) .....	101
5.1.2	Different packet types (Video & Text) .....	102
5.1.3	Cryptanalysis Results/ Security Analysis .....	105
5.1.4	Avalanche effect .....	105
5.1.5	Correlation Coefficient .....	107
5.2	Network Testbed & OPNET Simulations Results .....	109
5.2.1	Network QoS Parameters.....	109
5.2.2	Average delay .....	111
5.2.3	Average Jitter .....	112
5.2.4	Packet Loss Ratios .....	113
5.2.5	Voice Quality MOS and E Model values .....	115
5.3	Validation and Verification.....	116
5.3.1	Impacts of Network Delay, Packet Loss on The Factor R.....	117
5.3.2	MOS Scores .....	123
5.3.3	Encryptions Verification .....	126
5.4	Summary .....	127
Chapter 6 .....		129
Conclusion and Future Work .....		129
6.1	Conclusion .....	129

6.2 Potential Research Works .....	133
References .....	134
Appendix A .....	148
C++ Enhance Blowfish Algorithm.....	148
IPSec VPN (OpenSwan) Enhance Blowfish Algorithm: .....	168
Appendix B .....	174
Network Testbed Specifications.....	174
Appendix C .....	176
OPNET Configurations & Figures .....	176

# LIST OF ABBREVIATIONS AND SYMBOLS

AES	- Advanced Encryption Standard
AH	- Authentication Header
AMR-NB	- Adaptive Multi-Rate Narrow Band
AMR-WB	- Adaptive Multi-Rate Wide Band
ARPA	- Advanced Research Projects Agency
ATM	- Asynchronous Transfer Mode
BPL	- Packet loss robustness
BW	- Bandwidth
CBC	- Cipher Block Chaining
CBR	- Constant Bit Rate
CODEC	- Coder/decoder
CM	- Conventional Method
CPU	- Central Processing Unit
CS	- Circuit- Switched
CSNs	- Circuit-Switched Networks
DES	- Data Encryption Standard
DH	- Diffie-Hellman
DiffServ	- Differentiated Services
DNS	- Domain Name Server
DoS	- Denial of Services
DSCP	- Differentiated Services Code Point
ESP	- Encapsulated Security Payload
FR	- Frame Relay
FTP	- File Transfer Protocol
GSM	- Global System of Mobile Communications



GUI - Graphical User Interface

HDR - Packet Header

HTTP - Hypertext Transfer Protocol

ICV - Integrity Check Value

IDE - Integrated Design Environment

IEEE - Institute of Electrical and Electronic Engineers

IETF - Internet Engineering Task Force

IKE - Internet Key Exchange

IntServ- Integrated Services

IP - Internet Protocol

IPSec - IP Security, a set of protocol developed by the IETF to support secure exchange of packets at the IP layer

IPTV - Internet Protocol Television

ISAKMP - Internet Security Association and Key Management Protocol

ISO - International Organization for Standardization

ISP - Internet Service Provider

ITU - International Telecommunication Union

ITU-R ITU - Radio communication Sector

ITU-T ITU - Telecommunication Standardization Sector

ITU - International Union of Telecommunications

LAN - Local Area Network

MBps - Megabytes Per Second

Mb - Megabit

MD5 - Message Digest 5

MIME - Multi-purpose Internet Mail Extensions

MitM - Man-in-the-Middle

MODP- Modular Exponential

MOS - Mean Opinion Score

MOScq - MOS Conversational Quality Estimated

MPLS - Multi protocol Label Switching

NAV - Network Allocation Vector

NC - Network Coding method

NCD - Network Coding Decision

NVP - Network Voice Protocol

OS - Operation System

OSI - Open System Interconnection

PAMS - Perceptual Analysis Measurement System

PBX - Private Branch Extensions

PC - Personal Computer

POP' - Point Coordination Punction

PCM - Pulse Code Modulation

PDF - Probability Distribution Function

PER - Packet Error Rates

PESQ - Perceptual Evaluation Speech Quality

PKI - Public Key Infrastructure

PL - Payload

PLC - Packet Loss Concealment

PLOP - Physical Layer Convergence Protocol

POTS - Plain Old Telephony System

Ppc - Packets Per Conversation

PPDU - PLCP Data Unit

PPP - Point to Point

PSN - Packet Switched Network

PSTN - Public Switched Telephone Network

PSQM - Perceptual Speech Quality Measures

QDU - Quantization Distortion Units

QoS - Quality of Service

RFC - Request for Comments

RLR - Receiver Loudness Rating

RPE-LTP - Regular-pulse-excited long-term prediction

RS - Redirect Server

RSVP - Resource Reservation Protocol

RTCP - Real Time Control Protocol

RTP - Real time Transport Protocol

RTS - Request to Sent

RTT - Round Trip Time

SA - Security Association

SAD - Security Association Database

S-Boxes - Substitution Boxes

SDP - Session Description Protocol

SFD - Start of Frame Delimiter

SHA - Secure Hash Algorithm

SIFS - Short Inter-frame Space

SIP - Session Initiation Protocol

SLR - Sender Loudness Rating

SOHO - Small Office/Home Office

SPD - Security Policy Database

SPI - Security Parameter Index

SRTP - Secure Real-Time Transport Protocol

TCP - Transmission Control Protocol

VAD - Voice Activity Detection

VLAN - Virtual Local Area Network

VPN - Virtual Private Network

TLS - Transport Layer Security

UDP - User Datagram Protocol

UMTS - Universal Mobile Telecommunications System

URI - Uniform Resource Identifier

VoIP - Voice over Internet Protocol

WAN - Wide Area Network

WCDMA - Wideband Code Division Multiple Access

XML - eXtensible Markup Language

# LIST OF FIGURES

Figure 1: VoIP IPSec tunnel transmission [180] .....	18
Figure 2: SIP architecture overview.....	30
Figure 3: VoIP security threats [181].....	34
Figure 4: IPSec interrelationships [80] .....	37
Figure 5: IPSec AH packet diagram [74].....	39
Figure 6: IPSec ESP packet diagram [75].....	40
Figure 7: VoIP security challenges and threats.....	42
Figure 8: 3DES encryption functional [182].....	43
Figure 9: Blowfish Feistel structure [109] .....	45
Figure 10: Blowfish Feistel function [92].....	46
Figure 11: Operation of Blowfish Algorithm .....	47
Figure 12: VoIP codecs overview .....	49
Figure 13: Call and voice quality assessment .....	51
Figure 14: MOS vs Delay per codec .....	58
Figure 15: MOS vs Jitter per codec .....	58
Figure 16: MOS vs Packet loss per codec .....	59
Figure 17: Research methodology .....	66
Figure 18: Parameters affect VoIP QoS.....	67
Figure 19: Standard Blowfish encryption and enhanced Blowfish algorithms measurement .....	69
Figure 20: Network testbed diagram.....	72
Figure 21: Physical network testbed .....	73
Figure 22: Specific devices .....	74
Figure 23: VoIP quality measurement .....	77
Figure 24: Blowfish Feistel Structure of 16.....	80

Figure 25: Blowfish algorithm encryption process flow .....	81
Figure 26: Graphic representation of F function in Blowfish algorithm .....	82
Figure 27: Modification F function of enhanced Blowfish algorithm .....	83
Figure 28: Network testbed.....	90
Figure 29: Ekiga SIP based VoIP call process [183] .....	93
Figure 30: OPNET network design.....	94
Figure 31: Simulation VoIP enterprise network .....	96
Figure 32: Execution time for standard and enhanced Blowfish algorithms in second (s) for audio packet.....	101
Figure 33: Throughput (MB/s) for standard and enhanced Blowfish algorithms for audio packet .....	102
Figure 34: Throughput (MBytes/s) for standard and enhanced Blowfish algorithms for video packet .....	103
Figure 35: Execution time for standard and enhanced Blowfish algorithms in second (s) for video packet.....	103
Figure 36: Execution time for standard and enhanced Blowfish algorithms in second (s) for text packet .....	104
Figure 37: Throughput (MBytes/s) for standard and enhanced Blowfish algorithms for text data.....	104
Figure 38: Avalanche effect percentage with 64-bit key changes on plain text .....	106
Figure 39: Avalanche effect percentage with 64-bit key changes on secret key .....	106
Figure 40: Comparison of avalanche effect for both standard and enhanced Blowfish algorithms .....	107
Figure 41: Correlation coefficient for both standard and enhanced Blowfish algorithms with 64 bits key.....	108
Figure 42: Average of correlation coefficient 64-bit key cipher text for both standard and enhanced Blowfish algorithm. ....	108
Figure 43: VoIP calls monitoring and network testbed .....	110
Figure 44: Comparison of average delay .....	111

Figure 45: Comparison of average jitter .....	113
Figure 46: Packet loss ratio for voice codecs without encryption .....	114
Figure 47: Packet loss ratio for voice codecs with standard Blowfish encryption .....	114
Figure 48: Packet loss ratio for voice codecs with enhanced Blowfish encryption.....	115
Figure 49: MOS values for voice codecs under different scenarios .....	116
Figure 50: Equipment impairment factor (I <sub>e</sub> ) versus packet loss for G.711 codec .....	118
Figure 51: Equipment impairment factor (I <sub>e</sub> ) versus packet loss for G.729 codec .....	119
Figure 52: Equipment impairment factor (I <sub>e</sub> ) versus packet loss for AMR-NB codec .....	119
Figure 53: Relationship between I <sub>d</sub> factor and T <sub>a</sub> delay .....	120
Figure 54: Network delay versus factor R for G.711 codec .....	122
Figure 55: Network delay versus factor R for AMR-NB codec .....	122
Figure 56: Network delay versus factor R for G.729 codec .....	123
Figure 57: Comparison of G.711 codec based on varying scenarios.....	125
Figure 58: Comparison of G.729 codec based on varying scenarios.....	125
Figure 59: Comparison of AMR-NB codec based on varying scenarios.....	126
Figure 60: Unencrypted VoIP services .....	127
Figure 61: Encrypted VoIP services .....	127
Figure 62: Application Profile Configuration.....	176
Figure 63: Application Configuration Attributes.....	176
Figure 64: Codec Scheme .....	177
Figure 65: Profile Configuration.....	177
Figure 66: Profile Attributes .....	177

# LIST OF TABLES

Table 1: SIP Messages [45] .....	30
Table 2: ITU recommended values for VoIP quality.....	32
Table 3: QoS requirements per application (Low, Medium, High) [50] .....	33
Table 4: Comparison of VoIP codecs [117] .....	49
Table 5: MOS scale for subjective assessment [126] .....	52
Table 6: Recommended values for the advantage factor .....	55
Table 7: R to $MOS_{CQE}$ correspondence for estimative assessment .....	57
Table 8: Correlation between the methods, tools and parameters .....	68
Table 9: Network testbed experiment with different voice codec .....	91
Table 10: Network testbed experiment with different background traffic.....	92
Table 11: Properties of OPNET simulation scenarios .....	95
Table 12: Comparison of average delay from G.711, G.729, and AMR-NB codecs ..	111
Table 13: Comparison of average jitter from G.711, G.729, and AMR-NB codecs ...	112
Table 14: MOS values for voice codecs .....	115
Table 15: Comparison of MOS from G.711, G.729, and AMR-NB with and without encryption algorithms under different calls setup.....	124
Table 16: Device Specification .....	174
Table 17: Network IP Addresses .....	175
Table 18: Site-to-site IPSec tunnel .....	175



# CHAPTER 1

## INTRODUCTION

### 1.1 BACKGROUND

Since the success of voice transmission initiated by Alexander Graham Bell, who also invented the very first functional telephone [1], investigations related to communication technologies have never halted, but have continued into this modern 21<sup>st</sup> century era. In the past telephony times, voice transmission implied guaranteed services provided by dedicated communication media, known as circuits. In parallel with the existence of Circuit-Switched (CS) telephony systems, another concept of voice transmission emerged - Voice over Internet Protocol (VoIP). VoIP functions by converting human voice into a stream of digital data that is packetized into Internet Protocol (IP) packets and sent over the network to the other end[s] of the call [2].

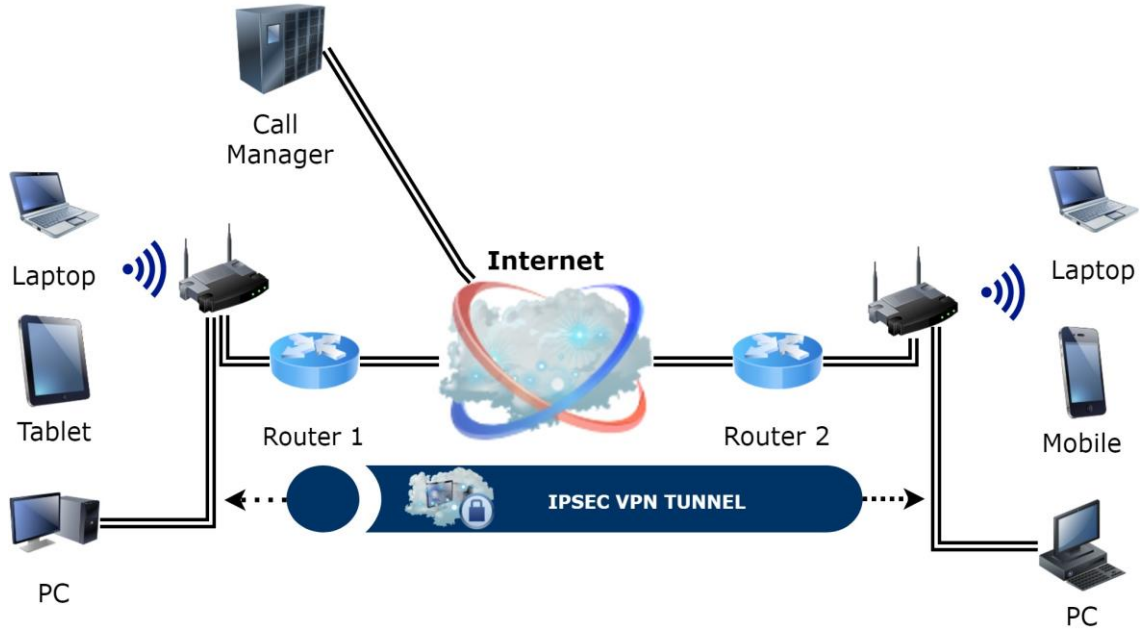
Furthermore, VoIP technology has introduced newly-integrated and cost-efficient services, along with intricacies [3]. This new technology can substantially increase traffic load, yet it can lead to technical problems if not addressed at the installation, operational and maintenance stage. One possible but costly solution is to set up an entirely new data network dedicated to VoIP. A more sensible approach, however, is to combine the data network with the VoIP service that demands a monitoring system if customer satisfaction is made priority.

At present times, VoIP services are being used widely in voice communication and as such, many studies pertaining to VoIP have focused on VoIP security, for example, the implementation of encryption and steganography as some methods to secure the communication between both parties in VoIP communication [4]. In addition, the illustration presented in Figure 1 displays a VoIP communication, where customers are connected via secured VoIP technologies; which represents a secure VoIP call from a sender to a connected receiver, along with additional encryption techniques by using IPSec so as to offer protection against intruders, to ascertain information (conversation) privacy, and to ensure the integrity of data transmission between both parties. Furthermore, the combination of security, privacy, and encryption portrayed in Figure 1

has an impact upon VoIP voice quality and performance, in terms of increment in end-to-end delay and packet loss, as well as lower network throughput. Therefore, as a network administrator, it is essential to manage and to control the performance of a network convergence before allocating the required resources. Thus, a variety of empirical, network topology and scenarios with in-depth investigations upon encryption algorithm and voice codecs analyses should offer a solution to the issues mentioned above.

As VoIP services growing day-by-day, security and privacy are turning out to be the most crucial and important aspect of it. Hence, secure voice conversation is needed to prevent any unauthorized recipient and intruder from retrieving and eavesdropping the original data. The encryption algorithm can be used along with IP Security (IPSec) to secure and privately convert voice data into unreadable form. Lots of encryption algorithms are available to be used with VoIP services, commonly Data Encryption Standard (DES), Triples DES (3DES), Advanced Encryption Standard (AES), Blowfish, RC2 and RC6 are vital in securing data and sensitive information. Nonetheless, these will consume and utilize a significant amount of computing resources such as processing time, memory, and power consumption. DES and 3DES encryptions are no longer secured as they are already obsolete and vulnerable to cryptanalysis and linear cryptanalysis attacks.

Additionally, the network design and experiments devised in this study were designed to test VoIP services under varying network topologies and voice codecs. Hence, the main purpose of this research is to investigate the performances of enhanced Blowfish algorithm and VoIP, as well as the quality of voice through the use of various voice codecs. Explicitly, this research studied the feasibility of implementing VoIP with enhanced Blowfish algorithm as the proposed security aspect. As such, VoIP performance had been tested to determine both the advantages and disadvantages of implementing the enhanced Blowfish algorithm.



**Figure 1:** VoIP IPsec tunnel transmission [180]

With the advent of VoIP, issues regarding Quality of Service (QoS) have become limelight among VoIP users [5], thus generating several solutions to overcome these irking issues [6]. In fact, the QoS refers to the requirement that guarantees good customer experience, as well as advanced feature to prioritize internet traffic. In line with that, the International Union of Telecommunications (ITU) suggested the Mean Opinion Score (MOS) [7] that offers terminology and the procedure to validate voice quality perception via assessment carried out upon a large number of customers. However, a more viable approach refers to the application of several widely used communication parameters to develop a mathematical model, especially to predict the call quality of VoIP [8]. For instance, if parameters, such as delay and packet losses, are employed, prediction of voice quality can be computed by using the E-model [9][10]. This model is useful for transmission planner to ascertain satisfaction among users concerning end-to-end transmission performance, consequently becoming apparent that phone quality estimation is indeed feasible and appears to be a significant aspect when designing more efficient network architectures [11][12]. This is also one of the main reasons for the QoS to emerge as an essential subject area in the last decade as ‘customers are always right’.

Furthermore, as managing VoIP in networks with QoS is already a challenge [13], the emergence of a secure network, while making it more attractive to customers, has increased the intricacy in allocating resources that can guarantee good quality [14].

Simultaneously, customer awareness of these security issues related to VoIP services has brought down the expectations of a good quality VoIP experience. Hence, this fact creates an opportunity for network administrators to use VoIP applications and services to offer users with greater insurance of security and privacy. With that, voice quality, particularly in this study, had been weighed in to improve the implementation of VoIP communications with secured VoIP services and networks through the use of enhanced Blowfish encryption algorithms. Furthermore, the MOS was employed to validate the voice quality of VoIP over heterogeneous networks, which has been projected to be a key feature for future network design.

## **1.2 PROBLEM STATEMENT**

The implementation of network security parameters by using Internet Protocol Security (IPSec) is a solution that secures VoIP services [15], although IPSec has been presumed to have a drawback and impact on VoIP performance. Besides, due to additional packet overheads, authentication, integrity check, encryption, and extra central processing unit (CPU) processing involved in the lengthy process, the application of encryption contributes to transmission delay and packet size overhead on VoIP [16]. These are also associated to the processing time required to encrypt/decrypt bits or blocks of data, as well as the increase in packet size due to the block size of the encryption algorithm [17].

This delay levied can be determined by the operation method applied in the encryption algorithms. As security and efficiency have become conflicting requirements, the introduction of a complex security layer that can guarantee packet content confidentiality, integrity, and authentication can slow down packet transmission [18]. Thus, the need for shorter computation time to encrypt and decrypt data would result in a higher throughput value, as well as the lower end-to-end delay, jitter, and the lowest packet loss rates [19].

Precise, acceptable voice quality and secured VoIP services are crucial for consumer, thus indicating that equity between VoIP security, performance, and voice quality need to be attained [20]. These VoIP applications and services are time-sensitive and vulnerable to network level attack, including protocol level attack. Therefore, several security mechanisms, such as IPSec, are necessary to offer a secured VoIP environment. Besides, the company's IT manager or network operator would gain benefit from the

efficient encryption algorithm, which can provide adequate security and better voice quality, even under limited bandwidth or limited resources provided by company and Internet service provider (ISP). Moreover, to date, countless voice activation operations and applications can be programmed and transferred over the Internet, in which end users and those visually-impaired could also reap benefits from this; whereby the application can be operated without consuming much bandwidth or compromising security during voice transmission and transaction.

### **1.3 AIM AND OBJECTIVES**

Therefore, this research aims to propose a novel and improved encryption algorithm that could maintain security requirements, in line with the standard of Blowfish encryption algorithm. Moreover, the performance of the encryption algorithm had been improved by reducing execution time and by increasing throughput. This research also executed the enhanced encryption algorithm in VoIP network architecture with various voice codecs to present a set of choices to ascertain better voice quality and VoIP services.

Hence, to achieve the aim, a few objectives had been listed, as follows:-

- i. To study, search and evaluate the literature related to existing encryption algorithm network testbed/ network simulator and voice codecs that can be implemented to VoIP services.
- ii. To analyze and synthesize the information gathered in that literature by identifying gaps in current knowledge.
- iii. To design and to develop an enhanced encryption/decryption algorithm based on an existing block cipher algorithm (known as Blowfish algorithm) that minimizes computational resources (i.e. execution time and throughput) without violating security requirement.
- iv. To design secure enhanced encryption algorithms based on standard Blowfish algorithm without sacrificing the privacy of users during data/voice transmission, besides maintaining the existing security level. The proposed algorithm should contribute to better end-to-end bandwidth utilization and reduction of transmission delay.

- v. To improve the performance of a VoIP network regarding the end-to-end delay, jitter, and packet loss by using the enhanced Blowfish encryption algorithm. This performance had been assessed by using various network topologies and designs.
- vi. To improve the aspect of voice quality in VoIP services and to quantify the effect of the enhanced Blowfish algorithm upon voice quality by using varied voice codecs.

#### **1.4 CONTRIBUTION**

This research highlights three contributions: -

- i. The improvement and the redesign of an encryption/decryption algorithm based on an existing block cipher algorithm (known as Blowfish algorithm) that could minimize computational resources to remain secure data, voice, and video transmissions, almost as good as the standard Blowfish algorithm.
- ii. Demonstrate that the enhanced Blowfish encryption algorithms perform better than the standard version in terms of end-to-end delay, jitter, and packet loss of VoIP network performance.
- iii. Improve the voice quality in VoIP services and quantified the effect of the developed encryption/decryption algorithm upon voice quality by using various voice codecs.

#### **1.5 THESIS OUTLINE**

Chapter 2 presents the background and the literature review associated to the investigated research topic. An overview on VoIP is presented, along with VoIP Protocol, VoIP security issues, VoIP quality measurement and parameter, Internet Protocol Security (IPSec), voice codecs, as well as reviews of prior studies related to this research.

Besides, a review of the solutions developed by other researchers aimed at enhancing the performance of VoIP and voice quality is presented towards the end of the chapter.

Chapter 3 presents the research methodology employed in this study; which is embedded in the discussion on methods used in designing encryption algorithm, as well as implementation of the algorithm on selected network design by assessing network performance, VoIP call quality, and experimental setup, together with the usage of various voice codecs.

Chapter 4 describes in detail how the research was carried out, the stages involved in this research, including the design of the enhanced Blowfish algorithm, the simulation tests using C++ programming language, the network testbed set up in computer labs involving VoIP softphone clients, and the open source software/operating system. In addition, different types of network topology and network parameters were also deployed by using the OPNET simulator, along with various voice codecs. For each deployment scenario, a complete set of simulation results had been used to validate the accuracy and the usefulness of the developed enhanced Blowfish encryption.

Chapter 5 presents the results and the solutions proposed in the context of enhancing Blowfish encryption algorithm deployment, VoIP performance evaluation without encryption, with standard Blowfish algorithm encryption, as well as the enhanced Blowfish algorithm encryption. As such, various types of voice codecs had been tested by using network testbed and OPNET simulation with both results validated and compared. Moreover, both encryption algorithms security aspects were also measured with the avalanche effect and correlation coefficient.

Lastly, Chapter 6 concludes this thesis with a summary of the solutions proposed and research limitations, followed by a discussion pertaining to potential future work.

## 1.6 PUBLISHED WORK AND PUBLICATION PLAN

- 1) A.J. Ghazali, W. Al-Nuaimy, A.K. Nandi, "Simulation of the encryption of NetFlow packet capturing system using IPSec," 5th International Conference on Computers and Devices for Communication (CODEC), vol., no., pp.1,4, 17-19 Dec. 2012.
- 2) A.J. Ghazali, W. Al-Nuaimy, A. Al-Ataby, I. Jamaludin "Comparison of Classification Models for NSL-KDD Dataset for Network Anomaly Detection," International Journal of Arts and Sciences (ISSN 1944-6934). pp. 04,01,22 -25 October 2013.
- 3) A.J. Ghazali, W. Al-Nuaimy, A. Al-Ataby, M. Al-Tae, "Building IPV6 Based Tunneling Mechanisms for VoIP Security," 13<sup>th</sup> [International Conference on Communication and Signal Processing & Information Technology \(CSP\)](http://www.ssd-conf.org/ssd16/), Leipzig, Germany, SSD 2016 pp. 171-176, March 21-24 (<http://www.ssd-conf.org/ssd16/>).
- 4) A.J. Ghazali, W. Al-Nuaimy, A. Al-Ataby, M. Al-Tae, "Building IPSec Netflow Monitoring System Using Open Source and IPv6 Tunnelling Mechanisms" ready for 2019 International Conference on Internet of Things and Intelligence System (IoTaIS 2019) Bali, Indonesia, November 5-7, 2019.
- 5) A.J. Ghazali, W. Al-Nuaimy, A. Al-Ataby, M. Al-Tae, "Enhanced Blowfish Algorithm for Secured VoIP Security," ready for submission to Mobile Networks and Applications (MONET) Journal



# CHAPTER 2

## BACKGROUND

This chapter introduces and provides a general overview of VoIP protocols, VoIP traffic metrics, as well as service and voice qualities in the VoIP system. Meanwhile, Section 2.2 presents issues related to VoIP traffic metrics, followed by Section 2.3 focused on VoIP security, along with several possible solutions, whereas Section 2.4 depicts an insight of utilised audio codecs. Next, Section 2.5 discusses some methods of assessing voice quality, while Section 2.6 offers an extensive overview of prior studies carried out by other researchers. Lastly, this chapter ends with a conclusion in Section 2.7.

### 2.1 VOICE OVER INTERNET PROTOCOL (VoIP)

The VoIP, which is also known as Internet telephony or Internet Protocol (IP) telephony, is a concept that refers to voice transmission over Packet-Switched Networks (PSNs), such as the Internet. On the other hand, the typical telephony is carried over Circuit-Switched Networks (CSNs), for instance, the Public Switched Telephone Network (PSTN).

The VoIP incorporates a VoIP-enabled device or a VoIP computer program that runs on a Personal Computer (PC) or a mobile phone. The signal of the acoustic voice is captured by a microphone and transferred to a voice codec, where at the sender's side, has the role of sampling the raw analogue audio signal and transforming it into a digital stream of data, which is next sent to be encapsulated as payload into IP packets. Meanwhile, at the receiver's side, the function of the codec is reversed; in which the payload embedded into the IP packets is extracted and combined into a digital stream of data that is converted into analogue signal and later, transferred to a speaker.

In fact, VoIP is among the oldest applications designed for PSNs. The first mention of transporting voice over PSNs was performed via Network Voice Protocol (NVP) in 1973 by the Advanced Research Projects Agency (ARPA) [21]. Interestingly, the idea of VoIP preceded the protocols that have made VoIP possible at this present time, such as User Datagram Protocol (UDP) (1980) [22], IPv4 (1981) [23], H.323 (1996) [24],

Real-Time Transport Protocol (RTP) (1996) [25] and Session Initiation Protocol (SIP) (1999) [26],

From the stance of a service provider, VoIP evolution has involved a minimum of three stages:

- I. Business solutions: initially, VoIP functioned as a solution for companies to decrease maintenance cost for networking infrastructure by eliminating PSTN-like internal network only by using data infrastructure for both data and voice communications. In fact, the connection with PSTN has been carried out by VoIP Private Branch Exchange (PBX). As such, VoIP is confined to the boundaries of the company's physical network.
- II. Closed service domain: Skype [27][28] is the pioneer that offers VoIP services over the Internet, and chooses to apply a distributed system based on peer-to-peer protocol. Moreover, Skype offers free Skype-to-Skype calls for the large public. However, in order to place a call with a phone number in the PSTN, a fee is applied. With that, users are not limited by the boundaries of a network, but by the domain of the service provider.
- III. Federated VoIP: typically, an Internet domain provides access to webpages or hosts e-mail addresses. Meanwhile, federated VoIP adds to its functionalities, for example, VoIP. E-mail alike, federated VoIP demands a dedicated server that is addressable on that particular domain. In addition, the protocols used by the federated VoIP are also employed to offer communication services, such as video chat, conferencing, text chat, and shared desktop. While the first two mentioned solutions are somehow limited by either the network or the domain, this federated solution addresses limitations by enabling a plethora of inter-domain communication services.

### **2.1.1 VOIP PROTOCOLS**

International Organization for Standardization (ISO), Institute of Electrical and Electronics Engineers (IEEE) and Internet Engineering Task Force (IETF) assigned most of VoIP standard protocols for Internet communications and International Telecommunication Union- Telecommunication Standardization Sector (ITU-T) handles telecommunications

protocols and standard format for PSTN. Voice transportation via PSNs involves technologies that range from signal processing to time-sensitive packet transport protocols. Several key aspects and protocols of such technologies applied in this particular study are elaborated in the following:

### **A) VoIP Transport Protocols**

The following depicts how the captured voice samples are combined into voice frames and transported over PSNs.

Transmission Control Protocol (TCP) [29] appears to be the most widespread transport protocol used in PSNs. In fact, one of the primary features of TCP is the guaranteed delivery by using receipt acknowledgement reports sent by the destination party of a transmission, and later, back to the sender. If a packet is lost at transmission, it is re-transmitted, thus guaranteeing data integrity. Meanwhile, another essential feature of TCP is the algorithm employed to hinder network congestion by dynamically adjusting the transmission throughput. Such TCP features can eventually lead to relatively large transmission delays, which is undesired for real-time applications, such as VoIP.

Other than that, UDP [30] is another widespread transport protocol employed in applications, where timely delivery of data is made priority, more than integrity. Hence, lost packets are not re-transmitted and the destination of these packets does not send any receipt acknowledgement. Commonly, VoIP and Internet Protocol Television (IPTV) fall in the category of applications that apply UDP as the transport protocol.

Meanwhile, Real-Time Transport Protocol (RTP) [31] refers to the protocol used to uniquely identify a media stream. The RTP headers are attached to media packets that contain sequencing information so as to help the destination application rebuild the data stream.

Next, Real-Time Control Transport Protocol (RTCP) [31] denotes the sister protocol of RTP. This RTCP gives statistics and control data about the associated RTP flow. Its statistics incorporates several aspects, such as, packet count, lost packet count, Round Trip Time (RTT), and jitter. These data can be used by real-time applications for the media stream parameters to adapt to the network conditions. Such adaptation may include audio codec change or content transmission rate.

This E-Model employed the RTP headers extracted from VoIP packets in MOS calculation. A complete description of the MOS calculation process is given in Section 2.4.

## **B) VoIP QoS Protocols**

The delivery of demanding applications, for instance, VoIP and video conferencing displayed in Table 3, has to satisfy expectations. Hence, the primary objective of QoS protocols is to ascertain that the specified traffic metrics are maintained within the acceptable boundaries [32].

Besides, the IETF has been mainly involved in developing protocols to provide guaranteed QoS levels, as given in the following:

- Resource ReSerVation Protocol (RSVP) [33][34] reflects a protocol applied to carry the reservation request initiated by an application to guarantee resources. The request is carried along the path within the network and each node attempts to meet the request specification. In fact, RSVP has been employed by some QoS control architectures, for instance, Integrated Service (IntServ), Differentiated Services (DiffServ), and Multiprotocol Label Switching (MPLS).
- IntServ [35] refers to an IETF standard that is designed to offer a fine-grained and flow-based QoS traffic control within a network. Besides, IntServ is defined as a set of functions used by network nodes to manage per-flow traffic in a coordinated manner.
- DiffServ [36] is another IETF protocol designed to offer QoS guarantees per-packet. With that, the Differentiated Services Code Point (DSCP) field of the IP header is applied to segregate the traffic into various categories. In fact, each node along the path prioritizes packets that belong to certain classes over others so as to minimize the delay for time- sensitive applications, such as that in VoIP.
- MPLS [37][38] is an IETF standard that mimic the behaviour of CSNs in PSNs. MPLS works by attaching labels to packets interpreted by routers as QoS requirements, which can be translated into opting a shorter routing path for time-sensitive applications.

However, the need to guarantee some QoS may not affect a well-organised network scheme within LAN. The high speed connection of an Ethernet interface in PC makes voice communication just as effective as it is with analogue phones. However, in order to guarantee a similar level of QoS over the Internet, allocation of resources that may not belong to the end users may be required; inducing additional protocols to address the issue. With that, two approaches are available at this stage.

The initial option offers QoS with protocols that work within the IP layer with two available alternatives: Integrated Services (IntServ) [35] and Differentiated Services (DiffServ) [36]. IntServ is a model that guarantees the QoS between the end nodes. Hence, in order to cope with this task, every single hop in the network must satisfy the demands of the session initiation host, or else, the communication would not start. IntServ was originally developed by Cisco System and its corresponding signalling protocol is Resource Reservation Protocol (RSVP) [33]. This protocol is a signalling system that reserves its resources for both multicast and unicast communications, along with a request-response method. As for the DiffServ model, QoS is defined by classifying the traffic and adopting the varied priorities. Hence, it is more scalable than IntServ as each router decides the priority linked to the data. However, in real case scenarios, DiffServ may not be as effective as it portrays. If a company agrees to use DiffServ with an ISP, it cannot be guaranteed that all data would indeed pass these ISP routers. Thus, crossing varying ISPs cannot guarantee that the priorities have been accomplished from the source to the destination, thus compromising the desirable QoS.

Next, the second option to provide QoS is within the second layer found in the OSI model, with three available options: Frame Relay (FR) [39], Asynchronous Transfer Mode (ATM) [40], and Multi-protocol Label Switching (MPLS) [41]. The FR and MPLS technologies are based on creating virtual circuits on the network and guaranteeing a minimum bandwidth. Meanwhile, the ATM differs from the rest because it offers a different way to organize the network traffic, depending on the requirements. If a parallelism is demanded, IntServ is equivalent to FR and MPLS at a higher layer, while DiffServ is an ATM/MPLS service above the IP layers. Although these solutions are accessible, they are often costly or simply not feasible. Subsequently, the VoIP traffic is often routed on the Best Effort basis without weighing in network resource allocation. Thus, constraints of PSNs must be considered to determine the quality of performance

exerted by VoIP communications, as depicted in the following subsection with both subjective and objective methods to assess VoIP calls.

.

### **C) VoIP Signaling Protocols**

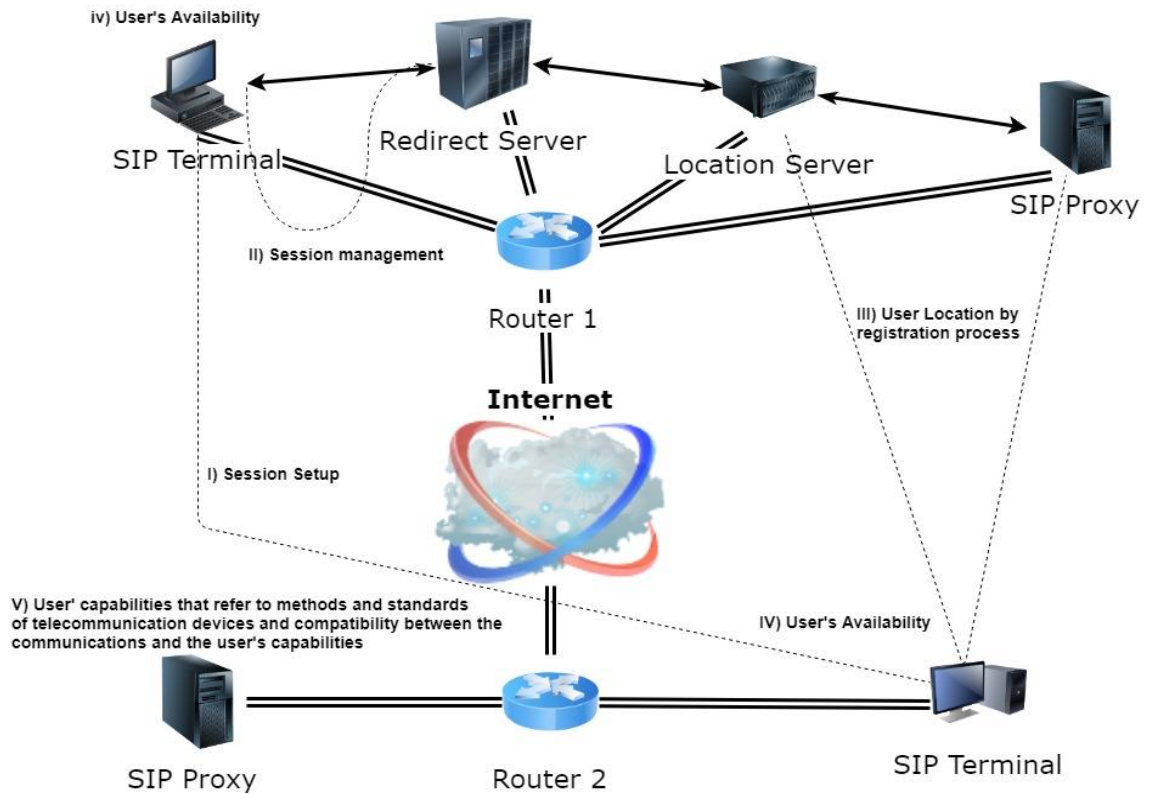
VoIP signalling protocol is a major part in the VoIP communication as it enables VoIP system to make the connection and call sessions between endpoint. As part of the VoIP infrastructures, signalling protocol enables components in the VoIP system to work together and to exchange information between network systems by providing the location of the endpoint before a session can be established, monitoring and releasing the connections, and controlling the system operations.

In recent years, H.323 and Session Initiation Protocol (SIP) are known to be the most commonly used protocols in the VoIP systems [42]. Most research has focused on H.323 and SIP due to its free and open source implementations. Nonetheless, there are also other related signalling protocols such as Inter-Asterisk exchange protocol (IAX), MegaCo/H.238 and Media Gateway Control Protocol (MGCP) which is also used in the VoIP signalling protocol.

SIP, one of the protocols standardised by IETF, is designed to support the bidirectional communication sessions, including VoIP setup. It is a text-based peer to peer protocol similar in some ways to the Hypertext Transfer Protocol (HTTP)[43]. It serves five functions to the VoIP system [44] :

- i. Session setup, which referred to the session parameter for both ends of the communications.
- ii. Session management that can provide the flexible and unnoticeable impact to users to manage and change a session.
- iii. The location of users, it can determine user locations by a registration process.
- iv. User's availability is a method to determine whether or not a user would be willing to answer a request to communicate.
- v. User's capabilities refer to the various methods and standards of telecommunication devices and the compatibility between the communications and the user's capabilities.

Figure 2 shows the architecture and protocol stack of SIP.



**Figure 2: SIP architecture overview**

In SIP architecture, there are several entities: SIP terminal, proxy server, redirect server and location server. The redirect server's function is to receive a request from the SIP terminal as well as to inform the caller about the next hop server while the proxy server acts as both client and server through receiving a request or making requests on behalf of other clients. Location server or registrar server contains the locations database as well as user preferences.

There are several types of SIP messages for call establishment: INVITE, ACK, BYE, CANCEL, REGISTER and OPTIONS as shown in Table 1.

**Table 1: SIP Messages [45]**

SIP Request Type	Function
INVITE	Session establishment request
ACK	Acknowledgement of receiving INVITE message
OPTIONS	Capabilities of the server is being queried
BYE	Client asks server to terminate the call

## 2.2 VOIP TRAFFIC METRICS

The basic traffic metrics used by many Quality of Service (QoS) protocols are as follows:

- Bandwidth refers to the amount of traffic (measured in bits) that traverses a point of the network within a given period of time (measured in seconds), hence measured in bits per second (bps).
- One-way-delay is the measure of end-to-end packet delay [46], which is comprised of a number of delay factors that occur at each node involved in the packet delivered.
- Processing delay refers to the time required by the network equipment to process the headers of the packet, as well as to determine the next required packet action, for instance, the next hop. Besides, another delay due to processing occurs upon packet reception at the physical layer when the received packet is examined to detect transmission errors.
- Transmission delay is the time needed for a sending source to push all the bits of a packet onto the transmission medium, which is strongly linked to the data-rate.
- Propagation delay reflects the time spent by the bits in a packet to traverse to the propagation medium, such as copper wires, optical fibre or ether.
- Queueing delay is the time spent by a packet at the outgoing queue of a node waiting for the availability of a transmission medium.
- Jitter refers to the variation of a packet delay between a particular set of packets that belongs to the same packet stream [47], in which the variation can take place due to changes that affect the network, for example, alternative routing or congestion.
- Packet loss reflects the proportion of packets sent by the source, but not received at the destination and it is typically measured as a percentage of the number of packets lost over the total number of packets sent [48]. One main cause of packet loss is queue overflow that occurs when the outgoing queue is full at the time of packet arrival. Packet loss can also happen in wireless environments, where the propagation medium may suffer from poor transmission conditions primarily due to channel interference.



Latency is defined as the delay of voice data when crossing an IP network excluding any processing or queuing. This parameter refers to a measure of the propagation of data delay through wires.

Queuing and processing refer to the delay related to the processing in the router/switches that read the IP destination to route the packets to their next hop. At times, as the propagation delay can be assumed as zero, the concept latency is employed to express the delay of the propagation, inclusive of delay due to queuing and processing.

Delay happens when packets of voice data take more time than expected to reach their destination. The packets might arrive late or probably not arrive at all, which is considered as packet loss. This delay, also called latency, causes some disruption in voice quality. High QoS should be considered for voice data as this is relatively less tolerant toward voice delay, latency and packet loss to ensure it does not make a huge impact so the voice data transmission and conversation can be acceptable.

Furthermore, by assuming latency as the summation of both delays, it is vital to distinguish the two latencies in voice communication; one per voice direction. Meanwhile, round trip latency is the summation of two-way latency. Table 2 displays the ITU Recommendation G.114 [49], which indicates that the range of round trip latency is acceptable when it is below 150 ms for most applications. Relatively, the range between 150 and 400 ms is acceptable for administrators who are aware of the connectivity being used. Nonetheless, 400 ms is unacceptable. Likewise, varied latency ranges can cause echo and talker overlap. If the round trip latency exceeds 50 ms, the echo from one of the speakers may be heard in the communication, suggesting the installation of an echo canceller in the vocoder. Meanwhile, the talker overlap is considered when the one-way delay exceeds 250 ms.

**Table 2: ITU recommended values for VoIP quality**

Delay	< 150ms	> 150ms < 300ms	> 300ms
Jitter	< 20ms	> 20ms < 50ms	> 50ms
Packet Loss	< 1%	1% < 5%	> 5%
<b>Performance</b>	<b>Excellent</b>	<b>Good</b>	<b>Poor</b>

In fact, packet loss occurs mainly due to two reasons: i) a glitch in the physical layer due to numerous corrupted bits, hence forcing the receiver to reject message, and ii) the finite memory space at the routers that eventually cannot allocate more space for incoming packets, thus failing to deliver information to its destination.

On the other hand, jitter refers to the variation of delay in packets that reach their destinations. Such variation in inter-packets arrival rate makes a conversation unbearable. This issue is commonly solved by introducing a buffering system to decrease its undesired effect.

In fact, all these traffic metrics-bandwidth, one-way-delay, jitter, and packet loss have been defined by the Internet Engineering Task Force (IETF), which refers to an organization that develops and promotes the Internet standards.

Several viable applications enabled by PSN and some related requirements for the metrics mentioned above are presented in Table 3. Hence, one can observe that VoIP imposes the strictest requirements in terms of one-way-delay, jitter, and packet loss, but not for the low VoIP bandwidth requirements, as depicted in Table 3. Meanwhile, a streaming traffic type implies nil rate control, regardless of the network state, while an elastic traffic type adapts its rate to match the network condition. In precise, the two traffic types are characterized by UDP and respectively, TCP type of traffic.

**Table 3: QoS requirements per application (Low, Medium, High) [50]**

Application	Bandwidth	Delay	Jitter	Loss	Type
VoIP	Low	High	High	High	Streaming
Video Conference	High	High	High	Med	Streaming
Streaming VoD	High	Med	Med	Med	Streaming
Streaming Audio	Low	Med	Med	Med	Streaming
E-Mail	Low	Low	Low	High	Elastic
File Transfer	Med	Low	Low	High	Elastic

## 2.3 VOIP SECURITY ISSUES

With the popularity of VoIP networks deployments [51], issues related to security and privacy aspects have emerged significant [52]. As such, three main components need to be addressed which are; authentication, privacy, and integrity [53]. The authentication process verifies if its user is true. Meanwhile, the process of integrity validates and checks if the data and contents are true while being transposed between the sender and the receiver. Lastly, the privacy process ensures that the data are protected by using encryption and decryption methods from being alternate and intercept [54] [55].

### 2.3.1 VOIP SECURITY THREATS

The operations of circuit-switches, for example, traditional telephone system, can unable cases of eavesdrop and other security issues. Figure 3 illustrates the taxonomy of probable security threats related to voice traffic on converged networks, such as packet-switches communication network.

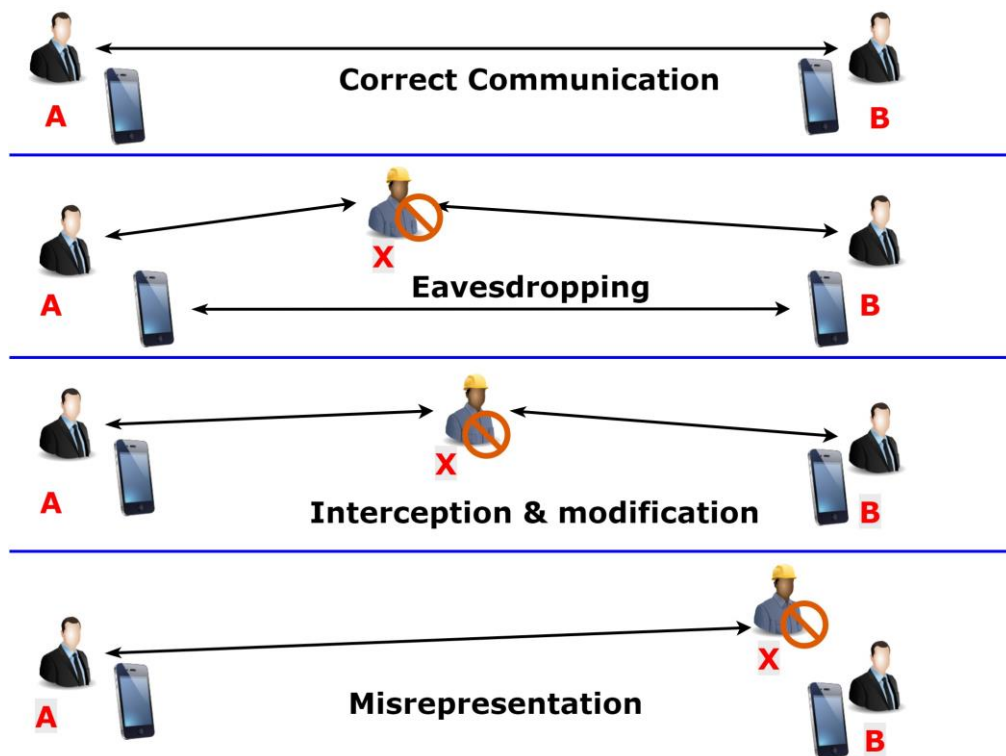


Figure 3: VoIP security threats [181]

Hence, the most frequent VoIP deployments (end-users, applications, and services) against security threats and vulnerabilities are summarized in the following:

- 1) Social threats: Misrepresentation of authority, identity, contents, and right is mostly aimed against humans for theft of service, unwanted contact, phishing, and spam [56].
- 2) Eavesdropping, modification threats, and interception: An intruder who eavesdrops [57] can capture the entire data stream and/or signalling between the participants and end users of VoIP in an unlawful manner and without authorization [53]. The data can be read and modified before sending to the VoIP network unless they have been encrypted [58].
- 3) Denial of services (DoS) threats: The DoS is contemplated as interruption of a service [59]. This potential attack denies users access to VoIP service that can exploit flaws in a call setup or in the service implementation. Besides, it may involve direct attack to the Domain Name Server (DNS) and the Session Initiate Protocol (SIP) server [43][60].
- 4) Service abuse threats: This threat usually comes from an employee/customer of an ISP/third party that uses VoIP services [61][62]. For instance, when the traffic is artificially increased, the charges for billing can be maximized or vice versa [42]. Other than that, one's personal details could be compromised and capitalized on by various forms of account thefts and stolen identities [63].
- 5) Physical access threats: Unauthorized and inappropriate physical access to VoIP devices or equipment is considered as physical access threats [64]. The intruder may gain unauthorized access to any physical layer of the network.

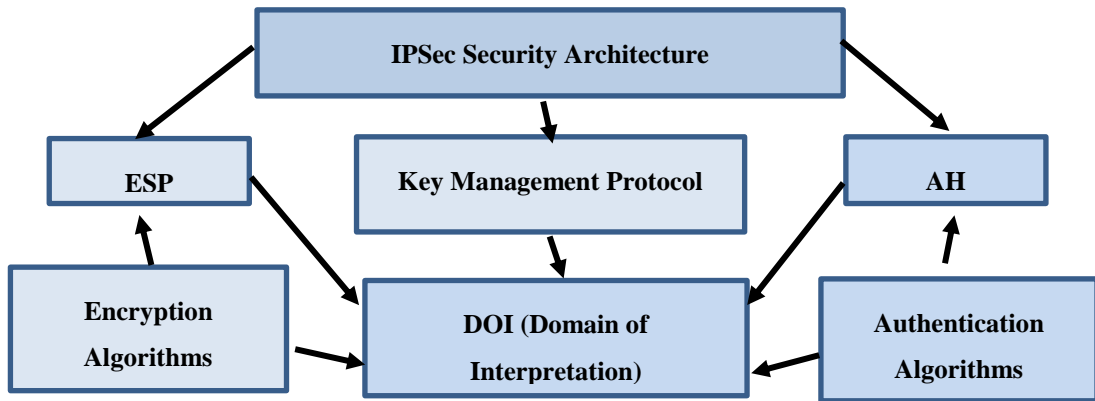
- 6) Interruption of services threats: A case of inaccessibility of VoIP services derives from non-intentional problems [65], such as loss of power due to weather and by nature [66]. The performance is compromised because of resource limitation and degraded call quality.

### **2.3.2 INTERNET PROTOCOL SECURITY (IPSEC)**

The IPsec refers to a protocol suite that secures Internet Protocol (IP) communications by encrypting and authenticating each IP packet of a data stream [67]. This IPsec includes protocols that establish authentication between agents at the beginning of the session and negotiations of cryptographic keys to be applied during the session [68]. The process of cryptographic keys negotiations allows two parties to remotely establish a shared secret over an insecure channel. Each key negotiation begins by agreement of both parties on a shared policy that states which security parameters will be used to protect key negotiations. Moreover, IPsec can be used to protect the flow of data between a pair of hosts (e.g. computer users or servers), between a pair of security gateways (e.g. routers or firewalls), or between a security gateway and a host [64]. As part of open standards framework [69], the IPsec consists of some related protocols to perform various purposes [70][71]:

- Internet key exchanges (IKE and IKEv2) are used to configure security association (SA), where the SA handles protocols and algorithms negotiation, besides generating both authentication and encryption keys to be used by IPsec [72][73].
- Authentication Header (AH) offers data origin authentication and connectionless integrity for IP datagram and provides replay attacks protection [74].
- Encapsulating security payload (ESP) gives data origin authentication, confidentiality, anti-replay service (a form of partial sequence integrity), connectionless integrity, and confidentiality for limited traffic flow [75].

Figure 4 illustrates IPSec architecture:



**Figure 4: IPSec interrelationships [80]**

The implementation of IPSec is programmed by using a security policy as a rule that processes varying datagram when received by a device. By using the security strategies, especially after bypassing AH and ESP, every particular packet is decided by the IPSec if it should be executed. The security policy in a device is stored in the security policy database (SPD) that provides general guidelines for security implementation and links to specific tasks [76]. One fundamental aspect of IPSec is the security association (SA), which is a relationship of a particular kind of secure connection between one device and another. Being a security mechanism that is used to secure communication between two devices, it serves as a “contract” that permits the transmission to be established. Besides, the security association database (SAD) contains all the information regarding the device’s inbound and outbound traffic [77].

Furthermore, the tool that forms SAs is identified as Internet key exchange (IKE), which reflects a safe and automatic way to deal with the details for the SA to be set up. Besides, IKE is a hybrid procedure that uses part Oakley and part SKEME linked to ISAKMP so as to attain an authorized keying material to be applied with ISAKMP, and for other security associations, such as AH and ESP, for IETF IPsec DOI. Meanwhile, the IKE daemon uses UDP port 500. The procedure comprises of two phases: the first phase is for authentication, while the second phase is for the key exchange [78]. In addition, ISAKMP offers a framework for verification and key exchange but does not define them. In fact, it is created to be the main exchange independent that supports a variety of key exchanges. Besides, Oakley and SKEME function as tools that form authenticated key exchange. The process consists of information payloads that carry

payloads construction, as well as the order in the process and usage. Meanwhile, IKE is a series of key exchanges known as "modes" [72]. In the authentication phase, two modes are accessible, which are main and aggressive modes. Although these modes are faster, they do not provide identity security for the negotiating parties. Besides, this mode may be defenceless against DoS attacks. Authentication phase is the medium where the two ISAKMP peers develop a protected and real channel for communication, which is known as Security Association (ISAKMP SA) [79]. Next, a quick mode that establishes the IPsec SA [A] can be found in the key exchange phase.

The negotiation of security associations, on behalf of services such as IPsec or any other service that needs key material, and/or parameter negotiation, is embedded in key exchange phase, which consists of IKE implementations support associated to the following attribute values [80]:

- Data encryption standard (DES) in cipher block chaining (CBC) mode with weak and semi-weak key-checks.
- Message digest 5 (MD5) and secure hash algorithm (SHA).
- Authentication via pre-shared keys.
- Modular Exponential (MODP) over default Diffie-Hellman (DH) group number one.

A format protocol offers data validation, integrity, and non-repudiation, but it does not offer data privacy, which is known as AH [74]. AH can add security to communication stream, encrypt non-volatile fields of the IP header, and create a message digest value at the initiation of the packet. In fact, the creation of message digest value and the encrypted AH header can be inserted between data portion of the packet and the original IP header by using data portion of the packet, as well as one-way hash of the IP header. Figure 5 illustrates the construction and the interpretation of an AH packet.

0 – 7 bit	8 – 15 bit	16 – 23 bit	24 – 31 bit
Next header	Payload length	RESERVED	
Security parameters index (SPI)			
Sequence number			
Authentication data (variable)			

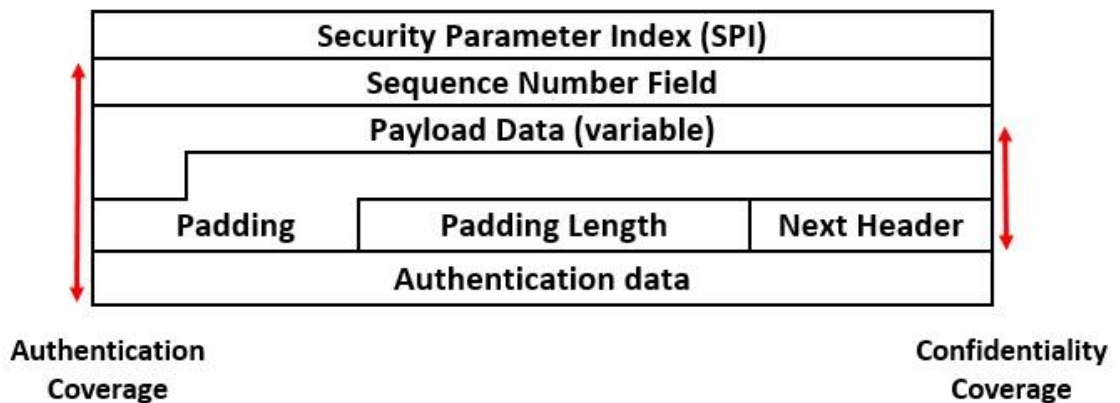
**Figure 5: IPSec AH packet diagram [74]**

The aspects of AH field [74] are given in the following:

- Next header - identifies the type of the next payload after the Authentication Header.
- Payload length - specifies the length of AH in 32-bit words (4-byte units), minus "2".
- Reserved – 16-bit field is reserved for future use, the sender must set “zero” as values and it should be ignored by the recipient.
- Security parameter index (SPI) - an arbitrary 32-bit value that, in combination with the destination IP address and security protocol, uniquely identifies the SA for this datagram.
- Sequence number - contains a monotonically increasing counter value, which is mandatory and is always present even if the receiver does not elect to enable the anti-replay service for a specific SA.
- Authentication data - a variable-length field containing an Integrity Check Value (ICV) computed over ESP packet minus authentication data.



ESP denotes a format protocol that runs data privacy via encryption [75]. Replay attack protection, on the other hand, is provided by ESP and over the use of a hash, it can give data origin verification and integrity. ESP can add safety to the communication stream via encryption of data payload in transport mode or via encryption and encapsulation of the entire IP packet when in tunnel mode. For instance, DES, 3DES, and AES are symmetric encryption algorithms, while MD5 HMAC and SHA1 HMAC are for data authentication and integrity that can be used and supported by ESP. Compared to AH, ESP does not protect IP packet header. Moreover, ESP runs directly on top of IP by using IP protocol number 50. Figure 6 illustrates how an ESP packet is constructed and interpreted:



**Figure 6: IPsec ESP packet diagram [75]**

The aspects of ESP Field [75] are listed in the following:

- Security Association identifier - a pseudo-random value that identifies the security association for this datagram.
- Sequence number - contains a monotonically-increasing counter value, which is mandatory and is always present even if the receiver does not elect to enable the anti-replay service for a specific SA.
- Payload data - a variable-length field containing data described by the Next Header field.
- Padding C - padding for encryption.
- Pad length - indicates the number of pad bytes that immediately precedes it.

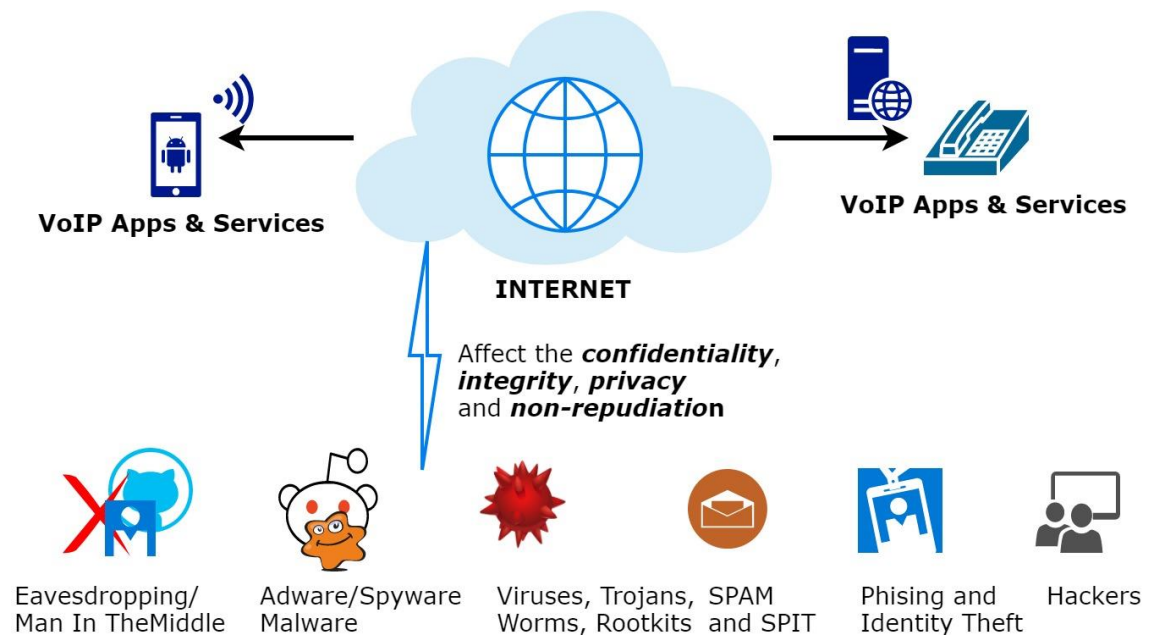
- Next header - identifies the type of data contained in the payload data field, e.g., an extension header in IPv6 or an upper layer protocol identifier.
- Authentication data - a variable-length field containing an Integrity check value (ICV) computed over the ESP packet minus the authentication data.

IPSec can be run either in tunnel mode or transport mode. Each mode has a particular use, and thus, care should be taken to ensure that a suitable mode is opted. The tunnel mode is most commonly used between gateways, or at an end-station to a gateway, whereby the gateway serves as a proxy for the hosts behind it [81]. Meanwhile, the transport mode is applied between end-stations or between an end-station and a gateway, if the gateway functions as a host, such as an encrypted Telnet session from a workstation to a router, where the router is the actual destination [82].

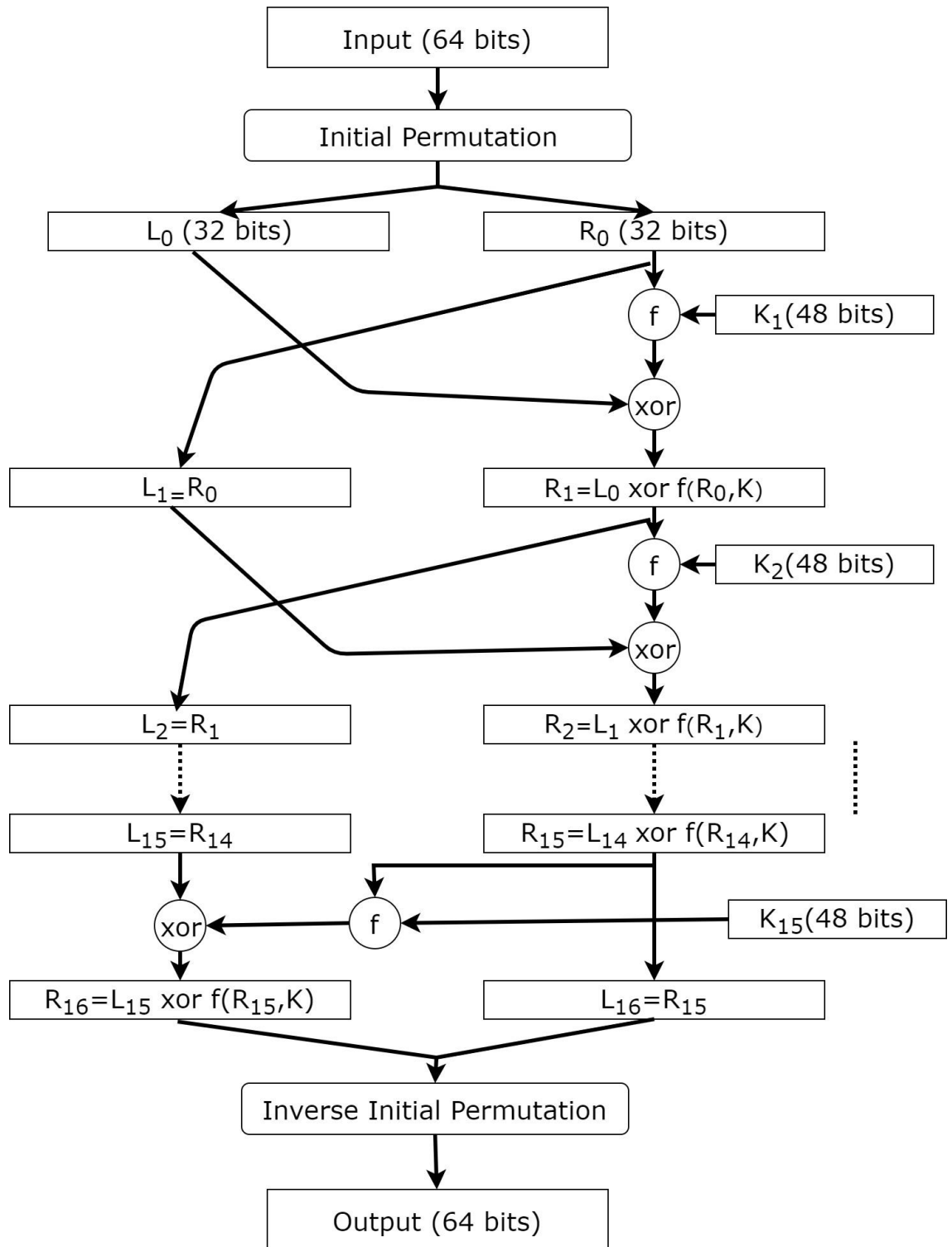
### **2.3.3 VOIP ENCRYPTION AND ALGORITHM**

Beyond doubt, as illustrated in Figure 7, security appears to be a concern when using the public Internet, especially in providing privacy, integrity, and security during data and voice communications. As for the VoIP network infrastructure, the VoIP traffic in an application requires security and as such, IPSec is a strategy that secures the VoIP network [83]. In fact, a study by [84] examined VoIP attacks and proposed security considerations, as well as methods to minimize security risk, based on some algorithms proposed in VoIP encryption [85] [86]. The most frequently used symmetric key encryption algorithms to secure VoIP services and communications are block cipher encryption, such as AES, 3DES, DES, and Blowfish [87]. Moreover, prior studies that investigated network employed the 3DES encryption method although it is still under evaluation [81]. The 3DES was developed in 1998 to substitute the DES [88] for it gives stronger encryption by using 2168 possible combinations via three round messages. As such, more security was discovered upon using 48 rounds in its computation and a key length of 168 bits, in comparison to the previous DES encryption algorithm [89]. Figure 8 illustrates a functional diagram that increases the level of security with 3DES encryption, which also adds to the average processing time. For instance, 3DES and AES encryption algorithms had been applied to evaluate the voice quality in wireless LAN [89] [90]. Similarly, the 3DES algorithm was implemented an IPSec experiment [91], whereby IPSec performance using various encryption algorithms, including 3DES, had been compared with Windows 7, wireless network access, and IPv4/IPv6 protocol. Other

than that, two popular encryption methods, DES and Blowfish, have been studied [92], which displayed that the Blowfish algorithm resulted in faster execution time (encrypt/decrypt) and proved to be more secure than the DES algorithm.



**Figure 7: VoIP security challenges and threats**



**Figure 8: 3DES encryption functional [182]**

### 2.3.4 BLOWFISH ENCRYPTION ALGORITHM

Developed by Bruce Schneier in December 1993, Blowfish algorithm appears to be an alternative to the existing encryption algorithms due to its powerful encryption that offers both suitability and security [93]. As this algorithm is unpatented and does not require any license, it has grown popular among the open source community [94]. Blowfish is a symmetric-key block cipher with an F-function design; the key length varies between 32 bits and 448 bits; and the block size is 64 bits, hence making it suitable for both domestic and exportable applications.

A 16-round Feistel cipher and large substitution boxes (S-boxes) are used. These S-boxes, generally, depend on the key. These Feistel ciphers are symmetrical structures found in block ciphers. In fact, they are iterated ciphers with an internal function known as round function [95]. Horst Feistel initially described them during his work on the cipher Lucifer at IBM [96]. Lucifer refers to a predecessor to the Data Encryption Standard (DES). One advantage of the Feistel network is the similarity shared between encryption and decryption, thus making both circuitry and code smaller [92]. Meanwhile, other ciphers that use the Feistel network are IDEA, RC5, and Skipjack [97].

The implementation of the Blowfish cryptosystem is comprised of two parts: a subkey/S-Box generation phase, and an encryption phase [98]. Since its introduction by Bruce Schneier in 1993, some researchers have analyzed the cryptosystem for its security features and have attempted to crack the system. Some popular cryptanalysis techniques used to crack are substitutions/permutations linear, differential cryptanalysis, avalanche effect, and correlation coefficient [99].

Moreover, the Blowfish algorithm is one of the best four popular encryption algorithms (DES, 3DES, and AES) [100][101][102], which had been compared by encrypting input files with varied contents and sizes [103]. The findings showed that Blowfish emerged as the fastest algorithm (execution time for encryption and decryption), with memory required for implementation and throughput [104]. However, since performance appear to be the primary aspect examined in this research, inevitable trade-off issues were noted between performance and security, as highlighted by many researchers [103][105][106].

The action of Blowfish algorithm is illustrated in Figure 9.

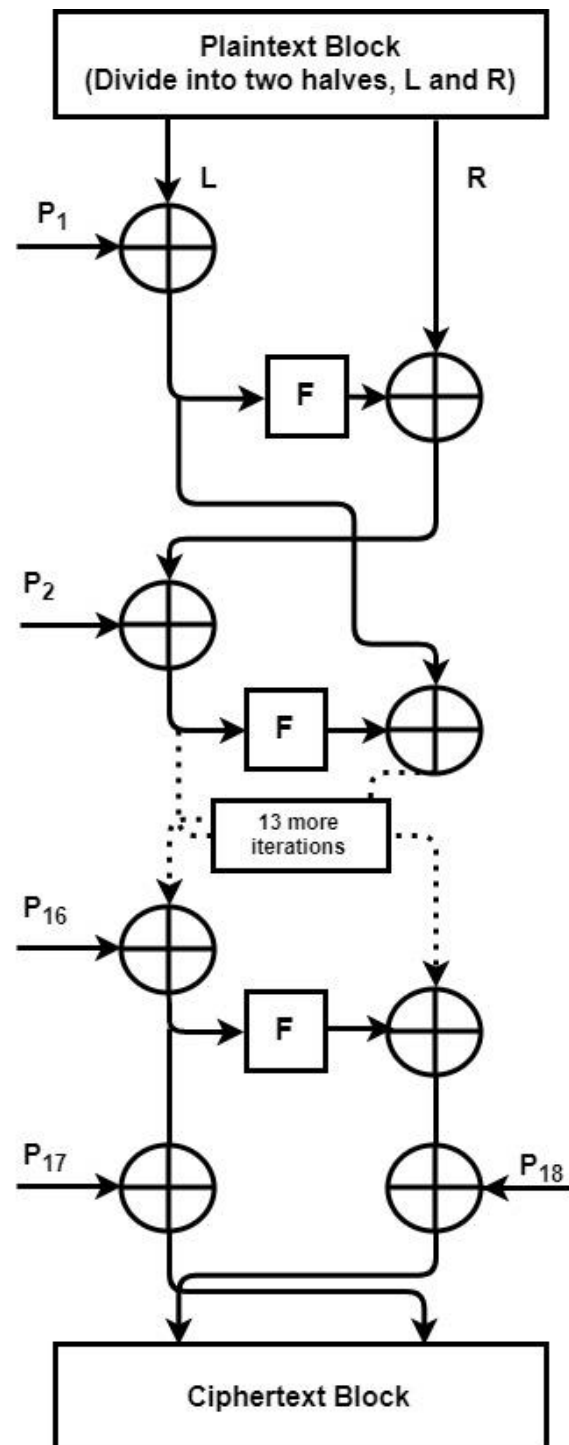
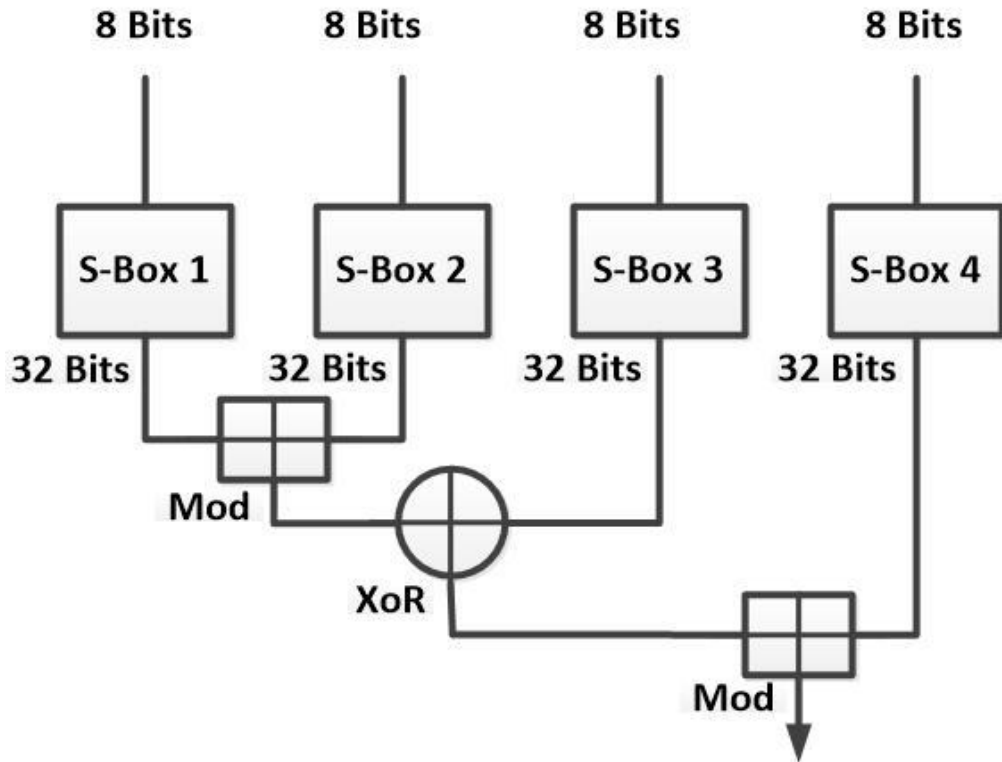


Figure 9: Blowfish Feistel structure [109]

Each line represents 32 bits. The algorithm uses an 18-entry P-array and four 256-entry S-boxes, which contain subkeys. The S-boxes take 8-bit inputs and give 32-bit outputs. In each round, only one entry of P-array is used. After the last round, each half of the data block is XORed with one of the two remaining P-entries [92]. Besides, Figure 10 portrays the function of the Blowfish Feistel.



**Figure 10: Blowfish Feistel function [92]**

The 32-bit input is split into four 8-bit quarters and each quarter serves as input to the S-boxes. Meanwhile, the outputs are added into modulo  $2^{32}$  and XORed to produce 32-bit final outputs.

Initially, the algorithm starts with key initialization step. A key schedule refers to an algorithm that calculates the subkey for each round with the key [107]. Furthermore, the key schedule of the Blowfish begins with initialization of P-array and S-boxes with values acquired from the pi hexadecimal digits that contain no obvious pattern. Next, the key is byte-by-byte XORed with all the P-entries in sequence. Moreover, an all-zero 64-bit block is encrypted with the algorithm. P1 and P2 are replaced with the resultant cipher text, which is then encrypted again using new subkeys, whereas P3 and P4 are replaced with new cipher text. This process continues substituting the entire P-array and S-box

entries. In total, it takes 521 encryptions to calculate all the subkeys. During the subkey generation process, the subkeys change slightly with every pair of generated subkeys in order to protect against any attack that may exploit the fixed and known subkeys [108].

The data encryption is considered done when all the subkeys are calculated. The action of Blowfish Algorithm [108] is illustrated in Figure 11 below.

Divide X into two 32-bit halves: XL and XR

For i = 1 to 16;

$XL = XL \oplus P_i$

$XR = F(XL) \oplus XR$

Swap XL and XR

Swap XL and XR (Undo the last swap)

$XR = XR \oplus P_{17}$

$XL = XL \oplus P_{18}$

Concatenate XL and XR

Divide XL into four eight-bit quarters: a, b, c and d

$F(XL) = ((S1[a] + S2[b]) \oplus S3[c]) + S4[d]$

**Figure 11: Operation of Blowfish Algorithm**

First, the 64-bit block input is divided into two 32-bit halves. Several exclusive-OR and Feistel function operations are performed at each round with one P-array entry per round for 16 rounds. After 16 rounds, each half of the data block is XORed with two remaining unused P entries. The 32-bit half data blocks are then concatenated to gain the cipher text, except if P1, P2.... P18 are used in the reverse order, where the decryption is exactly similar to the encryption.

Besides, many implementations support key-sizes up to 576 bits because, during initialization, the key bytes are XORed with all 576 bits of the P-array. However, the key size is limited to 448 bits so as to ascertain that every bit of every subkey is dependent



on every bit of the key, as the last four values of the P-array do not have any impact upon every bit of the cipher text [109].

## **2.4 VoIP AUDIO CODECS**

Human speech represents changes that occur in the local air pressure that surrounds the atmosphere. In all voice transmissions, these changes are captured by a microphone and transformed into analogue electrical signal [110]. Any analogue signal has, in theory, infinite bandwidth, however, in practice; bandwidth is limited by using filters. In fact, most information in human speech can be found within the range of 100Hz to 4000Hz band [111]. Signal sampling, hence, is required to transform an analogue signal into one that is digital. In the signal theory, the Nyquist rate [112] specified that the minimum sampling rate required to avoid aliasing should be equal or bigger than twice the bandwidth of a band-limited signal. Hence, for human speech, the common sampling rate is 8000Hz.

As for VoIP, the conversion between analogue and digital domains is performed by COders/DECoders (Codecs), as shown in Figure 12 [113]. A large variety of codecs that exist with the main variances between them are compression level, bandwidth efficiency, and speech quality. These features demand additional processing time at either the coder or the decoder side so that more complex codecs can increase the mouth-to-ear delay of the audio signal [114]. Nevertheless, this amount of time delay is insignificant for human perception. Thus, the extra time needed for processing is called look-ahead delay.

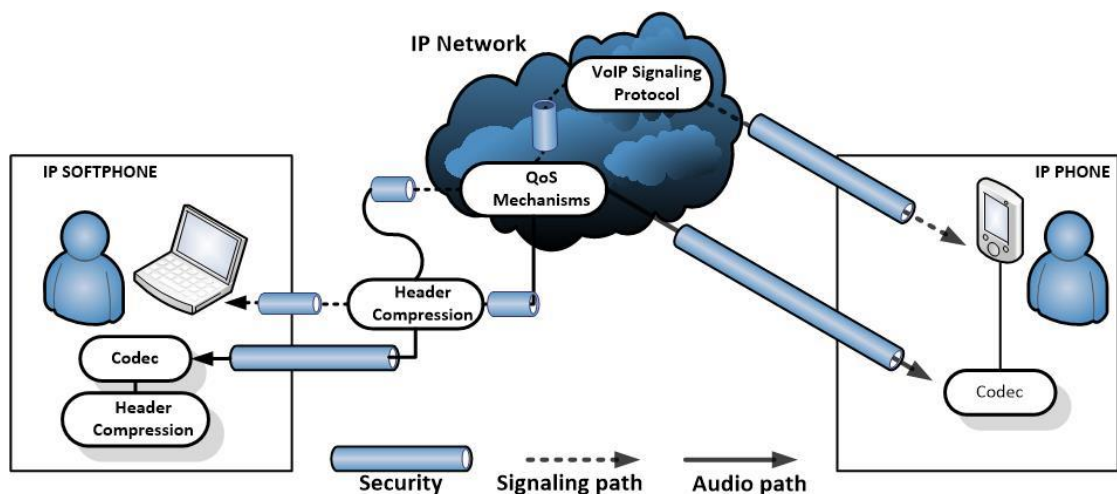
Codecs, which serve as algorithms, are used in VoIP systems to enable voice signals to be carried over IP networks. Codecs variations differ in complexity, voice quality, and bandwidth consumption, in which the categories of codecs can be in a narrow band, wideband, or multimode. Typically, more bandwidth is allocated to codecs to gain better voice quality [115].

Furthermore, the concepts of frame and look-ahead delay are present in complex codecs that apply compression. Additionally, Table 4 shows the most known audio codecs employed in VoIP. In fact, many VoIP codecs sample the audio signal at 8000Hz, except for Adaptive Multi-Rate Wide-Band (AMR-WB) that samples at 16000Hz; while the AMR-WB is considered to be a high definition codec for its high sampling rate [116]. Table 4 also depicts that the bitrate of the codecs can range from 4.75kbps for Adaptive Multi-Rate Narrow-Band (AMR-NB), which is very small, to 64kbps for G.711. Hence,

a frame is composed of multiple samples that are transformed by a compression algorithm into a smaller number of bytes needed to encode the digital data. The frame size in Table 4 refers to the time duration of the audio signal included in one frame. Thus, G.711 and G.726 are codecs that do not use compression. Therefore, the frame concept does not really apply, and the corresponding frame size values reflect the amount of time captured in one sample period. As G.711 and G.726 use no compression, there is no look-ahead delay. The far-most right column displays the maximum achievable VoIP call quality that is graded with Mean Opinion Score (MOS) scale.

**Table 4: Comparison of VoIP codecs [117]**

Codec	Sampling (kHz)	Bitrate (kbps)	Frame size + look-ahead (ms)	MOS
AMR-NB	8	4.75 ... 12.2	20 + 5.0	4.14
AMR-WB	16	6.6 ... 23.85	20 + 5.0	4.30
G.711 (PCM)	8	64	0.125 + 0.0	4.30
G.723.1	8	5.3, 6.3	30 + 7.5	3.65
G.726	8	16 ... 40	0.125 + 0.0	3.85
G.729	8	8	10 + 5.0	3.92
GSM FR	8	13	20 + 2.5	3.50
GSM EFR	8	12.2	20 + 2.5	3.80
iLBC	8	15.2	20 + 5.0	4.14



**Figure 12: VoIP codecs overview**

In this research, three codecs had been applied and tested, which consisted of G.711, G.729, and AMR-NB, as described in the following:

1. G.711 Pulse Code Modulation (PCM) [118] is probably one of the most well-known codecs employed in many audio applications, and it is the main codec applied in PSTN. No compression is used, and its data rate is constant during audio transmission.
2. One of the most popular codecs that use compression is G.729 [119]. The bandwidth used by G.729 is smaller than the bandwidth used by G.711. However, due to inter-frame dependencies introduced by the encoding algorithm, G.729 and similar codecs are prone to packet loss.
3. The Adaptive Multi-Rate (AMR or AMR-NB or GSM-AMR) appears to be among the most widely used Global System for Mobile communication (GSM) audio codec and used in audio compression format optimised for speech coding. It has been adopted from 3rd Generation Partnership Project (3GPP) in the year 1998 and used for both GSM and circuit switches Universal Mobile Telecommunications System (UMTS)/ Wideband Code Division Multiple Access (WCDMA) voice calls. The AMR speech codec consists of a variety of multi-rate narrowband speech codecs that encode narrowband (200–3400 Hz) signals at variable bit rates that range from 4.75 to 12.2 kbit/s with toll quality speech starting at 7.4 kbit/s.

As voice codecs can enhance bandwidth efficiency by employing Voice Activity Detection (VAD) algorithms [120], VAD is an algorithm used by codec to determine if the microphone captures the actual speech or merely background noise. When a VAD algorithm detects background noise, the codec will output very small-sized packets to inform the other party to play background noise, thus reducing the usage of network bandwidth during silent periods.

## 2.5 VOIP VOICE QUALITY ASSESSMENT

An established call can be measured using voice quality assessment via objective and subjective methods [121], which is further elaborated in the next section, which includes call quality using Mean Opinion Score (MOS) and E-Model. In addition, Figure 13, as adapted from [122], presents the measurement of call quality between user #1 and user #2 with the relation of the network QoS, whereby good QoS within a network results in satisfying and good call quality [123].

Moreover, voice quality assessment is made up of both objective factors, including hardware, software, and network conditions, such as delay, jitter, and packet loss. Meanwhile, the subjective factor includes user expectation and conversation effort. As for the VoIP network, voice quality can be measured by using both measurements. However, subjective assessment has limitations, where the objective measurement has detailed quantitative measure in terms of network QoS, such as delay, jitter, packet loss, and MOS [124].

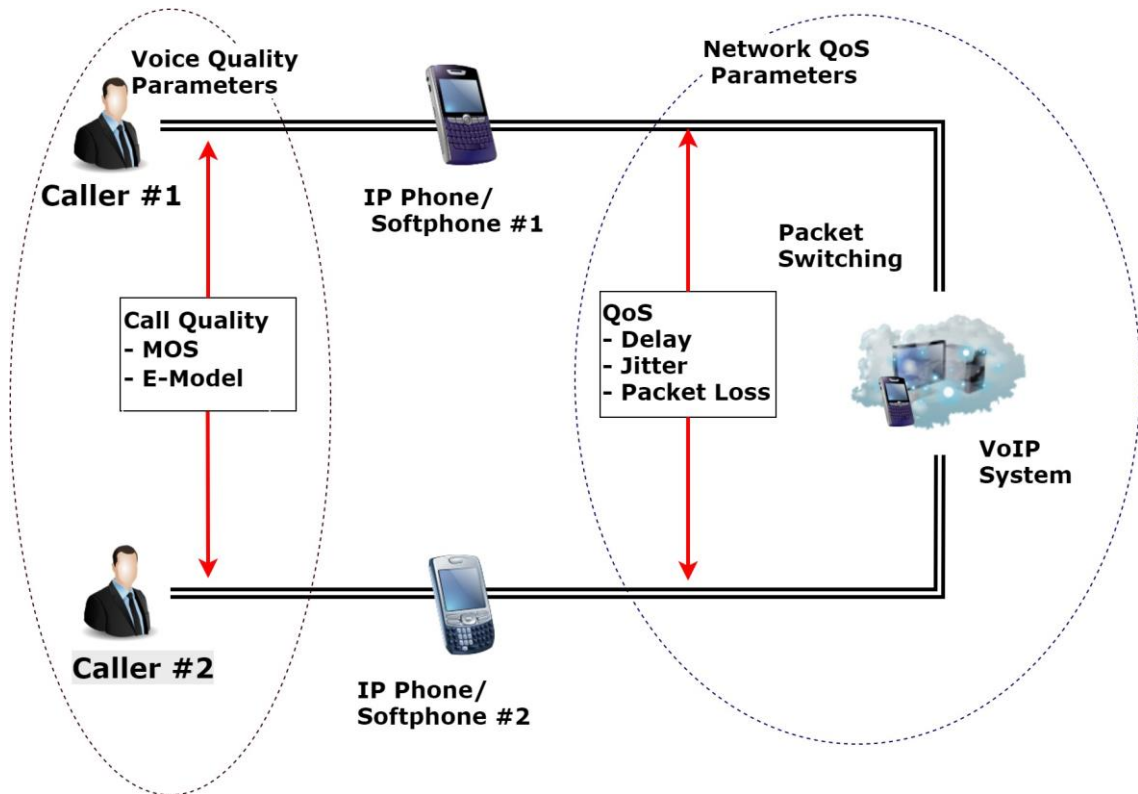


Figure 13: Call and voice quality assessment

### 2.5.1 MEAN OPINION SCORE (MOS)

The International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) report P.800 [125] introduced a scoring system to examine the quality of a speech transmitted via telephone lines. The score ranges from 1 to 5, as presented in Table 5. Traditionally, when human test subjects involved in grading the quality of a speech using this scale, the result is reflected in *Mean Opinion Score (MOS)*.

**Table 5: MOS scale for subjective assessment [126]**

Quality	MOS	Impairment
Excellent	5	Imperceptible
Good	4	Perceptible but not annoying
Fair	3	Slightly annoying
Poor	2	Annoying
Bad	1	Very annoying

The three varying methods available to determine the MOS of a voice call are given below:

- Subjective (S): human subjects are involved in grading the quality of speech;
- Objective (O): excludes humans from assessing the speech, but uses an objective model to predict speech quality based on the variances between the original and the received speech signal;
- Estimative (E): uses a model that considers communication transmission factors, while dismissing both original and received signals.

The accuracy of Subjective MOS scores is as high as human assessment by averaging the scores reported by all participants. The major limitation in this case, however, is the scarcity of human subjects. Besides, the process has high cost and

obviously cannot be performed in real time. Thus, both objective and estimative methods had been developed[127].

The most popular model for objective speech assessment is the Perceptual Evaluation of Speech Quality (PESQ) [128]. A reference audio sample is injected into a telephone network, and the output is recorded. Next, the PESQ methodology compares the reference and the recorded output, whereby the results of the comparison are directly linked to the quality of the transmission. Similar to the subjective assessment, the objective assessment also cannot be used in real-time. Moreover, these two methods do not capture the influence of network parameters, such as delay, jitter, and loss, on speech quality. As abovementioned, VoIP quality is correlated with the traffic metrics depicted above. Hence, a different method has to be used in order to assess the quality of VoIP.

### 2.5.2 E-MODEL

The most popular estimative model is based on transmission parameters, which is the E-Model (ITU-T G.107). The E-Model algorithm is an ITU-T standardized computational model for subjective call quality assessment [129]. Moreover, it has been widely accepted as an accurate tool to plan transmission network. Furthermore, the E-Model operates under the assumption that perceived quality impairments are additive [127]. Combining both codec and network impairments results in the Transmission Rating Factor (R) (Equation 1).

$$R = R_o - I_s - I_d - I_{e\_eff} + A \quad (1)$$

Equation (1) is comprised of the following elements:

- $R_o$  represents the signal-to-noise ratio obtained by considering both circuit and room noise;
- $I_s$  refers to the combination of all impairments that occur more or less simultaneously with the voice signal, such as quantization or loudness level;
- $I_d$  represents the impairments caused by the delay of voice signals, such as talker echo, listener echo, and absolute signal delay (mouth to ear delay);

- $I_{e\_eff}$  denotes the impairments caused by low bit-rate codecs and packet loss;
- $A$  is the advantage factor and represents a user's willingness to accept lower call quality in exchange for the advantage of access.

Meanwhile, Equation (2) weighs in many traditional telephony parameters that cannot be measured in VoIP calls, such as  $R_o$  and  $I_s$ . In fact, one of the first studies on E-Model with emphasis on voice transmitted over PSNs had been carried out by [130], where Clark et al., made numerous observations regarding E-Model parameters, which can be assumed by using the default values specified by [129]. As such, the variance between  $R_o$  —  $I_s$  takes in the effective value of 94, while Equation (2) takes the following form:

$$R = 94 - I_d - I_{e\_eff} + A \quad (2)$$

The E-Model is used with the receipt of every VoIP packet. In fact, the accuracy is low for  $R$  values obtained from the first that arrive in VoIP packets. This is influenced by the low measurement accuracy of packet loss and jitter, which function as metrics based on the link between the consecutive packets [11].

As such, the OPNET implementation of the E-Model used in the simulations had been carried out in this work. The remaining terms in Equation (2) are detailed below:

## 1. $A$

The advantage factor  $A$  is the only additive parameter in the formula of E-Model.  $A$  represents a user's willingness to accept lower call quality to compensate for being able to place calls in unusual circumstances, for example, being in a remote geographical area.

Moreover, based on the information found in E-Model [129], Table 6 displays the maximum values  $A$  can take, depending on the call use-case. Further details on the evolvement of  $A$  values, based on the technological progress in telecommunications, are presented in [131].

**Table 6: Recommended values for the advantage factor**

A Type of Communication System

0	Wired connections (e.g. PSTN)
5	Low speed mobility (e.g. indoors)
10	High speed mobility (e.g. vehicle)
20	Remote areas (e.g. multi-hop satellite connection)

## 2. $I_d$

$I_d$  is related to the impairments caused by the delay between the time when the sound is injected within the communications system, as well as the time it reaches the end-point, where it is played. In the telephone systems, this delay is commonly known as mouth-to-ear delay.

Apart from the network delay described in Section 2.2, two more delays exist in VoIP calls:

- Coding/Packetization delay - delay introduced at the source of the VoIP packets by the codec, as it represents the time needed to perform the conversion from analogue signal to digital, the look-ahead delay, and the time needed to compose the VoIP packet.
- Decoding/Jitter Buffer Delay - delay introduced at the destination of the VoIP packets by the decoder to decompose the VoIP packets, as well as to extract and decode the payload into audio samples. Eventually, the waiting time is imposed by the de-jitter buffer, which acts as a queue for audio samples with the aim of reducing the influence of network delay variation on the playout quality.

The ITU-T recommends a total mouth-to-ear delay below 150ms [129], primarily to preserve interactivity in a duplex phone call. Besides, values higher than 150ms would begin degrading the quality of the call, hence making interactivity impossible for values above 400ms.



### 3. $I_{e\_eff}$

$I_{e\_eff}$  reflects the effect of packet loss on voice quality for several lower rate codecs that introduce inter-packet dependencies, where the effect of the packet loss is exacerbated. In E-Model, this is denoted with  $I_e$ . Typical values of  $I_e$  for three of the most used codecs are 0 for G.711, 11 for G.729, and 5 for AMR-NB [132]. Furthermore, some codecs have a feature called Packet Loss Concealment (PLC), which is an interpolation process used to reconstruct lost packets by using past and later audio contents artificially. In E-Model, this codec feature is captured by packet loss robustness ( $Bpl$ ) factor. Several typical values for  $Bpl$  for three of the most used codecs are 4.3 (without PLC) or 25.1 (with PLC) for G.711, 19 for G.729, and 10 for AMR- NB.

Finally, the  $I_{e\_eff}$  can be calculated as given in Equation (3), which refers to the combination of  $I_e$ ,  $Bpl$ , and percentage of packet loss ( $Ppl$ ) [129].

$$I_{e\_eff} = I_e + (95 - I_e) \times \frac{Ppl}{Ppl + Bpl} \quad (3)$$

### 4. R to MOS

R is the evaluation of the transmission on a scale from 0 (poor quality) to 100 (excellent quality). Based on the involvement of the subjects in the actual speech, three situations are possible:

- listening-only (L), when the subjects assess the speech emitted from the speaker;
- talking (T), when the subjects grade the talking side only, which for example, can be influenced by the echo signal;
- and conversational (C), when subjects are involved in an active conversation.

Hence, in order to obtain the MOS for the Estimated Conversational speech Quality ( $MOS_{CQE}$ ), a conversion formula is provided in [129] which converts R values to MOS (Equation 4):

$$MOS_{CQE} = \begin{cases} 1 & \text{if } R < 0, \\ 1 + 0.035 \times R + 7 \times 10^{-6} \times R(R - 60) (100 - R) & \text{if } 0 < R < 100, \\ 4.5 & \text{if } R > 100. \end{cases} \quad (4)$$

The conversion from R to MOS is also given in Table 7, where the quality categories are divided into steps of 10 on the R scale within the range of 50 to 90 (lower limit).

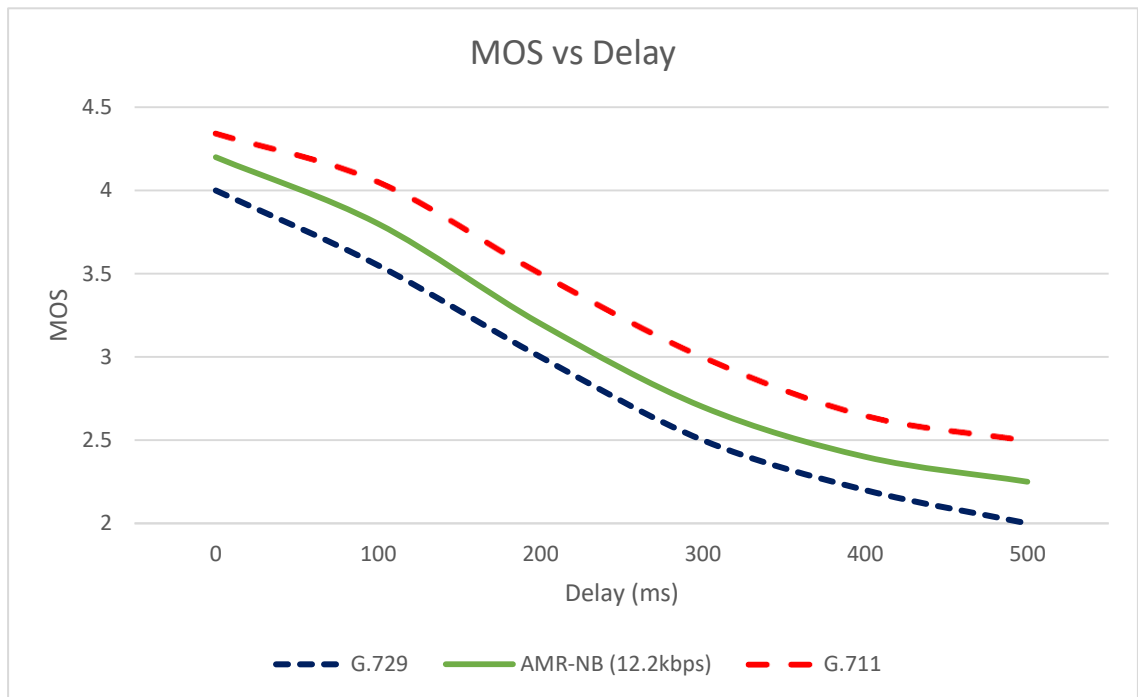
**Table 7: R to  $MOS_{CQE}$  correspondence for estimative assessment**

<b>R-value</b>	<b>MOS</b>	<b>User</b>
<b>(lower limit)</b>	<b>(lower limit)</b>	<b>Satisfaction</b>
<b>90</b>	<b>4.34</b>	Very satisfied
<b>80</b>	<b>4.03</b>	Satisfied
<b>70</b>	<b>3.60</b>	Some users dissatisfied
<b>60</b>	<b>3.10</b>	Many users dissatisfied
<b>50</b>	<b>2.58</b>	Nearly all users dissatisfied

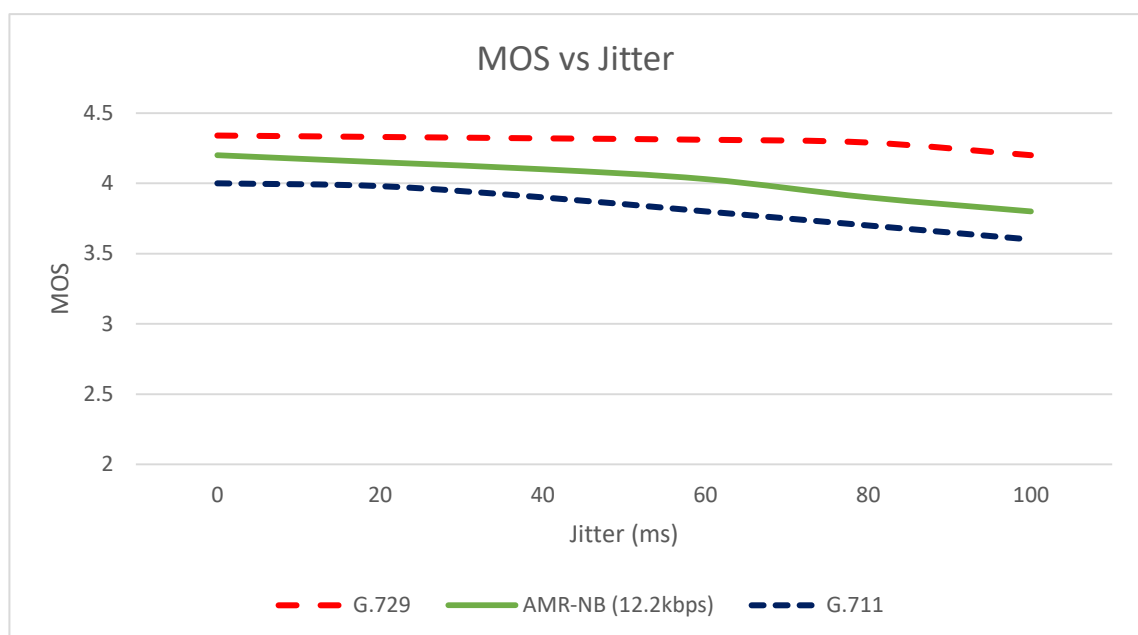
Meanwhile, Figure 14 displays the influence of the delay on the MOS for three of the most used codecs for VoIP. Besides, one can note that the lower the rate of codec, the lower the MOS is (see Table 4).

Next, Figure 15 depicts the influence of the packet delay variation or jitter upon MOS for the same three codecs. The same order between the codecs is maintained as in Figure 14.

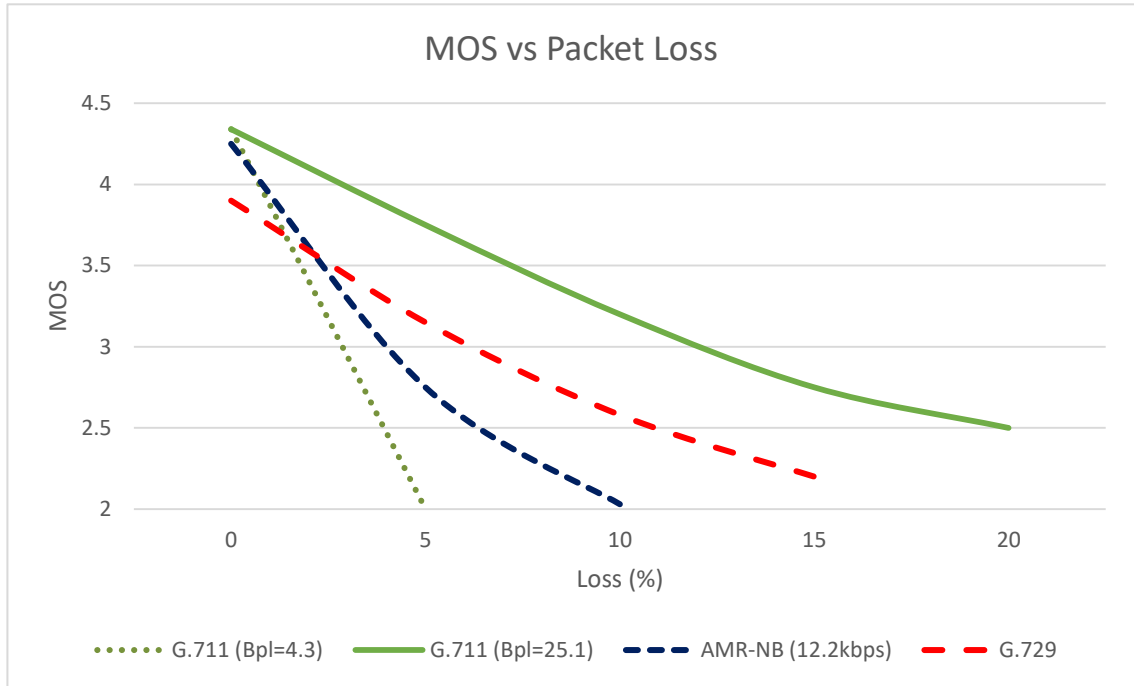
On top of that, Figure 16 presents the influence of packet loss upon MOS for the same three codecs, but with a difference, where G.711 can be used with its PLC feature either enabled or disabled. Here, the trend changes, when compared to the trend exhibited for delay and jitter. In precise, G.729 performs better than AMR and G.711 without PLC (Bpl =4.3), with the fact that G.729 is a very low rate codec. Nonetheless, G.711 with PLC (Bpl =25.1) outperforms the other codecs when its PLC feature is enabled. The results of MOS versus packet loss rate for different codecs are shown in Figure 16. It can be seen that with 0% packet loss, all the voice codecs reach the higher voice quality, the increment of packet loss starting from 2 to 4% clearly affect the voice quality and leads to the overlapping curves for all codecs. G.711(Bpl = 25.1) has the best packet loss robustness feature in all the codecs compared while G.711 (Bpl = 4.3) has the lowest quality no matter with or without packet loss.



**Figure 14: MOS vs Delay per codec**



**Figure 15: MOS vs Jitter per codec**



**Figure 16: MOS vs Packet loss per codec**

## 2.6 REVISITING PREVIOUS WORK

This section presents several past studies related to this research.

### 2.6.1 VOIP QoS AND VOIP SECURITY

VoIP QoS and VoIP security studies have further inspired other research areas. Interesting research regarding the impact of security and encryption on VoIP QoS in [19] [133][134] involved the implementation of various types of encryption algorithms. As the security implementation can directly influence VoIP QoS, the researchers have further investigated the end-to-end security with and without encryption algorithms, including DES, 3DES, AES, and Blowfish algorithms. The results of the network testbed indicated that the impact upon the overall performance of VoIP relied on the bandwidth available and encryption used, where the implementation of encryption algorithms may degrade the voice quality even with adequate network bandwidth.

With that, the blowfish encryption algorithm presents an acceptable level of security that is rapid and provides the least latency, jitter, as well as an efficient algorithm, to minimize lost packets ratios and less impact on voice quality. Through OpenVPN and Linux software programs, lab tests of various call setup had been carried out with and without encryption algorithms [135][86]. The evaluation used MOS as a method to determine

user satisfaction, while the combination of using varying network scenarios present various network issues. Both variety and varied scenarios, including variously available bandwidth, number of calls, and types of encryption algorithms, had been tested in this research, along with the evaluation of MOS measurements that was recorded as well. OpenVPN is a popular open source software program for IPSec implementation, while the encryption algorithms imposed in this research were AES, Blowfish, and 3DES. The results showed that the Blowfish algorithm displayed superior performance effectively with higher MOS values.

Other than that, a VPN cryptography had been developed and designed for VoIP services, where the experiment embedded varying network topologies, different types of encryption algorithms, such as DES, AES, and Blowfish, along with Linux open source operating system [136]. The results exhibited that the encryption algorithm was able to impose security for VoIP services, in which the MOS values recorded for Blowfish encryption algorithm had been higher among the rest. Besides, implementing VoIP without any security mechanism may be exposed to many risks and privacy intrusion. As such, the use of Blowfish encryption algorithm was proposed to encrypt the audio communication between the VoIP [87]. In fact, the study was conducted in a lab with two softphones using G.711 voice codec, while the results were based on three performance parameters, which are: network delay, jitter, and packet loss. The findings obtained from the research displayed that increment in network latency generated excellent quality for voice, while increment in jitter led resulted in acceptable voice quality, and for packet loss, no remarkable change was noted, when compared to VoIP without encryption.

In addition, the design of prototype VoIP network and the virtual modelling using OPNET simulation were examined by looking into the effects of security mechanisms upon network performance [137][138]. Both effects and performance analyses of different types of VoIP security mechanisms implemented by using OPNET simulation was also studied [52][139][140][64]. The research was conducted and designed by using OPNET simulator, as well as network QoS results, including delay, jitter, packet loss, and MOS values.

Besides, a number of studies had been involved in OPNET simulation software program [141], where the selection of suitable voice codecs relied on the network QoS of VoIP in varied networks. Furthermore, in order to determine the most effective and efficient results for network QoS, various simulations had been carried out by using G.711,

G.729A, G.723.1, and GSM-FR codecs using wired and wireless network designs. For instance, various codecs had been investigated, including G.711, G729, and G.723, whereby the simulation was run on OPNET Modeler 14.5. The findings showed that the best network performance could be achieved via trade-off between codec bandwidth requirements and desired quality [142].

In addition, application of apt voice codecs could contribute to better voice quality, while MOS results depended on available network bandwidth and network congestion. Meanwhile, as the VPN is presumed to have a negative impact on VoIP performance, OPNET simulation was employed to simulate the behaviour of VoIP that ran over IP VPN tunnel [43]. Moreover, this study used several scenarios, in which the performance of VoIP call had been compared for end-to-end delay, jitter, and call setup time for varying network configurations. As a result, the findings displayed that the combination of network design and VPN influenced end-to-end delay and network jitter.

Other than that, the thesis generated by Amna [143] regarding securing VoIP performance measurement presented the instruments for securing VoIP calls, the measurement of network QoS, and voice quality with varied VoIP services, which also embedded the implementation of security layer. The findings portrayed that the relationships between VoIP performance, VoIP security, VoIP services type, and the implementation of security had little impact upon VoIP voice quality and network QoS. Research conducted by [144] defined a lots of potential Caller ID spoofing attack, researcher proposed and design end-to-end detection of Caller ID mechanism to secure VoIP calls.

### **2.6.2 VOICE CODECS AND VOICE QUALITY MEASUREMENT**

A study looked into the role of several selected voice codecs to evaluate the performance of IPSec secured VoIP [124]. In fact, the impact of IPSec from past studies exhibited degradation in voice quality under limited bandwidth, where the researcher offered quantitative justification that the impact of IPSec on VOIP in terms of packet loss, delay, jitter, and voice quality of MOS values can be reduced by applying suitable voice codecs. As such, G.711 and Speex codecs were tested with apt codec, where the overhead linked with IPSec could be minimal and desirable voice quality may be attained with relatively higher MOS values. Moreover, the usage of various voice codecs and the effects of

cryptography on perceived QoS for VoIP streams with G.711, G.723, and G.729 had been investigated [145], in which MOS values and E-Model were employed as measurement methods to encrypt VoIP transmission. The findings showed that the IPsec encryption of VoIP was strongly related to the payload sizes, the choice of codecs, and the types of encryption algorithms. Meanwhile, Daengsi [146] ran an experimental design using G.722, G.711, G.729, and G.723.1 voice codecs, whereby the VoIP testbed system recorded the voice quality and the quality of varied types of voice codecs were compared in MOS values, which resulted in the proposal of E<sup>2</sup>-Model (enhancement of the E-Model in Thai language). Furthermore, the impact of IPsec on a VoIP network in terms of packet loss, jitter, and MOS percentages were determined and the results portrayed that the selection of suitable voice codec can offer better voice quality, reduce the amount of network overhead, and increase bandwidth utilization [124].

### **2.6.3 ENHANCES BLOWFISH ENCRYPTION ALGORITHM**

A study by Riza [147] pertaining to the implementation of standard Blowfish algorithm in C++ programming language, the code in C++ tested for the performance evaluation under various types of data had been examined. In fact, the study had been related to algorithm encryption/decryption time processing, throughput, security strength, and cryptanalysis.

Meanwhile, a security analysis of Blowfish algorithm by [148], analyzed the randomness of the Blowfish algorithm output, where the implementation of algorithm employed both C++ programming language and MATLAB programming to perform cryptanalysis in terms of avalanche effect and correlation coefficient. The findings displayed that the Blowfish algorithm generated good avalanche text from the second round and a good non-linear correlation between plaintext and cipher text.

The modified Blowfish encryption algorithm through a new method to generate S-Boxes and P-arrays was initiated by [149]. The modified algorithm was applied on speech coding algorithm G.729, where the method decreased time complexity in generating S-box and P-arrays. Moreover, the implementation of the modified Blowfish algorithm was tested by using MATLAB, in which the output was analyzed by using avalanche effect. In addition, the security measurement exhibited that the modified Blowfish offered a

similar level of security with the original Blowfish algorithm. However, the modified Blowfish highlighted an advantage with less computational overhead in key generation. In another study, the “F” function of Blowfish algorithm was modified by mixing XOR [150]. Later, four cases based on encryption quality, correlation coefficient analysis, key sensitivity test, and size of data file after encryption had been looked into. Data images were applied in this research and the results showed that the modified Blowfish algorithm was more secured and compact, when compared to the original Blowfish algorithm. The performance analysis of Blowfish and its modified version by [151] evaluated the algorithms via encryption quality, key sensitivity, histogram, and correlation coefficient analysis, in which the encryption had been tested using several digital images, exhibited that the modified version of Blowfish algorithm did not violate any security requirement. Research by [102][152][101] also discussed about the need of light weight cryptography and their design differences with normal block cipher, the comparison in term of energy, changing data types such as text or document, power consumption, changing packet size, and changing key size show blowfish algorithm has a better performance than RC2, DES, 3DES and AES.

## **2.7 SUMMARY**

This chapter highlights an overview on the works done in the field of VoIP services and application.

As such, several key topics have been discussed in this chapter, an overview of VoIP related to transport protocols, traffic metrics, QoS, security issues, and other related methods to assess voice quality.

Furthermore, the history of VoIP highlighted this Internet application as a topic with high interest in the area of communications, with the emphasis on Quality of Service aspects, which could affect the quality of VoIP call. In fact, an entire section is dedicated for measuring VoIP call quality. The presented security issues related to encryption and IPSec are some factors that can influence the quality of VoIP call.

Moreover, in this chapter, VoIP audio codecs technologies have been discussed, as they are being used for both users and service providers at present time. Voice quality assessment implemented by using MOS with a range of 1 to 5 rating score is depicted in this chapter as well.



Several prior studies are also reviewed so as to suggest ideas for implementation in this particular research in terms of methodology, tools, experimental runs, and procedures.

The concepts presented in this chapter should enhance one's understanding concerning the methodology presented in Chapter 3.

# CHAPTER 3

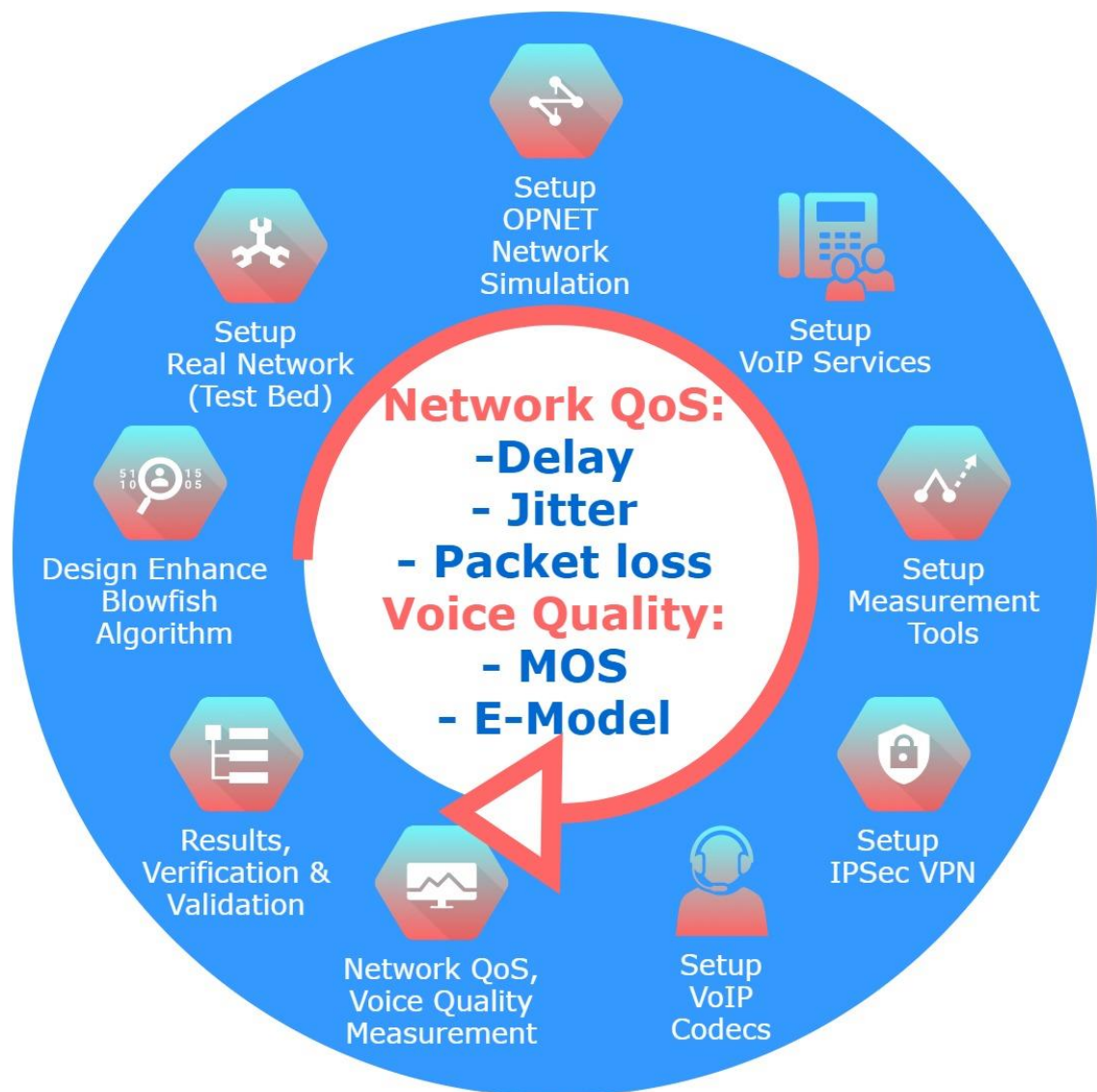
## METHODOLOGY

This chapter discusses the methodology applied in this research to gather relevant data in accordance to research objectives, so as to answer the research questions outlined in Chapter I. Section 3.1 explain the process involved in experimental and preparation. Section 3.2 explain about encryption algorithm enhancement design. Section 3.3 and 3.4 describe network testbed design and network simulation tool used in order to implement the research experimental design test scenarios and method for analysis. In this chapter, the aspects of research design, experimental tests on bed network design, OPNET simulation setup, as well as verification and validation analyses, are elaborated.

### 3.1 EXPERIMENTAL AND PREPARATION

The methodology of the study is illustrated in Figure 17 on page 66. The initial stage began by designing the enhanced version of Blowfish algorithm, where C++ programming was implemented in this research design, similar to that in prior studies [153] [147]. Next, some procedures and simulation activities were performed with C++ and MATLAB programming language in order to evaluate the performance of the algorithm. After that, a security analysis was carried out to compare the results of the algorithms based on some parameters, which are the speed of algorithms to encrypt/decrypt (execution time), metrics that scheduled algorithms for optimisation (throughput), as well as cryptanalysis using avalanche effect and correlation coefficient. On top of that, the simulation of design embedded varying data packet sizes and types of data (audio, video, and text). Furthermore, the real network (testbed) was set up in the University of Liverpool, UK labs, and MARA College in Malaysia by using specified devices, including gateway router and Internet connections that were provided by Internet Services Provider (ISP). In this network testbed, the IPsec VPN setup with both standard and enhanced Blowfish encryption algorithms had been examined, together with the setup of VoIP services, monitor service performance, analyses of gathered data and right until validation analysis. By designing and testing various network test scenarios, different experiments had been carried out based on alternating settings and network

parameters, which allowed implementation of a configuration with various traffic loads over a link to test both the outcome and performance of the changes on encryption and voice codec. As such, Wireshark and Jperf had been applied for simulation as the primary network performance tools. Wireshark verified the encryption process in a network testbed, while Jperf recorded network throughput, mean jitter, and packet loss. Besides, a virtual OPNET network simulation was designed in order to assist the testing, analyses, and verification processes.



**Figure 17: Research methodology**

Several factors might affect the quality and performance of VoIP calls and services; these factors include the VoIP protocol, available bandwidth, security imposed, VoIP equipment specification, network delay, jitter, packet loss and network architecture to provide QoS [154]. Figure 18 shows the parameter that affects VoIP QoS, the highlighted parts indicate parameters that has been used in this research.



**Figure 18: Parameters affect VoIP QoS**

In this research, for each parameter testing, there are different techniques used with a variety of tools. Table 8 below summarizes the correlation between these three aspects; the methods, devices and parameters. Firstly, to test the performance parameter, the Blowfish encryption design, performance and security testing is done through using the C++ and MATLAB programming. Secondly, for the Network QoS (Delay, jitter & packet loss), Voice Codecs (G.711, G.729 & AMR-NB), and Network Bandwidth (100 Mbps) & Background traffic (50Mbps, 100Mbps, 150Mbps & 200Mbps) and Voice quality (MOS & E-Model) parameters, the method used is Real Network testbed setup with and without encryption algorithm (Standard Blowfish & Enhanced Blowfish Algorithm) which uses the Ekiga softphone, Iperf, Wireshark/ TCPdump, OpenSwan IPsec VPN, RTP tools. For the Network QoS (Delay, jitter & packet loss), Network Bandwidth (1 Mbps, 10 Mbps & 100 Mbps) and number of calls made (10 calls & 200 calls), and Voice quality (MOS & E-Model) parameters are tested through the Virtual Network Simulation with and without encryption algorithm (Standard Blowfish & Enhanced Blowfish Algorithm) through the OPNET/Riverbed network simulator device.

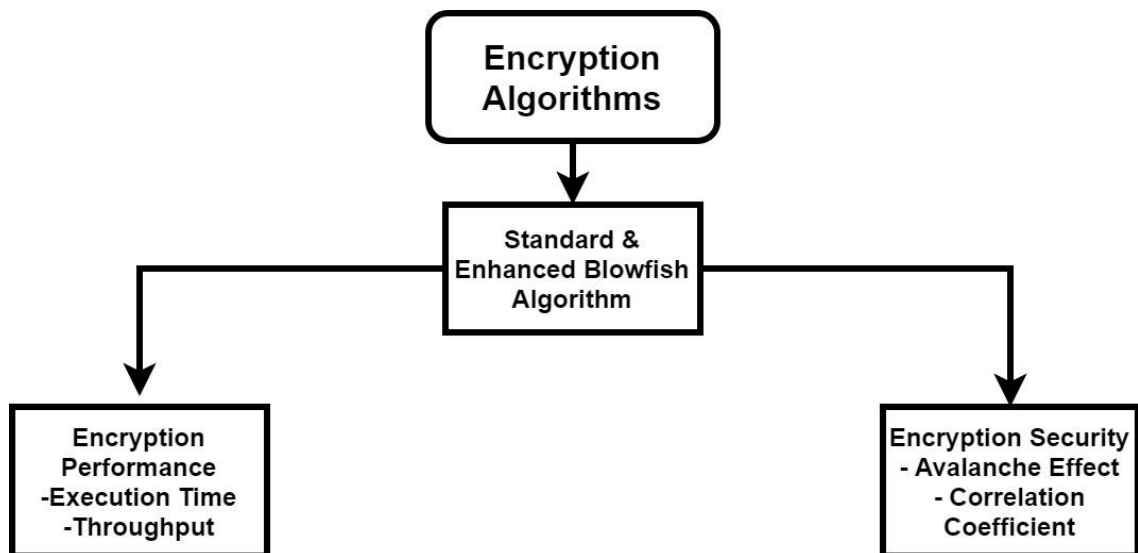
**Table 8: Correlation between the methods, tools and parameters**

Methods/ Techniques	Devices/ Tools	Parameters
Blowfish encryption design, performance and security testing	C++ and MATLAB programming	Performance (Execution time & throughput) Cryptanalysis (Avalanche effect & correlation coefficient)
Real Network testbed setup with and without encryption algorithm (Standard Blowfish & Enhanced Blowfish Algorithm)	Ekiga softphone, Iperf, Wireshark/ TCPdump, OpenSwan IPsec VPN, RTP tools	Network QoS (Delay, jitter & packet loss) Voice Codecs (G.711, G.729 & AMR-NB) Network Bandwidth (100 Mbps) & Background traffic (50Mbps, 100Mbps, 150Mbps & 200Mbps) Voice quality (MOS & E-Model)

Virtual Network Simulation with and without encryption algorithm (Standard Blowfish & Enhanced Blowfish Algorithm)	OPNET/Riverbed network simulator	Network QoS (Delay, jitter & packet loss) Network Bandwidth (1 Mbps, 10 Mbps & 100 Mbps) No. of calls made (10 calls & 200 calls) Voice quality Measurement (MOS & E-Model)
--	----------------------------------	--

### 3.2 ENCRYPTION ALGORITHM ENHANCEMENT

Encryption algorithms in this thesis involve both the standard Blowfish and enhance Blowfish algorithm where the algorithm is reconstructed on a Feistel network and S-Boxes which will be further explained in chapter 4, section 4.1. Figure 19 below depicts the performance and security measurement process. Encryption performance is measured through execution time; encryption and decryption time whereas for security cryptanalysis measurement, both encryption algorithms are tested through avalanche effect and correlation coefficient experimentation. These results are compared to ensure that the enhanced Blowfish encryption will produce a better performing Encryption Algorithms as well as maintaining adequate security characteristics.



**Figure 19: Standard Blowfish encryption and enhanced Blowfish algorithms measurement**

### 3.3 NETWORK TESTBED DESIGN

The experimental setup, as portrayed in Figures 20, 21 and 22, looked into the effect of encryption algorithms and the varied backgrounds of packet sizes upon QoS network and VoIP voice quality. The network design involved equipment setup with specification listed in Appendix B:- Tables 16 until 18, by using an open source software program and a softphone with Windows operating system. The network segmentation was performed by using two different VLAN; VLAN1 and VLAN2, which simulated two varied locations. Next, a larger network was broken down into smaller ones, where two networks were connected with router#1 and #2, which represented the bottleneck node in the network as the encryption algorithms were applied in VPN connection with 1Gbps speed links. Besides, the voice traffic had been generated by RTP Tool Box and PackETH software programs [155] [156].

The Ekiga softphone and the RTP toolbox softphone with various types of voice codecs from different VLAN communicated and manipulated multiple voice sessions simultaneously. Moreover, different ranges of Pulse Code Modulation (PCM) voice codecs had been tested in this research, which were G.711, G.729, and AMR-NB at 12.2 kbps encoding scheme. G.711 was selected because it is a popular codec that can offer excellent voice quality when compared to other narrowband codecs [113]. Meanwhile, G.729 is a popular codec for a wide network area [157], and AMR-NB is mostly used in GSM [158], which had been employed in this study as a comparison encoding scheme. Other than that, Wireshark and Iperf were applied as packet scan and monitoring tools in order to capture and to analyse both QoS and voice quality. The packet that carried voice data, which was transmitted between sender and receiver, had been captured by Wireshark and converted to eXtensible Markup Language (XML). The gathered data were used to calculate QoS network.

Besides, various types of background traffic had been generated by using packETH [159], which functioned as the main simulation tools to test and to flood the network with varied types of generated packets. In this testbed, three packet sizes with categories “small”, “medium”, and “big” had been employed [160]. The variety in packet sizes was required to measure the impact of packet size variation upon data transmission so as to maintain end-to-end network performance [161].

Moreover, the VoIP communication setup in this network testbed had been investigated based on the following scenarios:

- a) No Security: No encryption algorithm was used for VoIP communication and network traffic, whereby this setting was used as benchmark.[124][87]
- b) Standard Blowfish encryption algorithm: The VoIP communication and network traffic were imposed with OpenSwan IPsec with the standard Blowfish encryption algorithm.[124][87][162]
- c) Enhanced Blowfish encryption algorithm: OpenSwan IPsec and enhanced Blowfish encryption algorithm were implemented on VoIP communication and network traffic.[149][163]

The measurements of network QoS and voice quality using MOS score in the network testbed had been performed to assess the impact of encryption algorithms under various network parameters and allocation of varied bandwidths using the three scenarios mentioned above.



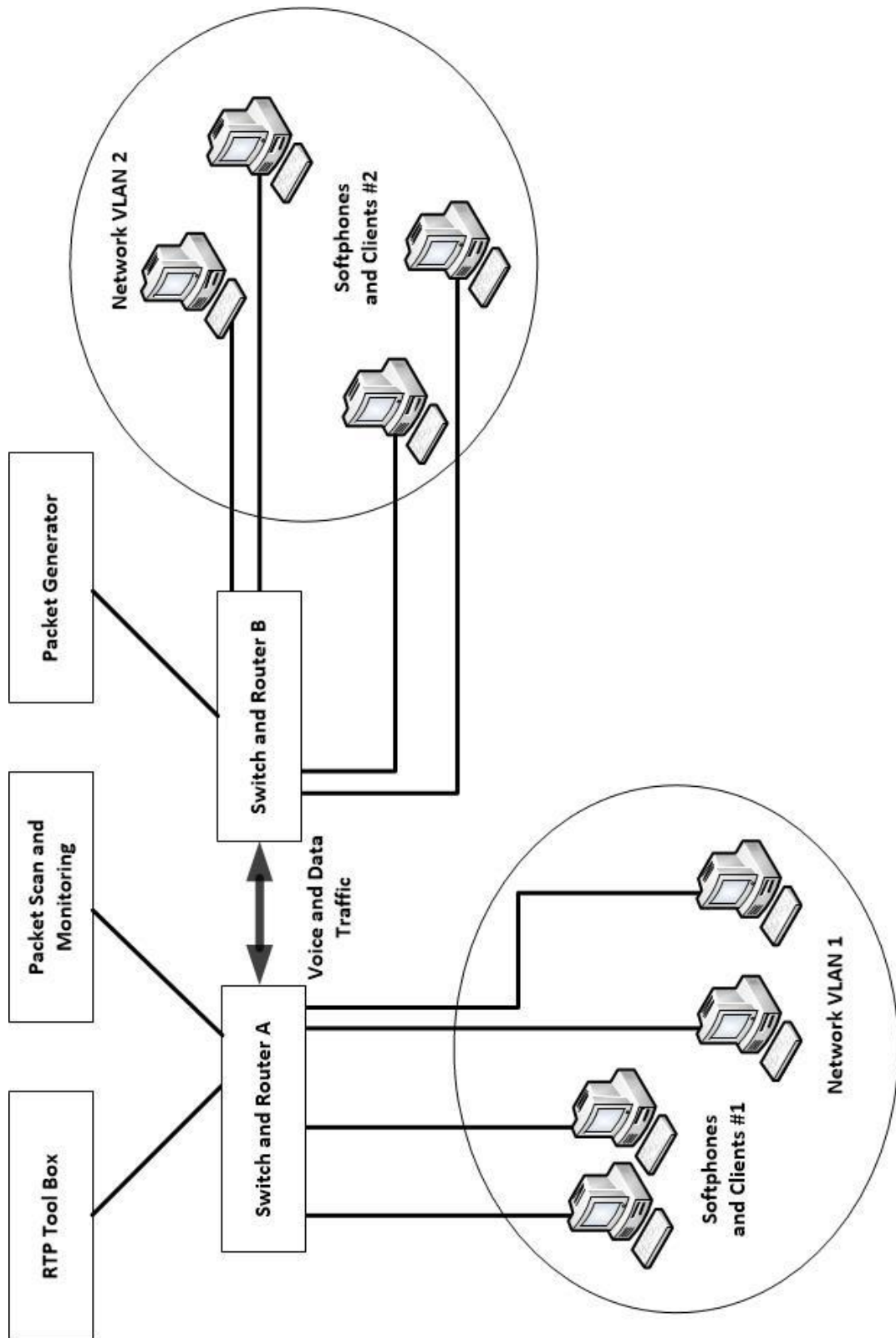
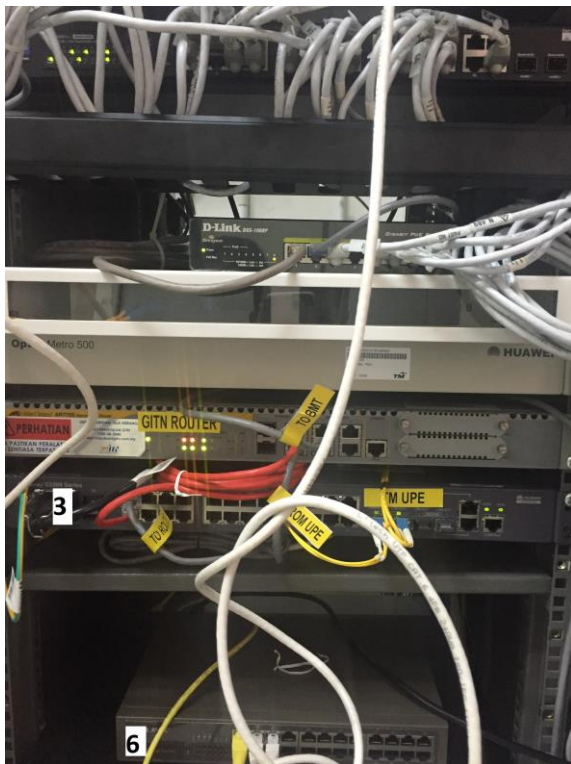


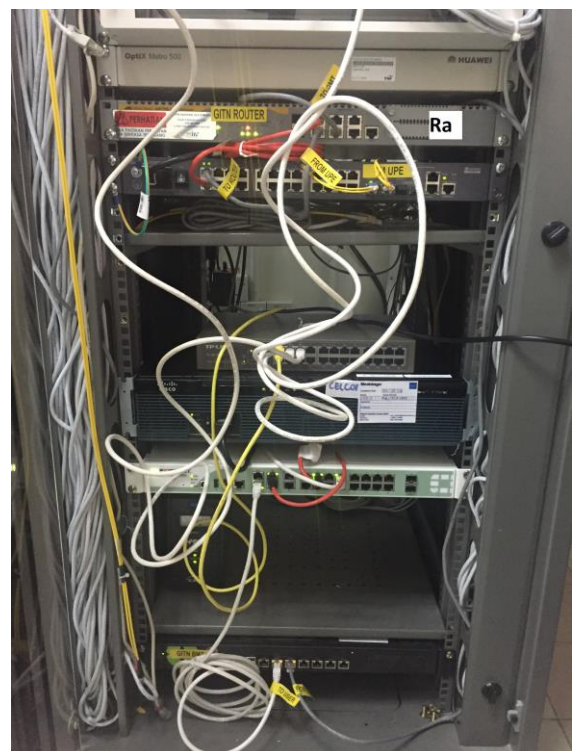
Figure 20: Network testbed diagram



**Figure 21: Physical network testbed**

**Legend:**

1-Monitoring Screen ; 2-WIRESHARK machine; 3-Switch #1; 4-IPerf/Jperf machine; 5-RTP ToolBOX; 6- Switch #2; 7-Client# 1; 8-Client# 2;



**Figure 22: Specific devices**

**Legend:**

**9**-Routers; **9a**-R1(ISP); **9b**-R2; **9c**-R3; **10**-network monitoring screen;

**11**-Keyboard, visual player unit, mouse (KVM)

### **3.4 NETWORK SIMULATION TOOLS**

A Network simulator refers to software tools that deploy, design, run, and predict the behaviour of network devices. Moreover, it is comprised of both hardware and software components that enable estimation associated to virtual network behaviour. In fact, there are many advantages in using the network simulator, for example, efficient time management in developing working hardware and software for networking devices, along with various choices of hardware and specific network scenarios, which can imitate the real network environment with simplicity and lower cost. Besides, this tool offer the opportunity to design, modify, test, and study the behaviour exerted by a network design to predict its strength and weakness before being deployment into the real world environment. As such, various types of network topologies with complex architectures can also be tested, which have enabled many researchers to investigate the performance and the behaviour of network with added flexibility, scalability, and modularity, thus dismissing the “trial-and-error” phase in hardware implementation [164]. As for this research, the OPNET (OPTimized Network Engineering Tool) simulator [165], which is presently known as Riverbed simulator [166], had been employed to perform simulation network design for various types of network scenarios.

#### **3.4.1 OPNET/RIVERBED SIMULATION**

The OPNET Simulator offers the capabilities to test, design, and develop a comprehensive network environment with specification, simulation, and performance analysis of network. Besides, different areas of network, including single local area network (LAN) and wide area network (WAN) to global satellite network, can be implemented by using OPNET simulator. Furthermore, based on the discrete event system mechanism, OPNET can simulate by modelling the events setup by user, also included in the OPNET, the programming tools for user to program each model and execute real time network protocol [138]. As the most popular technique in computer networking, the vital elements are “events”, which enable the model to change at certain discrete point [167]. OPNET, being a packet-based network simulator, measures network performance metrics based on the packet simulation generated. Moreover, with the fact that it is written in C++ programming language, it can simulate existing models and scenarios or a user can provide their codes written in C or C++, where the modeller would execute the “event” by scheduling and phasing in accordance to the selected model and

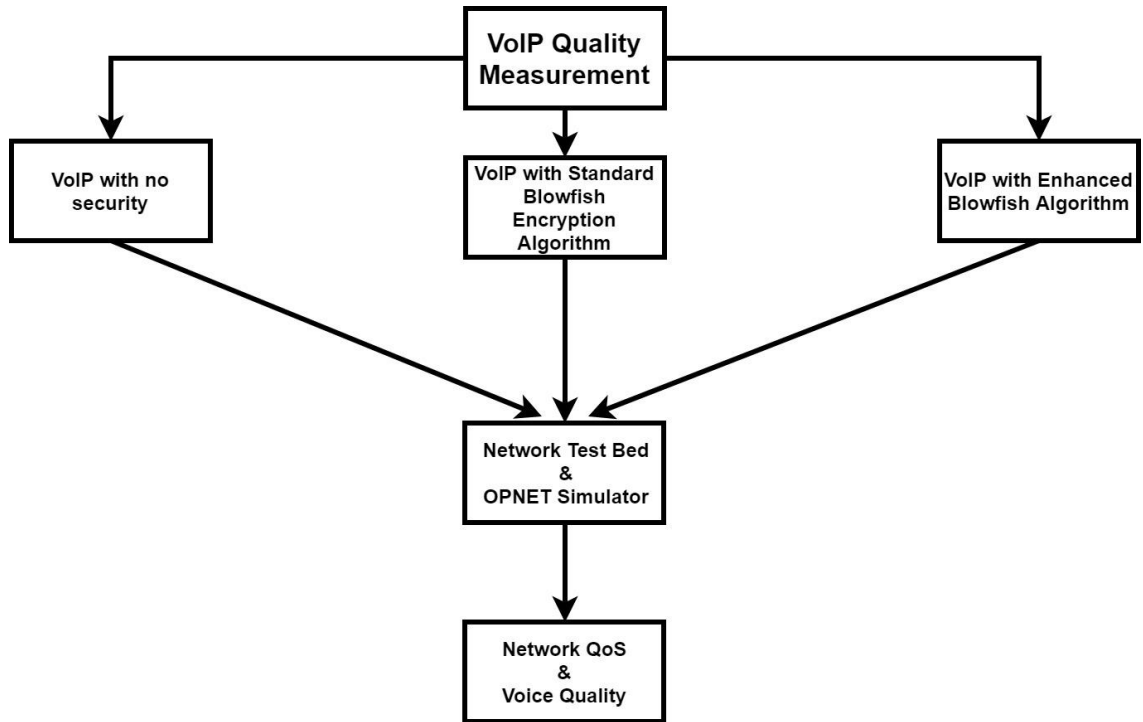
setting. Additionally, the graphical user interface (GUI) offers excellent visualization and simplified network design to support the development of model, protocols, and components.

As for this research, the OPNET simulation was applied to design various types of network topologies located in several distributed branches. By using the same scenarios implemented in the network testbed, VoIP services with varying voice codecs (G.711, G729, and AMR-NB), along with and without encryption algorithms, had been tested by using the OPNET simulator. The configuration and settings were further adjusted with various traffic loads over a link to determine the outcomes that derived from the altered codec performance, as well as the impact of security imposed. More details regarding the implementation are presented in Chapter 4.

### **3.4 SUMMARY**

In order to measure the impact of encrypted algorithm upon QoS network and VoIP voice quality as illustrated in Figure 23, a real network testbed was built and used to perform VoIP communication experimental tests. In addition, the obtained results were verified and validated by comparing the measurement of a variety network designs and topologies via OPNET simulation. Moreover, the environment was setup to evaluate the performance exerted by both the Blowfish encryption algorithm and the proposed enhance algorithm under various network scenarios.

In addition, the quality of voice was also measured by using mean opinion score (MOS), which was performed by both simulations (network testbed and OPNET) with varying voice codecs. In fact, the combinations of varied voice codec for both encryption algorithms were measured and evaluated to estimate the codec and the encryption algorithm that performed exceptionally and provided better voice quality.



**Figure 23: VoIP quality measurement**

## **CHAPTER 4**

# **BLOWFISH    ENCRYPTION    ALGORITHM**

## **ENHANCEMENT AND IMPLEMENTATIONS**

This chapter presents the simulation and implementation stages employed in this research. As such, a total of six stages related to simulations and implementation had been embedded. The initial stage involved both designing and testing the enhanced Blowfish algorithm with C++ programming language, where the standard Blowfish encryption algorithm was redesigned and modified to improve the aspects of execution time and throughput. Next, the security measurement, along with cryptanalysis, had been implemented by using the MATLAB programming language. Moving on, the second stage referred to the configuration of the network testbed setup, which incorporated the implementation of VoIP end devices that were connected to the existing network provided by ISP, while the OpenSwan VPN provided secured site-to-site IPSec in tunnel mode by employing both standard and enhanced Blowfish encryption algorithms. Meanwhile, the third stage involved installation and configuration of VoIP clients, as well as the measurement of VoIP voice quality. In addition, the Ekiga softphone was installed into both sender and receiver machines in the network testbed. As for the fourth stage, the OPNET simulator was implemented to simulate the VoIP applications by using various voice codecs and a variety of network behaviour. The fifth stage, through the use of various types of voice codecs, along with IPSec security parameter, the quality of voice was measured by using the gathered MOS, whereby the related values were collected, processed and compared. Finally, the last stage referred to validation and verification, where both the standard and the enhanced Blowfish algorithms, which were simulated by using OPNET, had been compared with hardware implementation. With that, MOS values were validated, data were gathered, and the sample results were verified.

## 4.1 FIRST STAGE: ALGORITHM EXPERIMENT & SIMULATION

### 4.1.1 ALGORITHM DESIGN AND TEST BY USING C++ SIMULATION

A key that ranged from 32 bits to 448 bits was used in Blowfish algorithm. From that key, 18 32-bit subkeys and four 8 X 32 S-boxes containing a total of 1024 32-bit entries were generated, thus concluding in a sum of 1024 32-bit values or 4168 bytes.

The Blowfish is a 16-round Feistel Structure, as illustrated in Figures 24 and 25. Every round is made up of a key- and data-dependent substitution, as well as key-dependent permutation. As for F function, as depicted in Figure 26, all the operations embedded 32-bit words and XOR. The only additional operations, for every round, had been performed in the following way [168]:

- Each block was broken into half.
- The new left half derived from the right half.
- After performing XOR on the left half, the results were derived upon performing F function at the key and the right half.
- The essential round was obtained even if the function **F** (Equation 5) was not turned upside down.

$$\mathbf{F(xL)} = ((\mathbf{S_{1,a}} + \mathbf{S_{2,b}} \bmod 2^{32}) \mathbf{XOR} \mathbf{S_{3,c}}) + \mathbf{S_{4,d}} \bmod 2^{32} \quad (5)$$



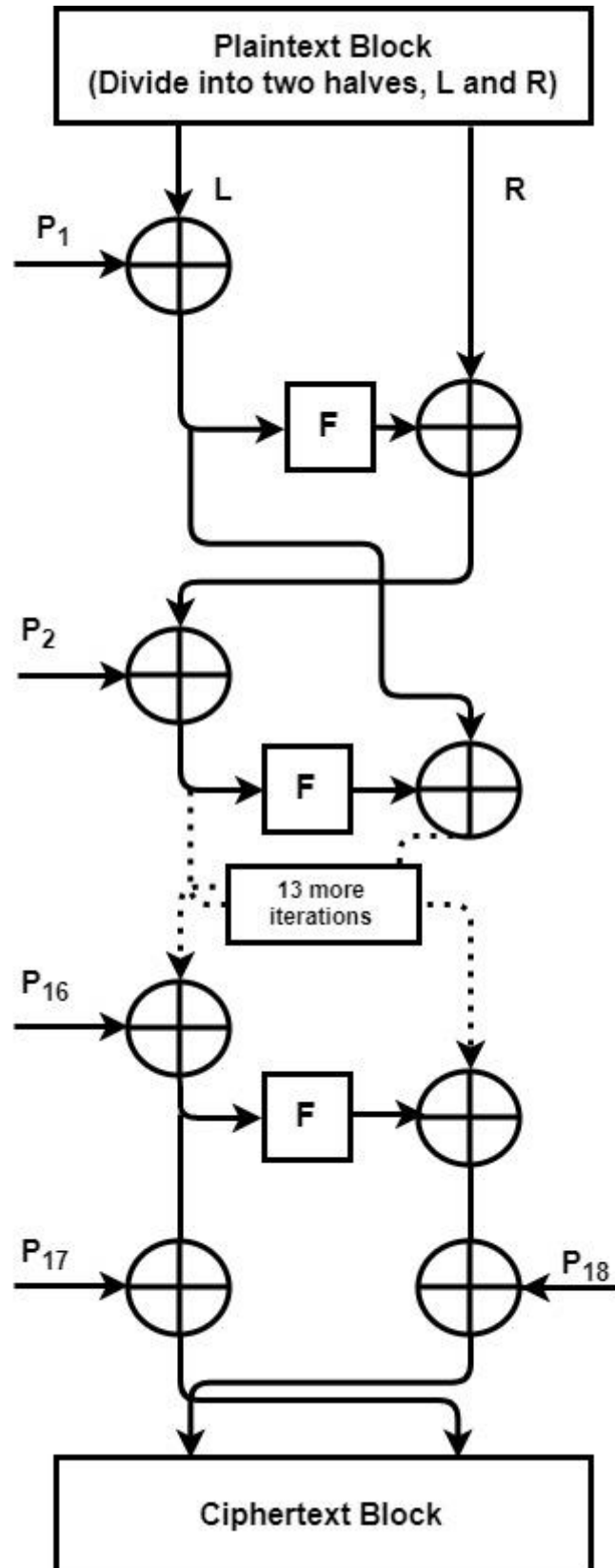


Figure 24: Blowfish Feistel Structure of 16

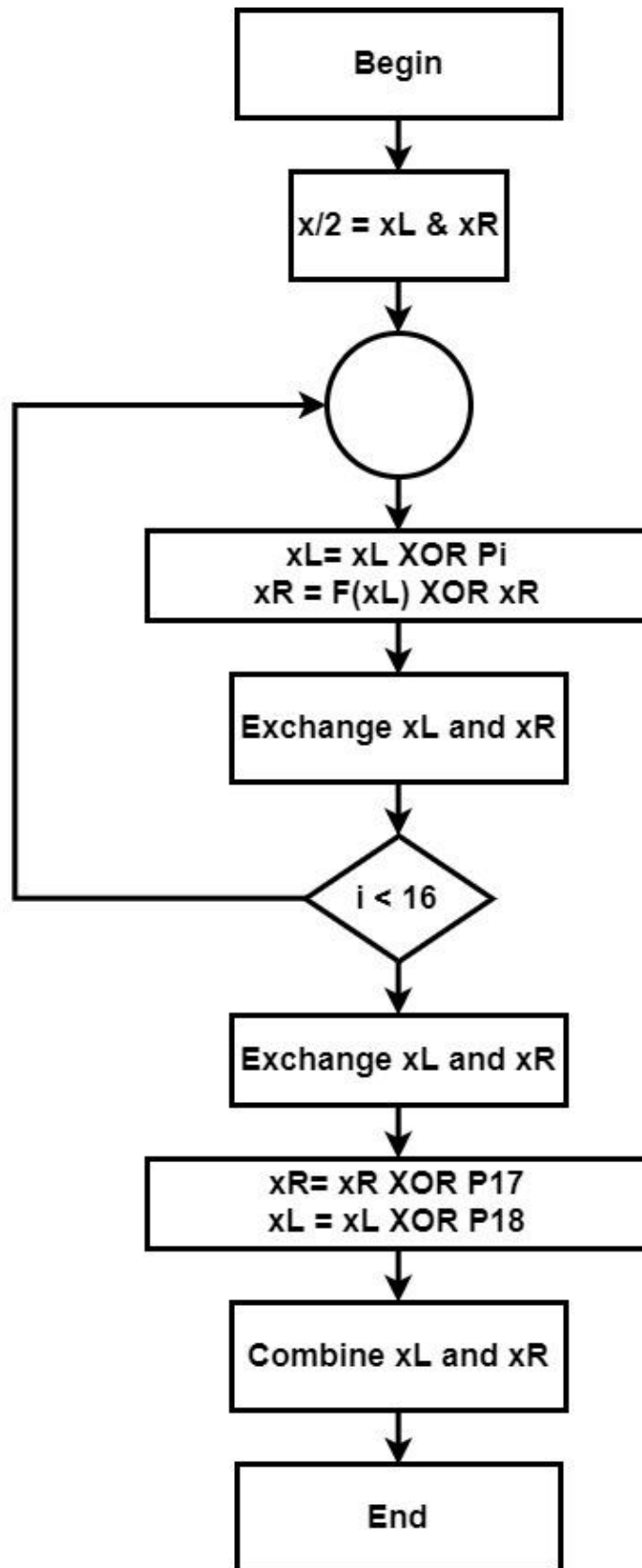
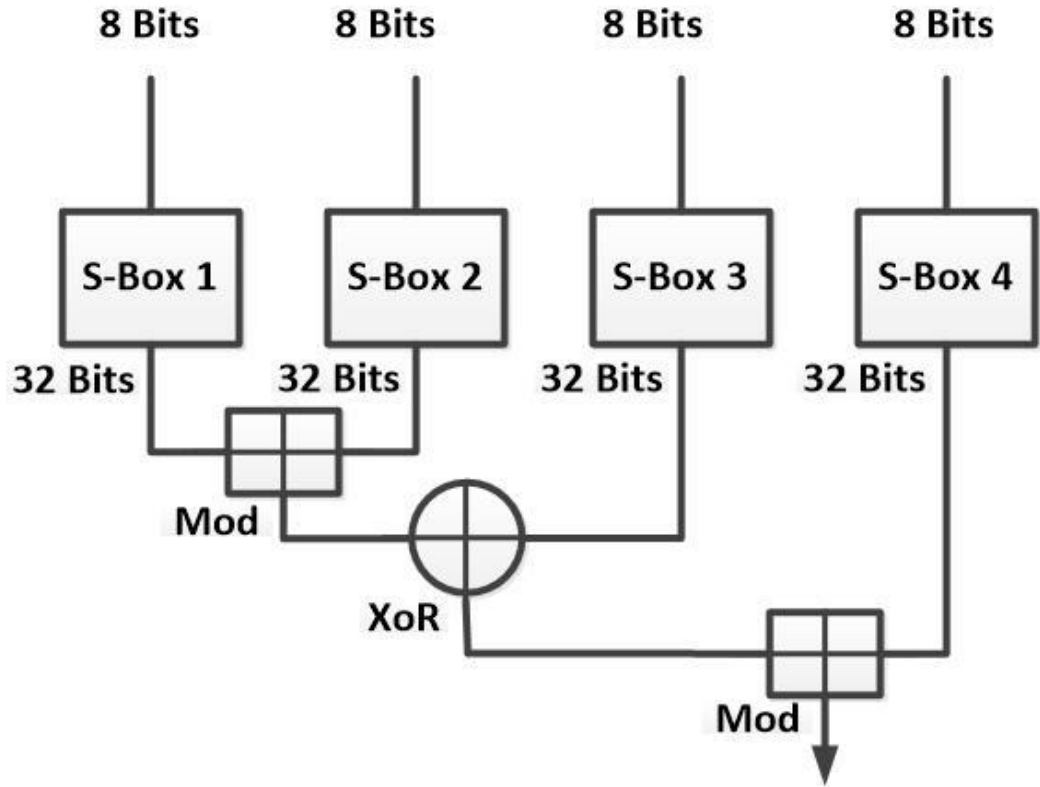


Figure 25: Blowfish algorithm encryption process flow



**Figure 26: Graphic representation of F function in Blowfish algorithm**

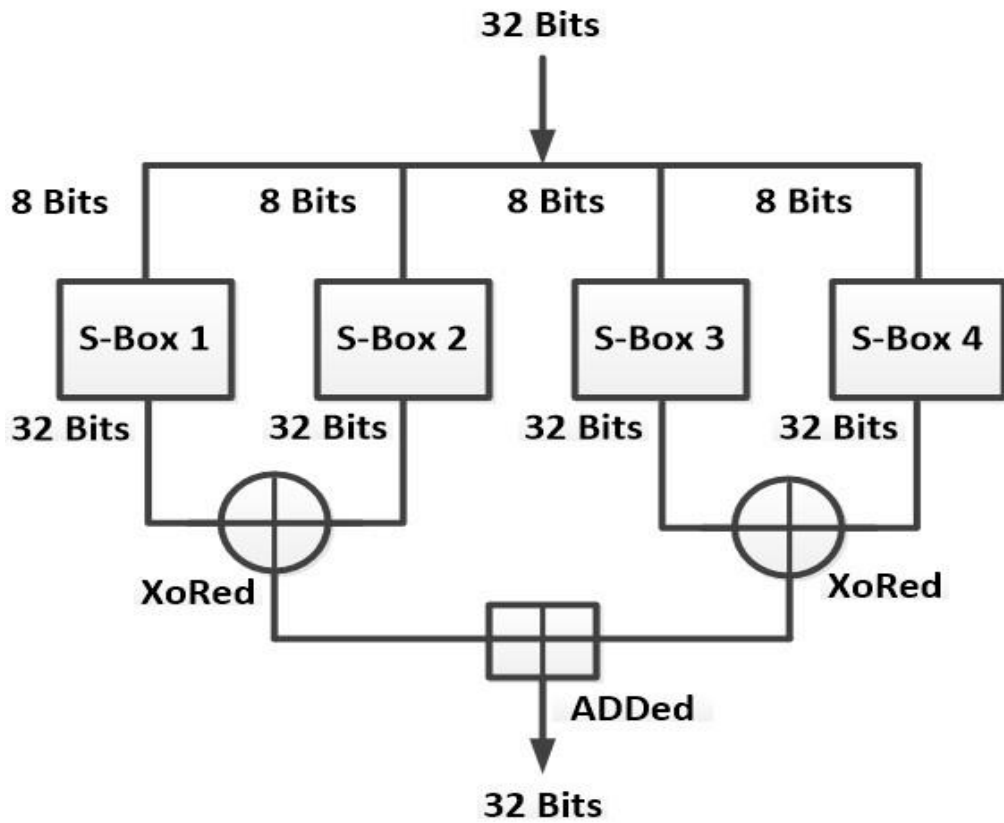
- Adhering to a prior study by [95], the standard Blowfish Algorithm in this research had been redesigned, as illustrated in Figure 27, by redesigning the original operation of the Blowfish algorithm **F** function (Equation 6):

$$F(xL) = (S_{1,a} \text{ XOR } S_{2,b} \text{ mod } 2^{32}) + (S_{3,c} \text{ XOR } S_{4,d} \text{ mod } 2^{32}) \quad (6)$$

- P-array and two S-Boxes were initialized
- The sub keys were prepared by encrypting both the key and P-Array
- Two S-Boxes used the F function to encrypt the values of S-box
- The 64-bit input plain text was divided into 32-bit halves (left (xL) and right(xR))
- The 32-bit left half xL was XORed with sub key P1 and assigned into xL. Later, the xL was fed into the F function.

- The modified F function consisted of four S-boxes. The F function split the 32-bit of input into four 8-bit blocks; Blocks a, b, c, and d.
  - 8 bits from S-box (S1) and 8 bits from S-box (S2) were XORed
  - 8 bits from S-box (S3) and 8 bits from S-box (S4) were XORed
  - The results from the operation of XORed S1 and S2 were applied with the results retrieved from the operation of XORed S3 and S4.
  - The process of F Function is depicted as follows: xL was divided into four: 8-bit blocks: a, b, c, and d. The results are given below (Equation 7):

$$F(xL) = (S_{1,a} \text{ XOR } S_{2,b} \text{ mod } 2^{32}) + (S_{3,c} \text{ XOR } S_{4,d} \text{ mod } 2^{32}). \quad (7)$$



**Figure 27: Modification F function of enhanced Blowfish algorithm**

The Pseudocode of enhanced F function with S-boxes operation is given below:

Step 1: xL was divided into four 8-bit blocks: a, b, c, and d

Step 2:  $(S_{1,a} \text{ XOR } S_{2,b} \text{ mod } 2^{32})$

Step 3:  $(S_{3,c} \text{ XOR } S_{4,d} \text{ mod } 2^{32})$

Step 4:  $F(xL) = (S_{1,a} \text{ XOR } S_{2,b} \text{ mod } 2^{32}) + (S_{3,c} \text{ XOR } S_{4,d} \text{ mod } 2^{32})$

### **Pseudocode of Encryption**

Step 1: The 64-bit input data were divided into two 32-bit halves (left and right): xL and xR

Step 2: for i=1 to 16

xL was XORed with P[i].

Find F(xL)

F(xL) was XORed with xR.

Interchange xL and xR.

Step 3: Interchange xL and xR.

Step 4 : xR was XORed with P[17].

Step 5: xL was XORed with P[18].

Step 6: Finally, xL and xR were combined.

### **Pseudocode of Decryption**

Step 1: The 64-bit input data were divided into two 32-bit halves (left and right): xL and xR

Step 2: for i=16 to 1

xL was XORed with P[i].

Find F(xL)

$F(xL)$  was XORed with  $xR$ .

Interchange  $xL$  and  $xR$ .

Step 3: Interchange  $xL$  and  $xR$ .

Step 4:  $xR$  was XORed with  $P[1]$ .

Step 5:  $xL$  was XORed with  $P[0]$ .

Step 6: Finally,  $xL$  and  $xR$  were combined.

As Blowfish Algorithm is a 16 round Feistel structure, it will be iterating a simple encryption function 16 times [169][170][171].

### Operation of Enhanced Blowfish Algorithm

- The SBox employed values obtained from the randomness digits of  $\pi$  [104], where the function split the encoded values (hexadecimal) of the 32-bit input into 4 bytes, which were applied as inputs to access the S-Boxes. Next, the outputs were summed with two XORs ( $SBox1 \text{ XOR } SBox2$  and  $SBox3 \text{ XOR } SBox4$ ) to produce the final 32-bit output. An example of the operations that altered the S-boxes for enhanced Blowfish algorithm are given below:

Given 32 bits as  $F(xL) = 0x0426cc66$ ,

The 8 bits were divided into 4, and later, randomized with the extra bits to generate 32 bits by the S-Box function, as depicted in the following:

byte1 = 0x04,  $S1(\text{byte1}) = 0xb8e1afed$

byte2 = 0x26,  $S2(\text{byte2}) = 0x8e7d44ec$

byte3 = 0xcc,  $S3(\text{byte3}) = 0x1ab93d1d$

byte4 = 0x66,  $S4(\text{byte4}) = 0x5121ce64$

Next, the modified S-Boxes were applied in the enhanced Blowfish algorithm:

$$F(xL) = (S_{1,a} \text{ XOR } S_{2,b} \bmod 2^{32}) + (S_{3,c} \text{ XOR } S_{4,d} \bmod 2^{32}).$$

$$0xb8e1afed \wedge 0x8e7d44ec + (0x1ab93d1d \wedge 0x5121ce64)$$

$$= 0xaf099828 \quad \text{Where,}$$

$$\wedge = \text{XOR}$$

$$+ = \text{Additional}$$

#### 4.1.2 S-BOX DESIGN AND ENHANCEMENT:

The most intense areas of studies pertaining to Blowfish algorithm are associated to S-box design [98], whereby any change that takes place to the input vector in an S-box will result in random-looking alterations in the output, indicating a relationship that is nonlinear and challenging to approximate with linear functions[148].

The standard Blowfish algorithm employed 4 S-boxes, where each contained 8-bit input and gave 32-bit output (8 X 32) elements; equivalent to 256 fractional parts (hexadecimal representation) of  $\pi$ . Meanwhile, the  $n \times m$  S-box had  $n$  input bits and  $m$  output bits, larger S-boxes, by and large, more resistant to both differential and linear cryptanalyses (larger dimension  $n$ , larger the lookup table).

The alteration made upon the enhanced Blowfish supported the parallel evaluation of two XORed operations; ( $S1, a \text{ XOR } S2, b \text{ mod } 2^{32}$ ) and ( $S3, c \text{ XOR } S4, d \text{ mod } 2^{32}$ ). The parallel XORed operation reduced the time of two XORed from  $S1, S2, S3$ , and  $S4$  that can be run simultaneously, in comparison to the standard Blowfish algorithm that was performed separately/sequential.

The enhanced Blowfish algorithm applied the existing size and dimension of S-Boxes, therefore, suggesting changes in operation structure for S-boxes from the standard Blowfish algorithm to not compromise or violate any security matrix as the permutation and substitution-generated cipher text to obtain invulnerable, but measurement on the security was performed in the next phase.

The use of 2 parallel XORed and 1 ADDED in the enhanced Blowfish algorithm failed to alter/abrupt the role of substitution/permutation in the cipher text, but the confusion of producing a cipher text is lower when compared to standard Blowfish algorithm[150].

This alteration and modification have improved the standard Blowfish algorithm to some extents. The results of all the tests conducted also lead to a common conclusion that the performance features of the enhanced Blowfish algorithm with different cases makes it perform better and maintain the security as good as standard Blowfish algorithm.

### 4.1.3 PERFORMANCE METRICS

As depicted in [97], the experiments that evaluated the performances of Blowfish algorithm and enhanced Blowfish algorithm had 8 different packet sizes of audio, video, and text data types, along with several performance metrics [104], such as those elaborated in the following:

### 4.1.4 EXECUTION TIME (ENCRYPTION AND DECRYPTION)

One of the many performance metrics is defined as the amount of time required for a plaintext message to be converted into cipher text (Encryption), and vice versa (Decryption). Hence, the average time, as given in equation (8), was calculated for both standard and enhanced Blowfish algorithms based on the gathered data.

The formula applied to calculate the average rate is as follows:

$$AvgTime = \frac{1}{Nb} \sum_{i=1}^{Nb} \frac{Mi}{ti} (Kb/s) \quad (8)$$

Where

AvgTime = Average Data Rate (Kb/s)

Nb = Number of Messages

Mi=Message Size (Kb)

Ti=Time taken to Encrypt Message Mi

Being a significant aspect, the encryption execution time had been considered as a factor that assessed the strength or the weakness for encryption algorithms. The speed measure was also related to the amount of time for encryption/decryption that employed varied parameters, such as key length and data length.

### 4.1.5 THROUGHPUT

The throughput of an algorithm encryption scheme was calculated by dividing the size of the plaintext in Megabytes (MBytes) encrypted with the total of encryption time (Equation 9):

$$Throughput = \frac{Tp}{Et} \quad (9)$$



Where

T<sub>p</sub> = Total Plain Text in Megabytes

E<sub>t</sub> = Encryption Time

Higher throughput value indicates more efficiency of encrypting any input with an encryption algorithm. Thus, it is important to calculate the throughput for the encryption as it can present the performance of the algorithm [172].

#### 4.1.6 CPU PROCESS TIME AND MEMORY USAGE

The experiment was conducted in Microsoft Visual Studio 2010 C++, MathWorks MATLAB R2016a simulation by using a personal computer (PC) with i7-2600K 3.40GHz CPU and 12.0 GB RAM that ran on 64-bit Windows 7.

#### 4.1.7 CRYPTANALYSIS/ SECURITY MEASUREMENT

This reflects the process of recovering a plaintext or a key from a particular cipher text by an unauthorized recipient. In fact, security is the most important factor in evaluating cryptographic algorithms, when compared to performance assessment [170], where the features consist of security, such as correlation coefficient resistance and avalanche effect of the algorithm from cryptanalysis, randomness of algorithm output, and relative security when compared to other candidates [173].

#### 4.1.8 AVALANCHE EFFECT

A desirable property of any encryption algorithm would be considered by the significant changes of a cipher text (output) if the plaintext or the key (input) is modified. Hence, when a fixed key and a small change in the plaintext lead to a large change in the cipher text, a block cipher could satisfy the text avalanche effect [170]. In fact, changes to one bit in either the plaintext or the key could generate a significant change to the bits in the cipher text. Mathematically, this is defined as in equation (10)

Mathematical model:

$$\forall(x,y)/H(x,y)=1, average(H(F(x)))=(n/2) \quad (10)$$

Where

$F$  = Avalanche effect

$H$  = Hamming distance between the output of a random input vector

$N/2$  = Average or generated by randomly flipping one of the bits

Where  $F$  reflects the Avalanche effect when the Hamming distance between the outputs of a random input vector and one generated by randomly flipping one of its bits should be, on average,  $n/2$  or 0.5.

#### **4.1.9 CORRELATION COEFFICIENT**

The values that are within the acceptable range for correlation coefficient [174] are given below:

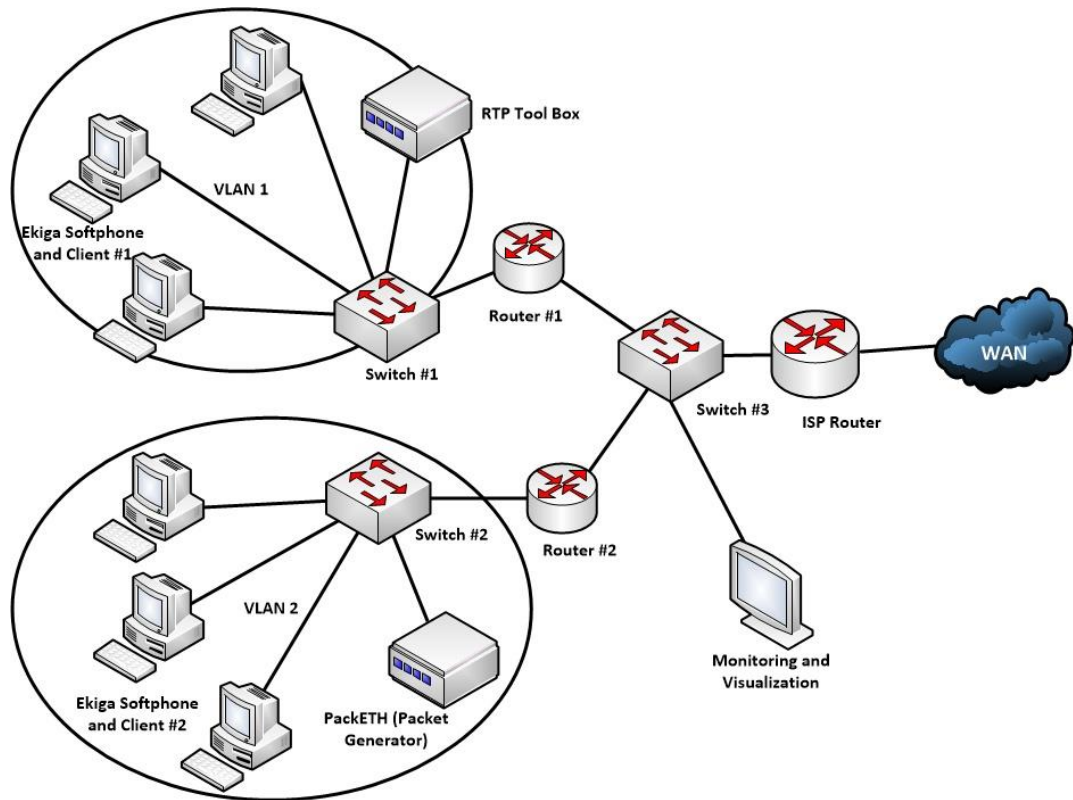
- Non-linear relationship is represented by 0
- Perfect positive linear relationship is represented by +1
- Perfect negative linear relationship is represented by -1
- Weak positive (negative) indications of linear relationship via unstable linear rule have values between 0 and 0.3 (0 and -0.3)
- Indications of a moderate positive (negative) linear relationship have values between 0.3 and 0.7 (-0.3 and -0.7)
- Indications of a strong positive (negative) linear relationship have values between 0.7 and 1.0 (-0.7 and -1.0)

#### **4.2 SECOND STAGE : NETWORK DESIGN AND TESTBED :**

It had been proven that both encryption and decryption processes have contributed to several issues related to voice quality in VoIP services, whereby the delay generated could be perceived by the users, thus giving a negative impact upon the quality of voice communication [135]. Some common instances that have affected VoIP services are packet loss, packet duplication, delay, jitter, as well as out-of-order and incomplete

packets. Therefore, in order to investigate the impacts of both the standard and the enhanced Blowfish algorithms upon the quality of VoIP services, the network testbed displayed in Figure 28 was configured by using OpenSwan VPN [175].

Furthermore, based on some studies [135] [87], several different scenarios were created in order to evaluate the impact of encryption upon VoIP calls. As portrayed in Figure 28, the network testbed was designed by using a few personal computers (PC) configured with open source software programs, internet connection from the internet service provider (ISP) connected, as well as being routed from main routers #3, #1, and #2, which represented varying virtual local area network (VLAN) that divided the network into 2 segments; clients and RTP tool box, in which voice calls are made and established by using the installed Ekiga softphone and RTP tool box for each LAN.



**Figure 28: Network testbed**

Moreover, additional software programs, such as Wireshark Analyzer [176], had been employed to capture network traffic data, as well as to perform offline packet analysis. Besides, the graphical user interface (GUI) of Wireshark could be accessed via Wireshark Virtual, which can also be used as verification and validation for IPsec encryption. On

top of that, routers #1 and #2 were configured with both standard and enhanced Blowfish encryption algorithms using the OpenSwan VPN software program. As the voice call was set up and established between client #1 and client #2, the VoIP traffic competed with the background UDP traffic sent by a pair of PackETH [156] traffic generator within each LAN. In fact, the application of PackETH can generate any possible packet and sequence of packets on the Ethernet link, a GUI in the latest version of PackETH 1.8.1, so as to provide a simple, flexible, powerful, as well as to support numerous adjustments of parameters employed in this experiment. Table 9 shows the experimental setup for varying voice codecs without any encryption or encryption, where the aspects of delay, jitter, and packet loss were recorded based on varied voice codecs tested under 1Mbps, 10Mbps and 100Mbps bandwidth allocated.

**Table 9: Network testbed experiment with different voice codec**

Experiment Number	Bandwidth	Security Imposed	Voice Codecs
1	1 Mbps	No encryption	G.711
	10 Mbps		G729
	100 Mbps		AMR-NB
2	1 Mbps	Standard Blowfish	G.711
	10 Mbps	Encryption	G729
	100 Mbps	Algorithm	AMR-NB
3	1 Mbps	Enhanced Blowfish	G.711
	10 Mbps	Encryption	G729
	100 Mbps	Algorithm	AMR-NB

Network bandwidth controlled by router #1 was configured for bandwidth utilization, where the bandwidth in this network testbed was controlled to assess the impact of network bandwidth upon voice quality during VoIP implementation with standard blowfish encryption, enhanced blowfish encryption, and without any encryption algorithm. Besides, the variables of 50mbps, 100mbps, 150mbps, and 200mbps for UDP background traffic were increased in order to test the impact of background traffic upon packet loss ratio for both VoIP networks and a variety of voice codecs. Table 10 displays the experimental setup and the parameter for varying sizes of network background traffic with and without encryption for both standard and enhanced Blowfish encryption algorithms.

**Table 10: Network testbed experiment with different background traffic**

<b>Experiment Number</b>	<b>Background traffic</b>	<b>Security Imposed</b>	<b>Bandwidth Allocation</b>
4	50 Mbps	No encryption	100 Mbps
		Standard Blowfish encryption Algorithm	
		Enhance Blowfish encryption Algorithm	
5	100 Mbps	No encryption	100 Mbps
		Standard Blowfish encryption Algorithm	
		Enhance Blowfish encryption Algorithm	
6	150 Mbps	No encryption	100 Mbps
		Standard Blowfish encryption Algorithm	
		Enhance Blowfish encryption Algorithm	
7	200 Mbps	No encryption	100 Mbps
		Standard Blowfish encryption Algorithm	
		Enhance Blowfish encryption Algorithm	

### 4.3 THIRD STAGE : VOIP CLIENTS AND MONITORING TOOLS

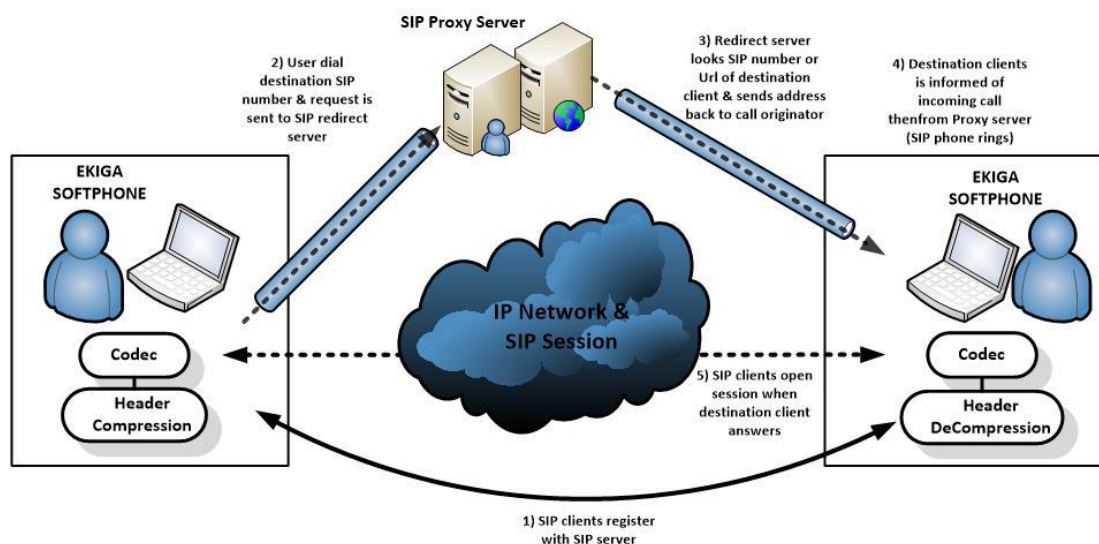
In order to simulate both VoIP services and call setup; the Ekiga softphone, a selected Session Initiation Protocol (SIP) that works on personal computer, was installed and

configured in client #1 and client #2 with varied and optional voice codecs, such as G.711, G.7.29, and AMR-NB. Both clients were registered to Ekiga server where registration, authentication, redirect, and SIP proxy server were provided. Figure 29 illustrates the call setup process for Ekiga softphone SIP communication.

### 4.3.1 SIP CALL SETUP

The VoIP call established in this testbed had been conducted in these stages:

- 1) Both clients were registered with SIP provider/SIP Server, where user name, password, and VoIP softphone SIP server information had been obtained.
- 2) A specific SIP number (Client#2) was dialed by Client#1 and request was sent to SIP redirect server.
- 3) The redirect server looked into the requested SIP number or the URL of destination client and resent the address to the call originator.
- 4) Destination client was informed about the incoming call from the SIP proxy server – Ekiga SIP phone began ringing.
- 5) Both clients communicated after the destination client answered the call, where the open session for both SIP clients had been established.

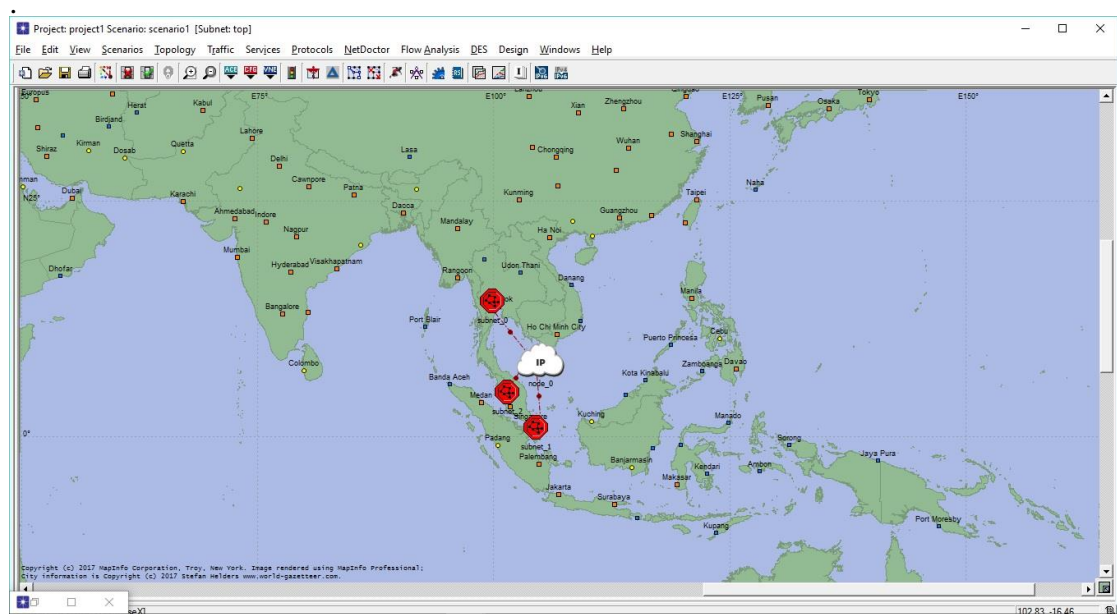


**Figure 29: Ekiga SIP based VoIP call process [183]**

#### 4.4 FOURTH STAGE: OPNET, CODECS AND NETWORK PERFORMANCE.

##### Network Design & Implementation

The simulation that was used in the OPNET network simulator refers to a network structure that was modelled based on a specific network design and several parameters. Figure 30 portrays the simulation of network environment by using some network components that communicated within a company network in Malaysia, whereby several office branches were configured and generated with VoIP communication network (links, equipment, application, etc.). In fact, various types of network setups, including network test scenarios (topology, routing protocol, resources, etc.), were tested by implementing the existing security (encryption algorithms) and the combination of varied VoIP services (protocol & codecs).



**Figure 30: OPNET network design**

The implementation of enhanced Blowfish encryption algorithm by using the OPNET simulation had been tested by performing several experimental setups with scenario parameters, as depicted in Table 11. In fact, the properties of OPNET simulator were configured with various scenarios and parameters that differed by call numbers (background traffic), voice codecs, allocated bandwidth, and component properties including VoIP security for both with and without enhanced or standard blowfish algorithms. Next, the network performance and the voice quality of VoIP were recorded and compared (delay, jitter, and packet loss). After that, the simulations based on OPNET

were compared with hardware implementation in the network testbed for comparison, validation, and verification purposes.

**Table 11: Properties of OPNET simulation scenarios**

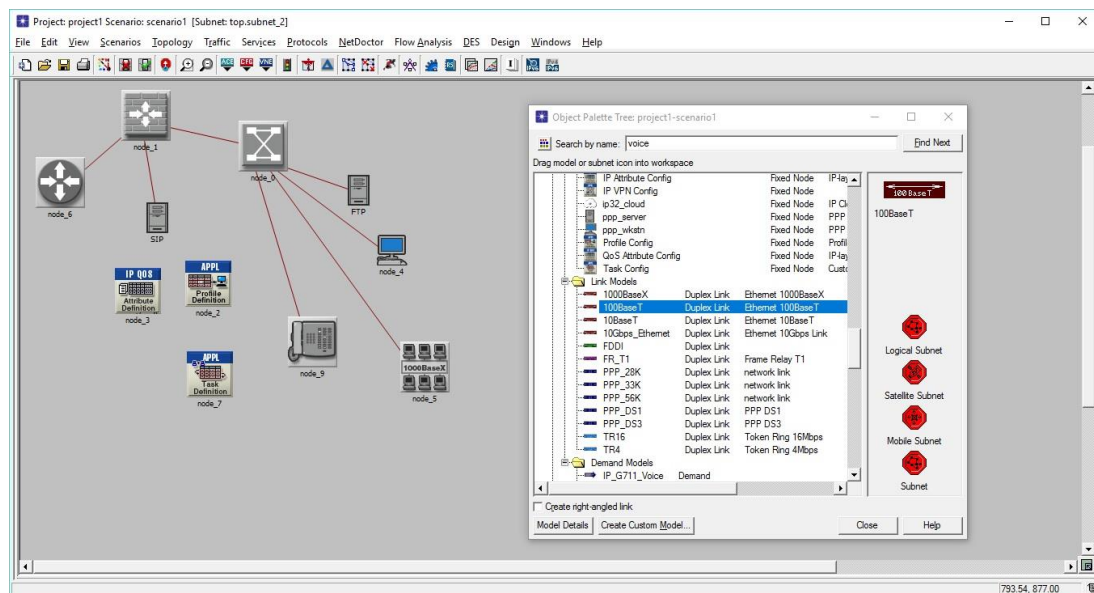
Experiment Number	No of VoIP Calls made	Codecs	Scenario Parameter	Bandwidth Allocation
1-3	10 Calls	G. 711	No encryption	1 Mbps
4-6		G.729	Standard Blowfish encryption Algorithm	
7-9		AMR-NB	Enhance Blowfish encryption Algorithm	
10-12	200 Calls	G. 711	No encryption	1Mbps
13-15		G.729	Standard Blowfish encryption Algorithm	
16-18		AMR-NB	Enhance Blowfish encryption Algorithm	
19-21	10 Calls	G. 711	No encryption	10 Mbps
22-24		G.729	Standard Blowfish encryption Algorithm	
25-27		AMR-NB	Enhance Blowfish encryption Algorithm	
28-30	200 Calls	G. 711	No encryption	10 Mbps
31-33		G.729	Standard Blowfish encryption Algorithm	
34-36		AMR-NB	Enhance Blowfish encryption Algorithm	
37-39	10 Calls	G. 711	No encryption	100 Mbps
40-42		G.729	Standard Blowfish encryption Algorithm	
43-45		AMR-NB	Enhance Blowfish encryption Algorithm	
46-48	200 Calls	G. 711	No encryption	100 Mbps
49-51		G.729	Standard Blowfish encryption Algorithm	
52-54		AMR-NB	Enhance Blowfish encryption Algorithm	

This scenario was built to boost computer processing demand. With 200 simultaneous calls, the origin and destination of the calls will encrypt and decrypt a significant amount of data. The objective is to check if, in this situation, some of the algorithms distinguish from the others and what the difference is in terms of the quality of the encrypted calls and the non-encrypted ones.



#### 4.4.1 OPNET NETWORK DESIGN.

Networks based in Kuala Lumpur, Malaysia; Narathiwat, Thailand; and Geylang, Singapore were designed and configured by using the OPNET simulator. Besides, a middle-sized enterprise that adopted the VoIP services had been involved, where each located branch was connected and communicated. These three facilities communicated by using SIP-based VoIP services, where the main call centre and the headquarters were both located at Kuala Lumpur. As the main call signalling, the SIP protocol was employed to handle call setup and call establishment. As such, the call centre in Kuala Lumpur handled the base network for SIP VoIP server, the SIP proxy server, and together with FTP server, which was all deployed in the same location to overload the network traffic with various types of network protocols and services, as illustrated in Figure 31.



**Figure 31: Simulation VoIP enterprise network**

In order to compare the VoIP codecs, 3 various types of voice codecs were selected and tested in this simulation. Besides, the implementation of the encryption algorithm served as part of the proposed VoIP security. Thus, voice codec and network topologies were tested so as to evaluate both QoS and voice quality. With that, G.711, G.729, and AMR-NB codecs were selected and tested in this simulation, in which the retrieved results were compared with prior network testbed.

Furthermore, with varied network scenarios, loads of traffics, and security implementation; the network QoS parameters (jitter, end-to-end delay, packet loss) and the voice quality (MOS) were evaluated to analyse the results obtained from the enhanced

Blowfish algorithm for varying scenarios to establish an opinion. The results provided data to evaluate the performances exerted by various codecs with the implementation of enhanced blowfish algorithm, along with various network scenarios.

Moreover, various components that had been available in the OPNET modeller were employed to design a base model with complicated network design, as well as basic and advanced node models, such as routers, switches, server, and wireless devices that allowed simulation on complex scenarios with differing link models, path models, and network protocol support.

Besides, all the three branches in this topology were connected by using the connection provided by WAN technology: Point-to-Point Digital Signal 3 (PPP\_DS4), which is also known as T4 line, as well as a bandwidth up to 274 Mbit/s (274 Mb) connected by using IP32\_Cloud. This simulation design enabled both the evaluation and the comparison of performances exerted by VoIP codecs G7.11, G7.29, and AMR-NB under varying configuration and scenarios. The next section explains the details for all the components embedded in the OPNET simulation for this study.

#### **4.4.2 APPLICATION CONFIGURATION**

A profile (Appendix C, Figure 62) was constructed by using different application definitions, whereby in this research, a few profile name was generated, such as Voice, Ftp, e-mail, Http, and database. The simulation configured was setup with different traffic loads and network usage patterns.

With these identical applications, different usage parameters were configured with various tier attributes. These defined attributes, which are unique for each profile, enabled the execution of various types of common network applications.

Besides, each tier adhered to the incoming traffic by having a specific name and port that are interconnected to a variety of nodes within the designed network.

#### **4.4.3 APPLICATION SPECIFICATION**

Both the standard and the custom network applications applied in the simulation portrayed the attributes of both applications and activities of VoIP services in this particular study (Appendix C, Figure 63). The description was applied for specific applications, such as voice, web-browsing, ftp, database, and email; all with heavy and

low traffic. Such traffic load simulated varying traffic patterns in order to evaluate the VoIP codec and the security parameter under various traffic conditions.

#### **4.4.4 VOICE ENCODER SCHEMES**

A voice encoder scheme (Appendix C, Figure 64) was set in accordance to the traffic generation required in the network. As such, PCM voice encoder schemes G7.11, G7.29, and AMR-NB (12.2K) were selected from the voice codec attribute.

#### **4.4.5 PROFILE CONFIGURATION**

The patterns of activities exerted by either a user or a group of users in relation to specific applications used over a period of time had been configured by using profile configuration. Several profiles, hence, were run on a given network topology or a single node. These profiles were executed on the node by configuring them repeatedly, concurrently or serially (Appendix C, Figure 65).

In this simulation, the profile configuration setup that was set as either concurrently or simultaneously, which portrayed the real network, behaved with the activities run by group. Hence, VoIP services, web surfing, database access, ftp file transfer, and email activities were defined in the application definition so as to simulate the scenarios of branch activities and network traffic pattern (Appendix C, Figure 66).

Moreover, the traffic pattern, together with applications and profile configuration, were named and specified in these varied object nodes, but it is essential for the traffic to adhere to the application definition.

### **4.5 FIFTH STAGE: DATA COLLECTION AND PROCESSING**

In order to measure the impact of encryption algorithms upon VoIP communication; data collection had been required for investigation, while data analysis was performed between different experimental setups. The processes of sending and receiving voice calls from one or more locations to another reflected the data collection process. As such, in the experiment using network testbed, the Wireshark captured data that were propagated from client 1, client 2, RTP Tool, and Iperf in the form of .pcap, which were later converted to XML, whereas the .wav. RTP tool was transmitted and recorded as voice files in .wav and .pcm, which had been employed to create and to generate MOS score based on E-Model/R-factor, in which the graph generated is displayed as MOScqe.

Furthermore, the OPNET simulator had the ability to generate traffic based on attributes, applications, and profile configurations, whereby the traffic generated both network traffic data and utilization from varying settings. The data processing via discrete event approach/workflow is as follows:

- a) The configuration from nodes, project, topology, and scenarios created traffic that generated statistics
- b) Simulation setup based on Discrete Event Simulation (DES)
- c) Results and reports/graphs were generated

#### **4.6 SIXTH STAGE: VALIDATION AND VERIFICATION**

The empirical process that had incorporated the implementation of OPNET simulation network settings had been validated with the network testbed/hardware implementation, as presented in Chapter 3. In fact, the varying results of the simulation model retrieved from both experiments were validated by comparing the results from both experiments. Meanwhile, the verification process concluded that the simulation model was indeed good and portrayed a fair approach of the hardware model if the variance between both results is negligible. Moreover, the varied results obtained throughout the OPNET simulation were considered as realistic because the network testbed implementation results for network delay, jitter, packet loss, and MOS were obtained by using all network scenarios, including IPSec security implementation. In fact, the verification process incorporated the iteration of both experiments to determine the aspect of consistency, as well as to review the results.

#### **4.7 SUMMARY**

The enhanced Blowfish algorithm was designed from the standard Blowfish algorithm by altering the F function. Besides, 6 phases of simulations and implementations were incorporated, where each stage had been essential to increase consistency, verification, and validation processes. Moreover, both algorithms were employed to analyze the performance exhibited by VoIP under various network situations and simulation configurations. The next chapter presents the related results, analysis, and comparison from the experimental runs.

# CHAPTER 5

## RESULTS, ANALYSIS AND COMPARISON

This particular chapter presents and discusses the results obtained from the experiments conducted, as described in Chapter 4. With that, three significant scores are presented in this chapter, which are i) the evaluation of both performance and security measurement of standard Blowfish and enhanced Blowfish algorithm, ii) the measurement of the impact from encryption upon network QoS, and iii) voice quality measurement with varied voice codecs. Besides, VoIP security encryption for both algorithms were configured and tested by using C++, MATLAB, network testbed, and OPNET simulation. As such, the results for this research are i) the performance of blowfish encryption algorithm (execution time and throughput), along with encryption security cryptanalysis, ii) the impact of encryption upon network QoS (Jitter, delay, and packet loss), as well as iii) voice quality score (MOS).

### 5.1 SIMULATION RESULTS

Both aspects of execution time and throughput from the algorithms gained in C++ reflected bandwidth requirement, where higher throughput significantly contributed to better voice quality [177]. After that, the retrieved results for voice quality in network testbed and OPNET simulation were validated and verified. Moreover, the Network QoS measurement was related to end-to-end services performance, VoIP QoS, such as packet loss, delay, and jitter, which had been examined together with varying sizes of bandwidth allocated and background traffic.

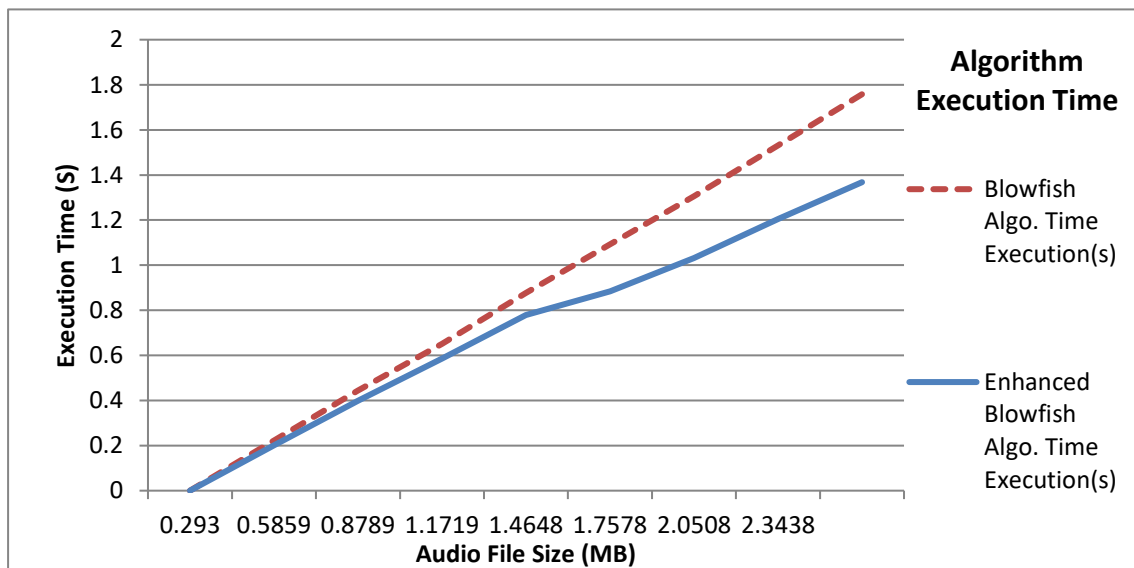
The voice quality score had been based on the objective MOS score calculated from E-Model, whereby the effects of delay, packet loss rate, and jitter upon voice qualities had been determined. On top of that, the implementation of various voice codecs contributed to the varying MOS scores. Thus, in order to determine the effect of the proposed enhanced Blowfish algorithm upon VoIP communication and voice quality, several combinations of various voice codecs, bandwidth allocation, calls setup, and background traffic were tested. As such, the results retrieved could enhance one's comprehension

concerning VoIP services, which suggests a more reliable security measurement and reasonable voice quality.

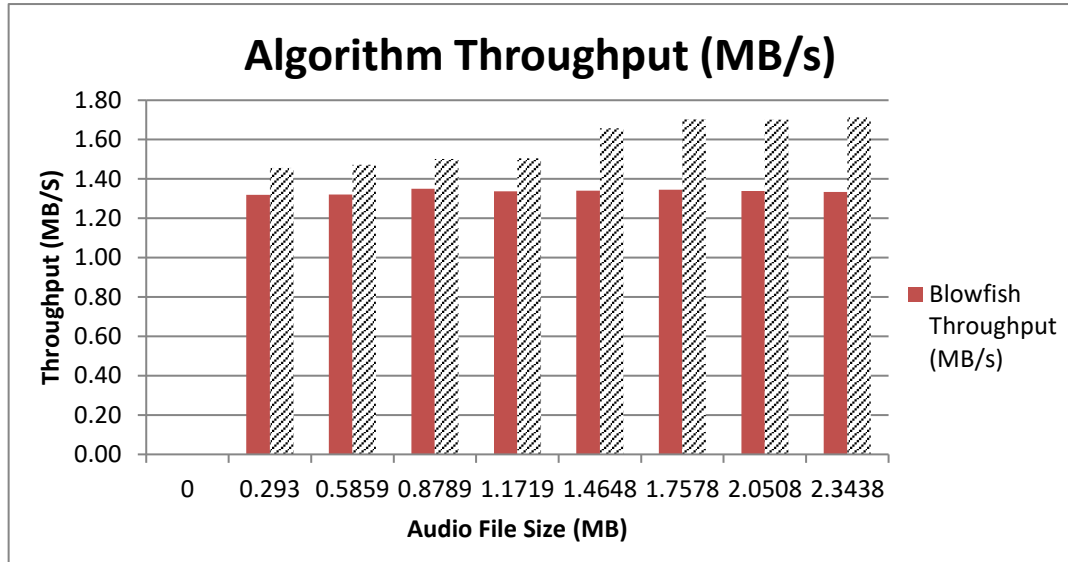
### 5.1.1 ALGORITHM PERFORMANCE (EXECUTION TIME & THROUGHPUT)

The empirical procedures employed in this research embedded 2 different encryption algorithms (Standard Blowfish and enhanced Blowfish algorithms) by employing 64-bit secret keys with data from varied packet sizes (ranging from 0.2MB to 2.5MB) and types (audio, video, and text). The varying packet sizes and data types were employed based on some variable parameters to determine the impact of data length upon speed, which contributed to different processing time.

Figures 32 and 33 illustrate the average execution time and throughput of audio for both standard and enhanced Blowfish algorithms by using various packet sizes. A comparable result to data shown below is derived from the analysis of the execution time between both algorithm, a significant reduction of execution time approximately starting from audio file size 1.4648 MB, 11% reduction of execution time. As expected, the bigger audio size, the greater the execution time achieved by using enhanced Blowfish algorithm, where at 2.3438 MB, approximately up to 22.8% reduction of execution time can be achieved by using enhance Blowfish algorithm.



**Figure 32: Execution time for standard and enhanced Blowfish algorithms in second (s) for audio packet**

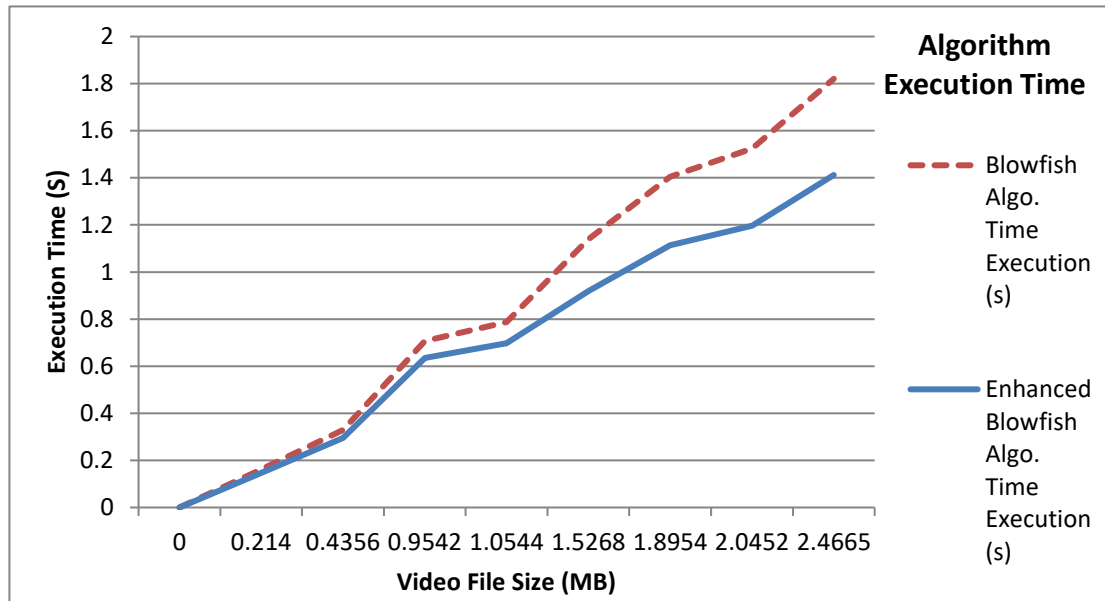


**Figure 33: Throughput (MB/s) for standard and enhanced Blowfish algorithms for audio packet**

As the throughput is the rate of production or the rate at which something is processed, the reduction of time processing by using enhanced Blowfish algorithm can produce higher throughput compared to standard Blowfish algorithm. The execution time ratio's percentage is also reflected for algorithm throughput where the higher values of throughput of enhanced Blowfish algorithm at 2.3438 MB was approximately 22.8% compared to throughput achieve by standard Blowfish algorithm.

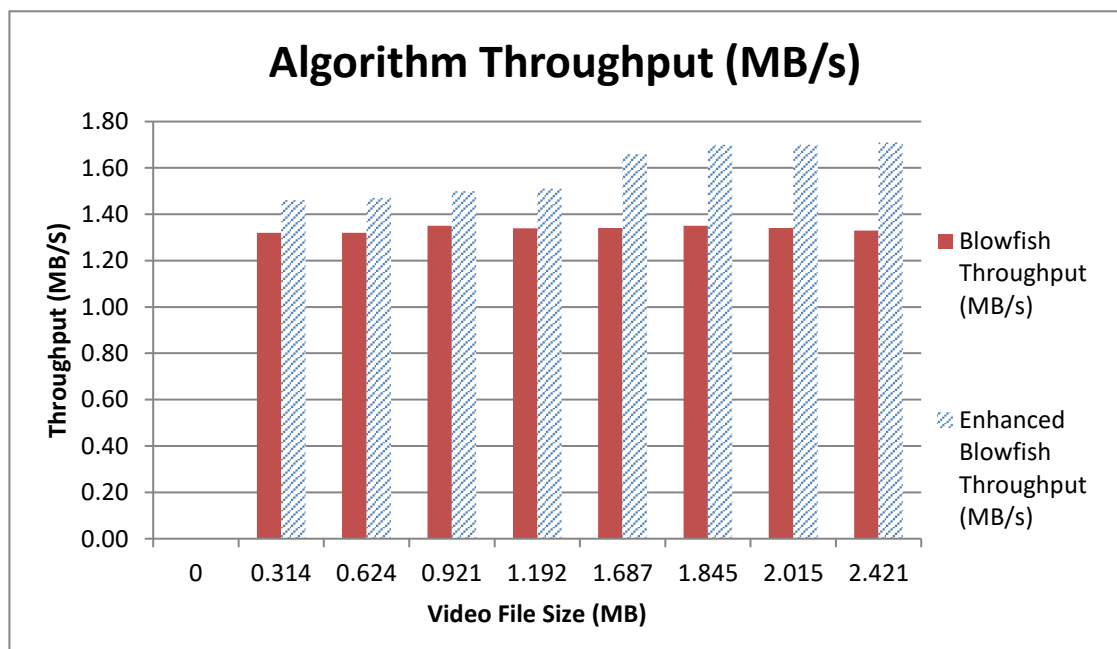
### 5.1.2 DIFFERENT PACKET TYPES (VIDEO & TEXT)

Figures 34 and 35 portray the average execution time and throughput of video for both standard and enhanced Blowfish algorithm by using various packet size. The results obtained by execution time for video data types show that starting from data size 1.52681 MB, enhanced Blowfish algorithm achieve 22 % reduction of execution time compared to standard Blowfish algorithm and its significance increase up to 23.5 % reduction of execution time for video data size 2.4565 MB.



**Figure 35: Execution time for standard and enhanced Blowfish algorithms in second (s) for video packet**

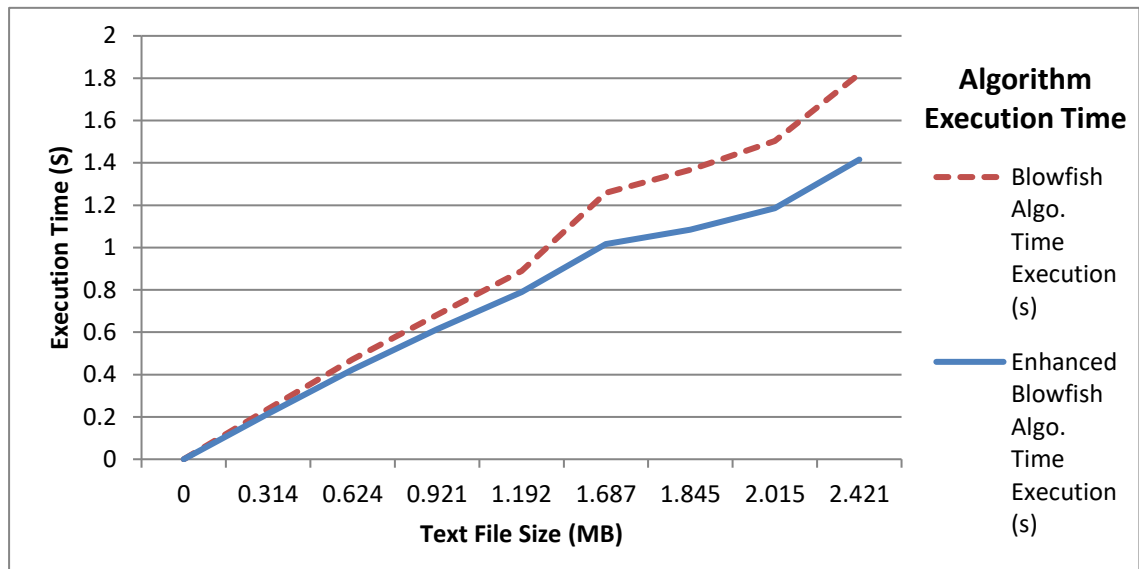
As the video size increases, as shown in Figure 35, video size 1.687 MB using enhanced Blowfish can be seen with 14.2 % higher throughput compared to standard Blowfish algorithm. While, starting from video file size 1.845 MB to 2.421 MB, an increment of throughput approximately 30.7 % has been achieved by enhanced Blowfish algorithm compared to standard Blowfish algorithm.



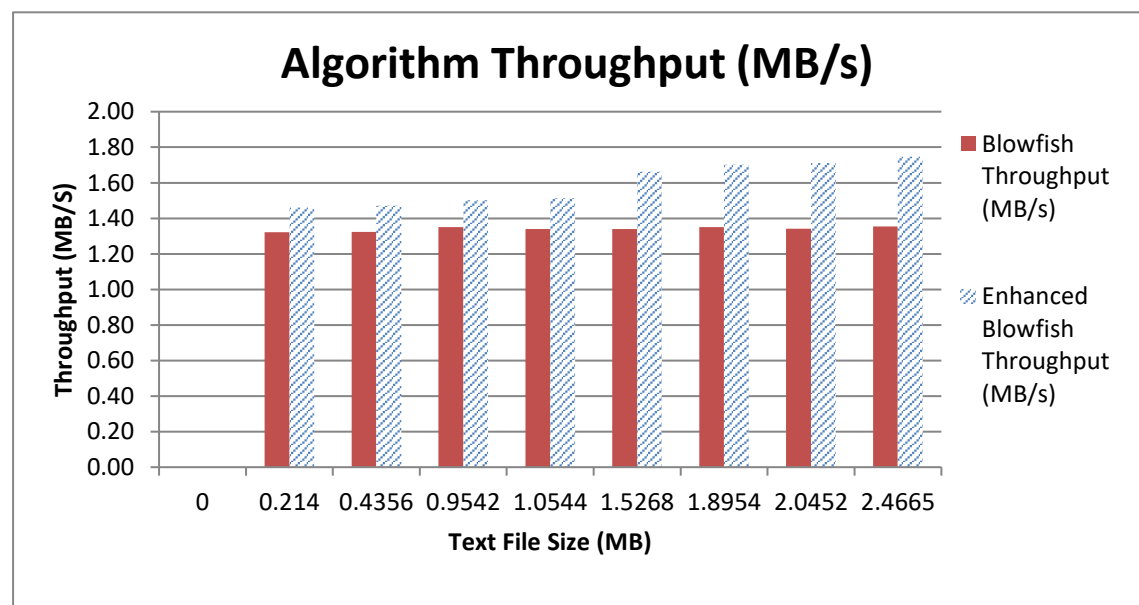
**Figure 34: Throughput (MBytes/s) for standard and enhanced Blowfish algorithms for video packet**



Figures 36 and 37 present the average execution time and throughput of the document (text) for both standard and enhanced Blowfish algorithms by using various packet sizes. Results show almost similar and significant as tested conducted by using video and audio files which indicated reduction up to 29% execution time starting from 1.685 MB to 2.421 MB data and increment of throughput up to 30% starting between data range 1.526 MB to 2.466 MB.



**Figure 36: Execution time for standard and enhanced Blowfish algorithms in second (s) for text packet**



**Figure 37: Throughput (MBytes/s) for standard and enhanced Blowfish algorithms for text data**

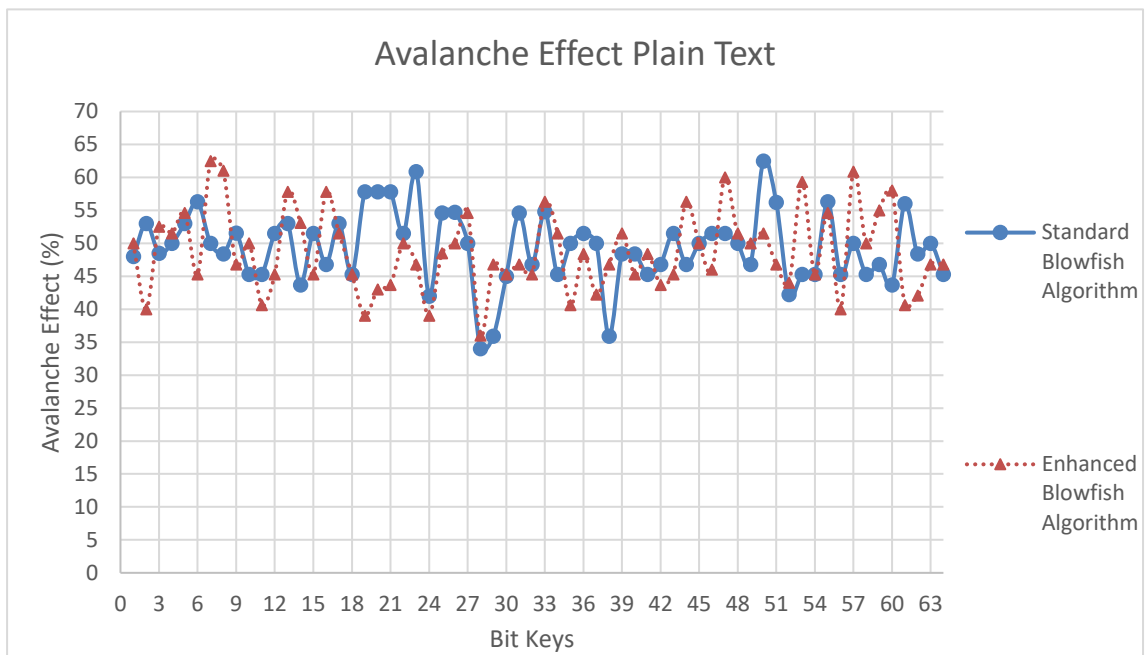
By using 8 various sizes and types of packets (text, audio, and video) simulated using C++ programming, both standard and enhanced Blowfish algorithms were tested. As a result, Figures 32, 34, and 36 display that the execution time for the enhanced Blowfish algorithm was lower than that of the standard Blowfish algorithm, thus significantly contributing to higher throughput, as portrayed in Figures 33, 35, and 37. Additionally, the experimental results also concluded that by implementing various types of input data (text, audio and video), the outcomes of throughput for both standard and enhanced Blowfish algorithms remained almost similar regardless of the types of data.

### **5.1.3 CRYPTANALYSIS RESULTS/ SECURITY ANALYSIS**

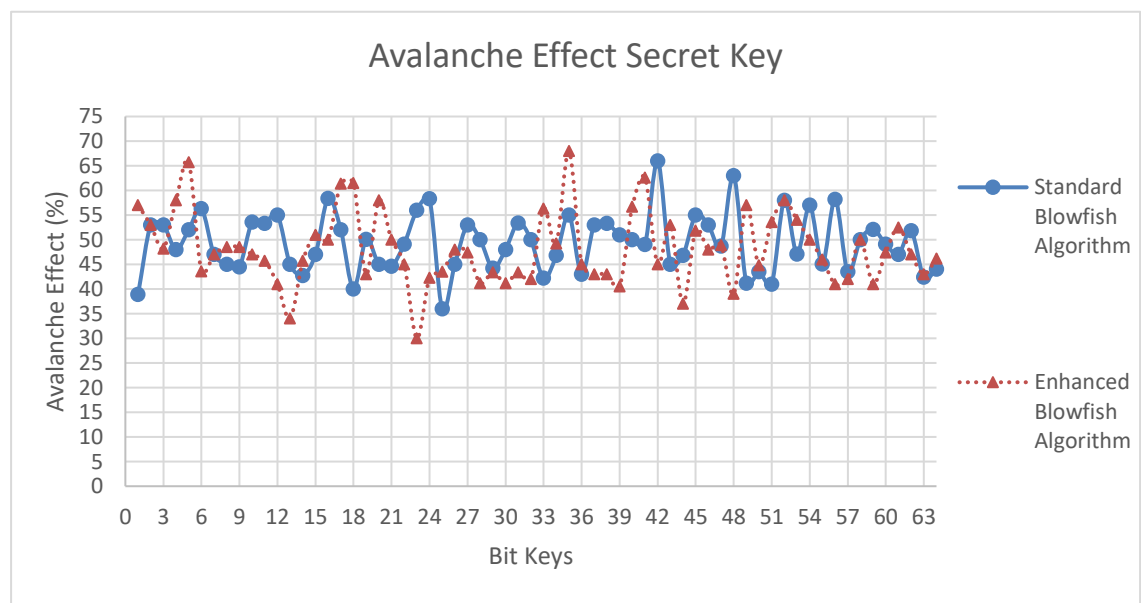
As described in the cryptanalysis and security analysis in section 4.1.7, two security analyses that applied avalanche effect and correlation coefficient had been employed to evaluate the security aspect of both standard and enhanced Blowfish algorithms. As such, the implementation of both algorithms had been performed by using C++, whereas MATLAB programming had been applied for the implementation of avalanche effect and correlation coefficient.

### **5.1.4 AVALANCHE EFFECT**

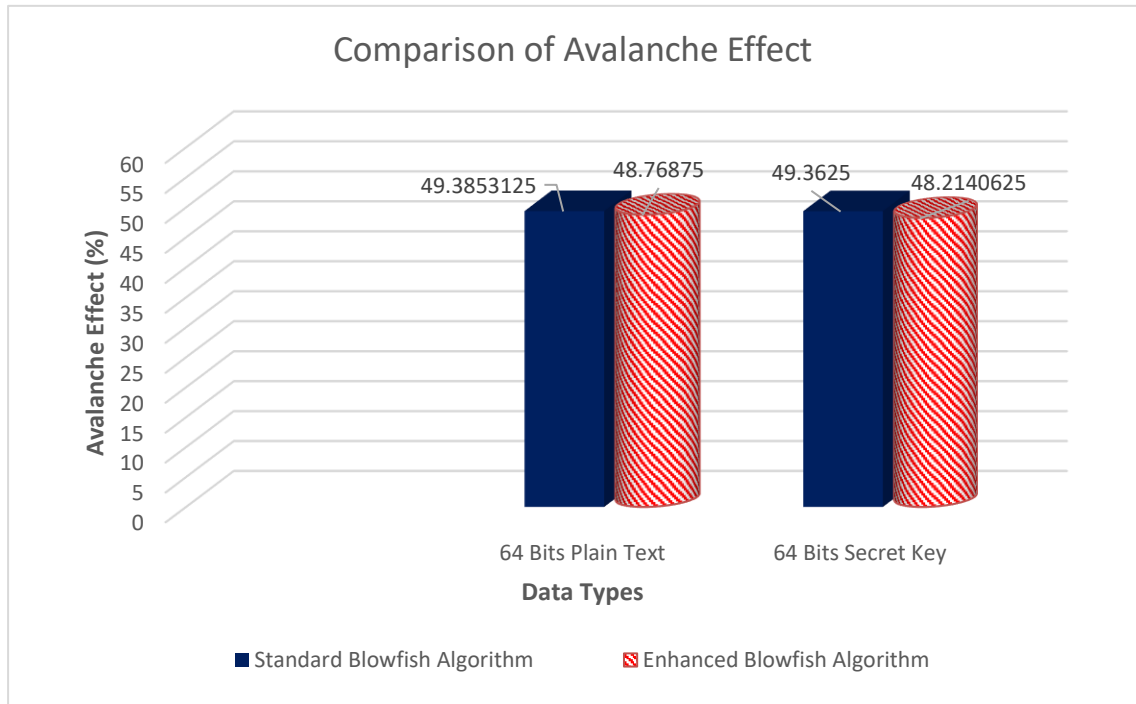
The Avalanche effect analysis looked into cryptanalysis and resistance of encryption algorithm against potential attacker, as depicted in Chapter 4. In fact, the avalanche effect test displayed that the enhanced Blowfish algorithm also did possess exceptional diffusion characteristics, which was near similar to that of the standard Blowfish algorithm. Figure 38 illustrates the avalanche effect upon cipher text for both standard and enhanced Blowfish algorithms with 64-bit secret keys. Next, Figure 39 presents the avalanche effect on cipher text for both standard and enhanced Blowfish algorithms with 64-bit plain text. Meanwhile, Figure 40 shows the comparison of avalanche effect between the standard and enhanced Blowfish algorithms with 64-bit cipher text. As a result from the comparison of avalanche effect for both standard and enhanced Blowfish algorithm, approximately 1% decreased which is slightly different from standard blowfish algorithm, yet it exhibits strong avalanche effect and it had been noted that the enhanced Blowfish algorithm offered higher encryption quality with minimal memory requirement and computational time, besides providing adequate resistance against attack.



**Figure 38: Avalanche effect percentage with 64-bit key changes on plain text**



**Figure 39: Avalanche effect percentage with 64-bit key changes on secret key**



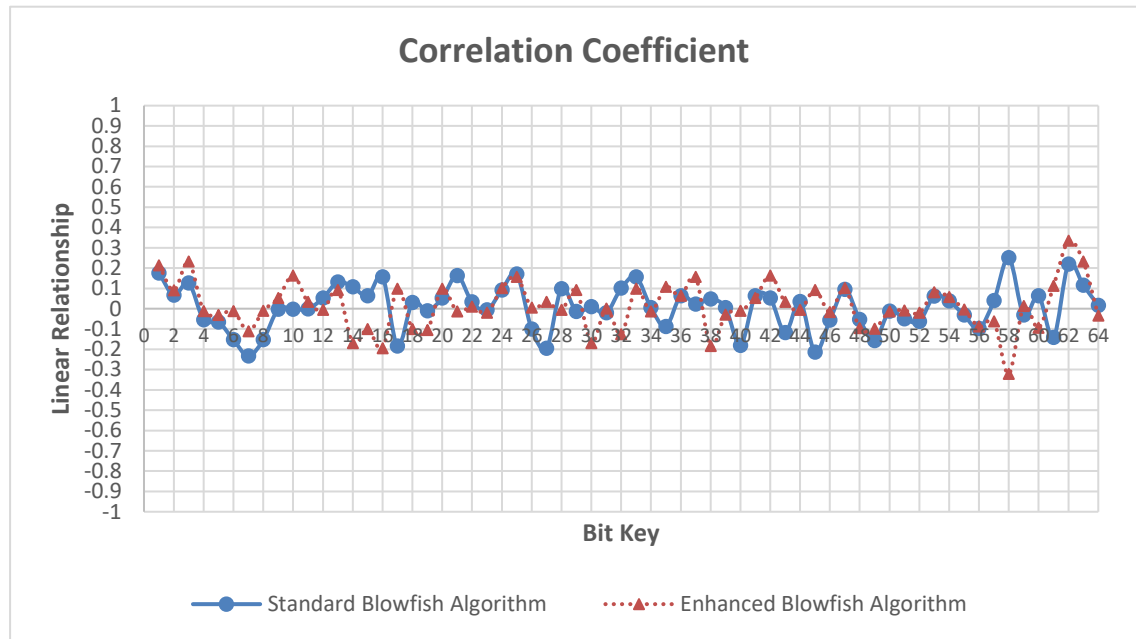
**Figure 40: Comparison of avalanche effect for both standard and enhanced Blowfish algorithms**

### 5.1.5 CORRELATION COEFFICIENT

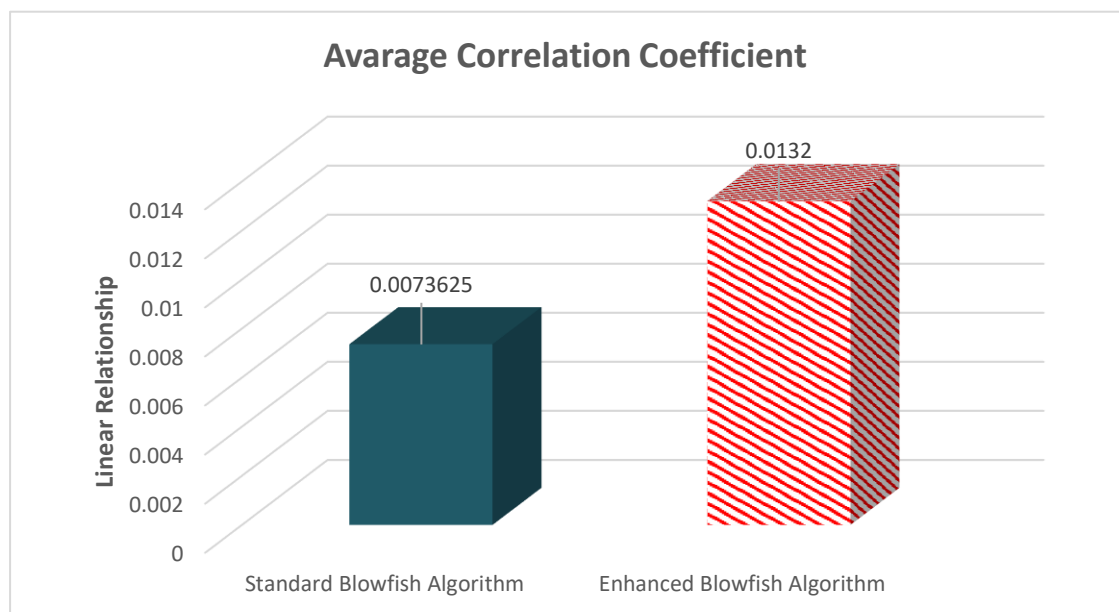
As an important aspect of security for block ciphers, correlation coefficient deals with dependency between individual output and input bits; a measurement of two variables that affect each other or dependence of one on the other. Furthermore, correlation coefficient can measure dependency between plaintext and cipher text, whereby the confusion effect of the block cipher could be identified via correlation values. The correlation values were represented by numbers between (-1) and (1), while the degree of linear relationship between two variables was reflected by (-1) for decreasing linear relationship, (1) for increasing linear relationship, and 0 for independent variables.

Figure 41 exhibits the correlation coefficient for both standard and enhanced Blowfish algorithms, where the values of correlation coefficient closer to zero indicate a non-linear relationship between input and output bit keys. As a result, the standard Blowfish algorithm recorded the lowest value at -0.2333, while -0.3218 for the enhanced Blowfish algorithm -0.3218. Nevertheless, higher values that indicate weak positive linear relationships were obtained for standard Blowfish algorithm at 0.2505, whereas 0.3336 for enhanced Blowfish algorithm. Moreover, the average correlation coefficient values,

as presented in Figure 42, for standard Blowfish algorithm was 0.0007, while 0.0132 for enhanced Blowfish algorithm, thus pointing out that both algorithms had weak positive linear relationships. Furthermore, the enhanced Blowfish portrayed a good non-linear correlation between plaintext and cipher text, similar to that of the standard Blowfish encryption algorithm.



**Figure 41: Correlation coefficient for both standard and enhanced Blowfish algorithms with 64 bits key.**



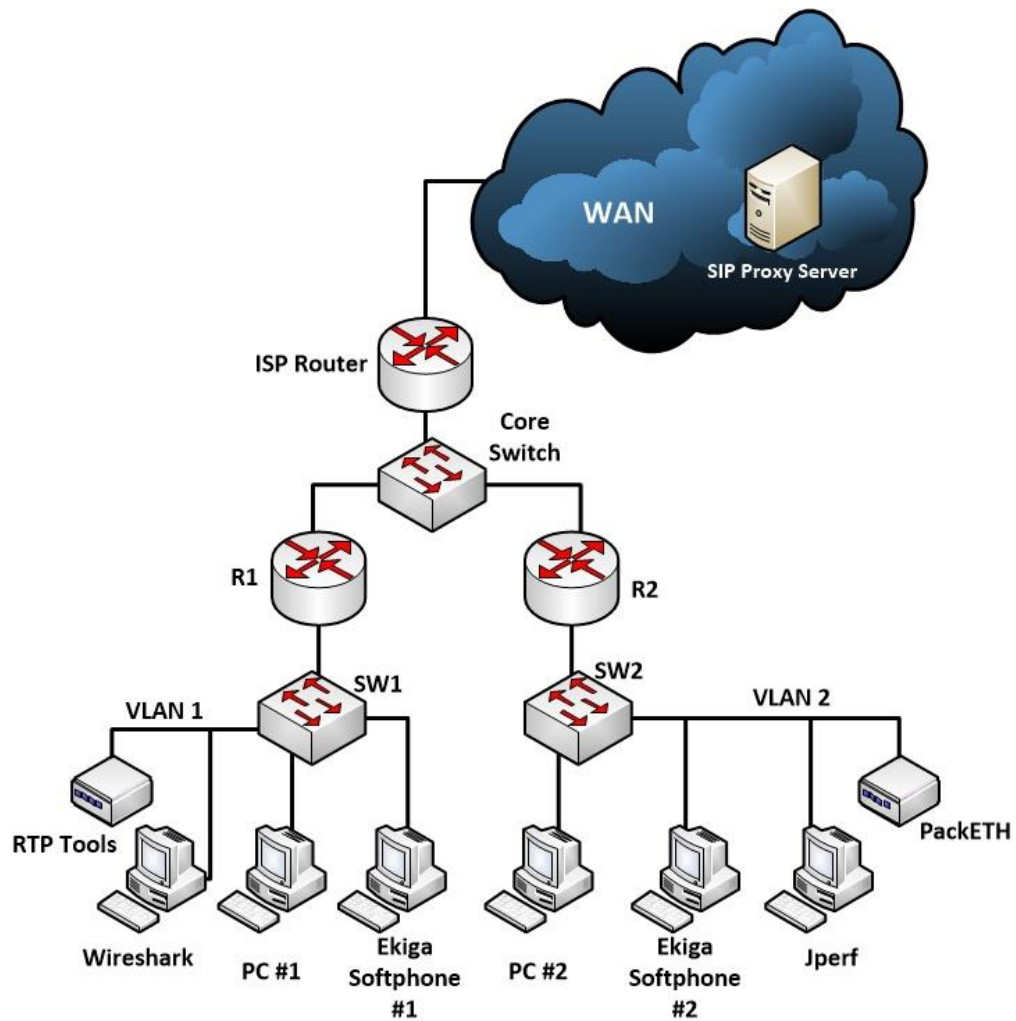
**Figure 42: Average of correlation coefficient 64-bit key cipher text for both standard and enhanced Blowfish algorithm.**

The alteration made to F function in the standard Blowfish algorithm resulted in enhanced encryption quality for both execution time and throughput. Nonetheless, such modification displayed a negligible effect upon several security features, where a slight variance was noted between avalanche effect and correlation coefficient, in which the enhanced Blowfish algorithm generated lower avalanche effect and higher correlation coefficient, in comparison to those projected by the standard Blowfish algorithm. As such, one can conclude that the enhanced Blowfish could still provide acceptable resistance against security penetration and attacks.

## **5.2 NETWORK TESTBED & OPNET SIMULATIONS RESULTS**

### **5.2.1 NETWORK QoS PARAMETERS**

The experimental setup that incorporated various bandwidths, along with the usage of traffic based on User Datagram Protocol (UDP) and Transmission control protocol (TCP), was generated by using PackETH and RTP toolbox. In fact, both PackETH and RTP toolbox can create and generate any possible packet, including voice traffic and a sequence of packets on Ethernet link, graphical user interface (GUI) in the latest version of PackETH 1.8.1 and RTP toolbox, so as to offer a simple, flexible, and powerful tool in order to support many alterations in parameters for the experiment. In addition, variables 50Mbps, 100Mbps, 150Mbps, and 200Mbps of UDP background traffic were increased so as to test the effect of background traffic upon packet loss ratio in VoIP networks. Besides, Jperf [178] and Wireshark were also employed as the main measurement tools for the testbed. In this testbed, as presented in Figure 43, Jperf and Wireshark were installed and set for both data collection and processing. During the call set up made between these two nodes (Clients #1 and 2#), Jperf and Wireshark collected and processed the sent/received data from each client that produced jitter values, delay, packet loss rates, and MOS score. Furthermore, the use of varied voice codecs and VoIP security in this experiment measured the effects of voice codecs and encryption algorithms variation during data transmission in maintaining end-to-end network security, as well as network QoS performance [179].



**Figure 43: VoIP calls monitoring and network testbed**

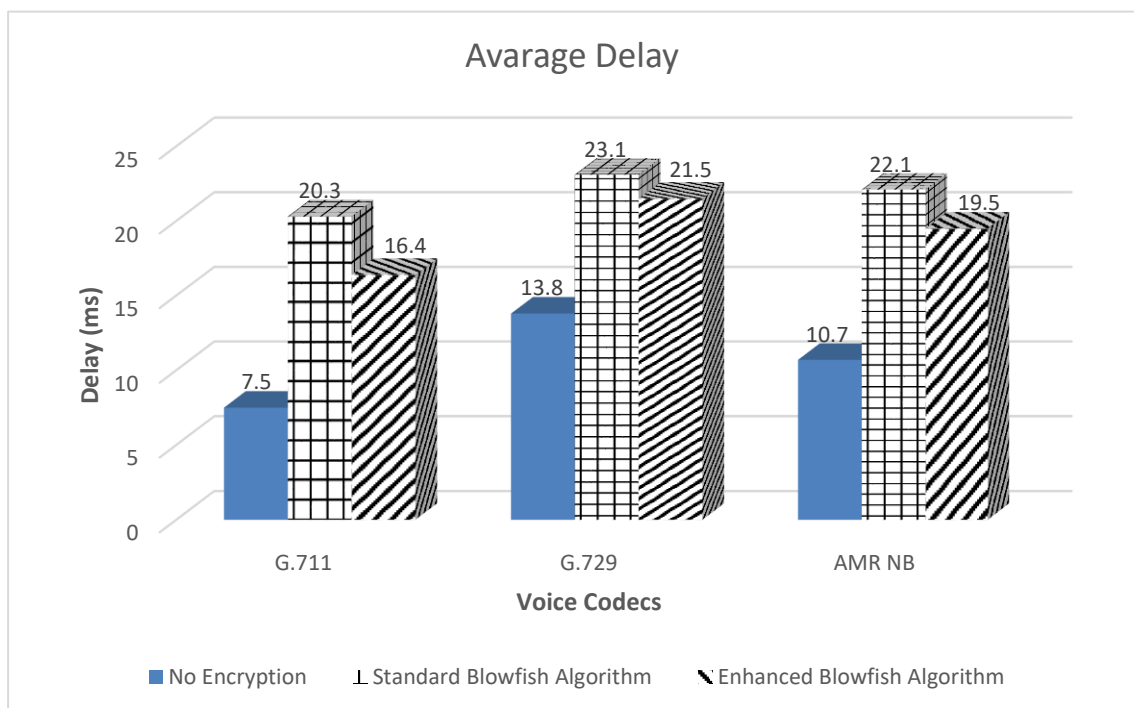
Figures 44-48 illustrate three different network QoS that were recorded, where simulation was created by 1 call within 10 minutes of call duration under VoIP open system (without IPsec), VoIP IPsec with standard Blowfish algorithm, and VoIP IPsec with enhanced Blowfish algorithm. By using the available 100mbps bandwidth link, a VoIP communication with a variety of voice codecs had been transmitted. Moreover, network average delay, jitter, and packet loss were recorded with the influence of various UDP background traffic sizes generated by PackETH; packet traffic generator starting from 0Mbps, 50Mbps, 100Mbps, 150Mbps, and 200Mbps traffic overload. Besides, with the use of softphone and RTP tool box, VoIP data transmission had been produced during VoIP call setup, whereby various packet traffics were employed to perform and to create overloaded links, which was later compared with VoIP performance under heavy traffic and extreme conditions.

### 5.2.2 AVERAGE DELAY

The results of average delay are tabulated in Table 12 and Figure 44. The findings portrayed that G.711 performed better among the three voice codecs, whereby the encryption algorithms increased the average delay on voice quality, whereas G.729 appeared to be the worst. Moreover, the implementation of encryption algorithm had managed to increase the average delay for all voice codecs, where the enhanced Blowfish encryption algorithm exhibited better results, when compared to the standard Blowfish encryption algorithm.

**Table 12: Comparison of average delay from G.711, G.729, and AMR-NB codecs**

	G.711	G.729	AMR-NB
No Security	7.5 ms	13.8 ms	10.7 ms
Standard Blowfish Algorithm	20.3 ms	23.1 ms	22.1 ms
Enhanced Blowfish Algorithm	16.4 ms	21.5 ms	19.5 ms



**Figure 44: Comparison of average delay**

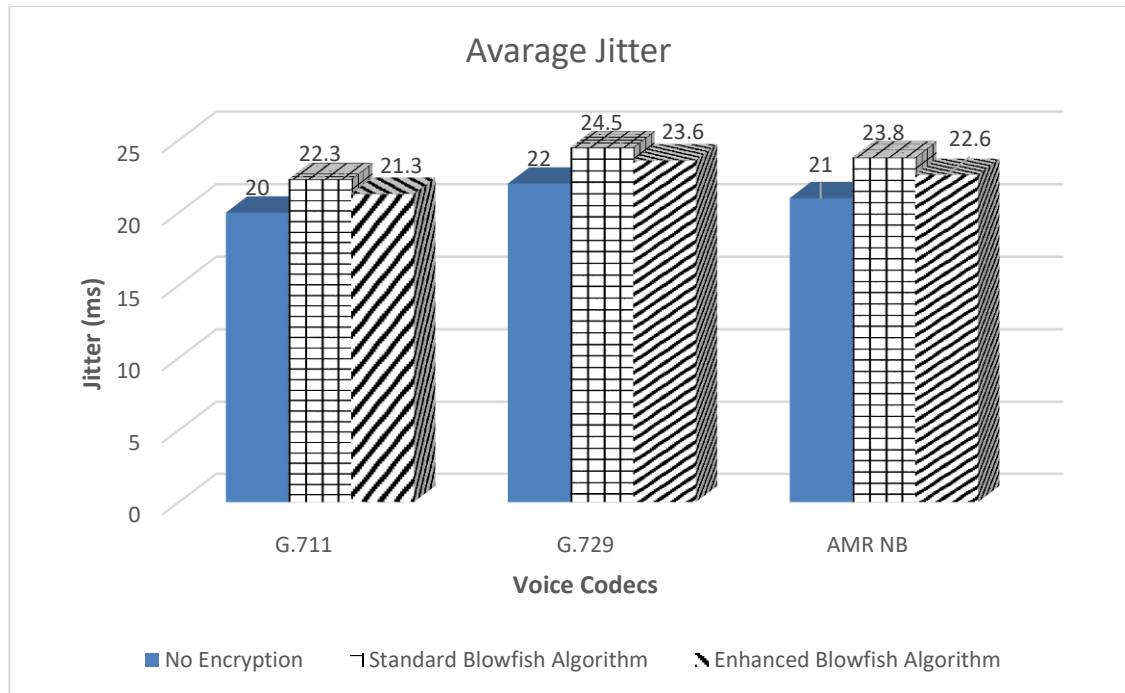


### 5.2.3 AVERAGE JITTER

Table 13 shows the average jitter for voice codecs without security and encryption algorithm imposed. As a result, similar to those for average delay, almost the same average had been obtained for all voice codecs, except for G.711 that displayed the lowest average jitter and G.729 that recorded the highest for all the scenarios tested. In addition, the results given in Figure 45 also pointed out that the AMR-NB codecs remained at an intermediate average jitter between all the three codecs. The results also provided rather significant average delay, where a higher average jitter was recorded for G.729 voice codec, while the implementation of standard Blowfish algorithm displayed higher values, but the enhanced Blowfish encryption algorithm deferred slightly from VoIP communication without any encryption algorithm.

**Table 13: Comparison of average jitter from G.711, G.729, and AMR-NB codecs**

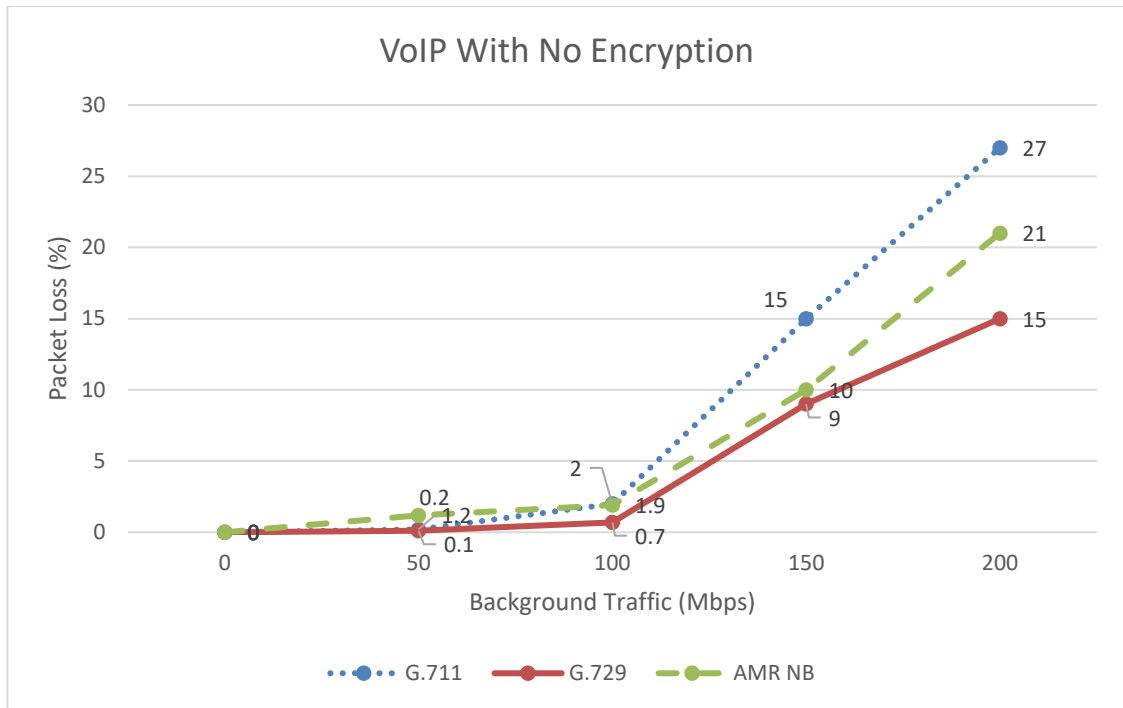
	G.711	G.729	AMR-NB
No encryption	20	22	21
Standard Blowfish Algorithm	22.3	24.5	23.8
Enhanced Blowfish Algorithm	21.3	23.6	22.6



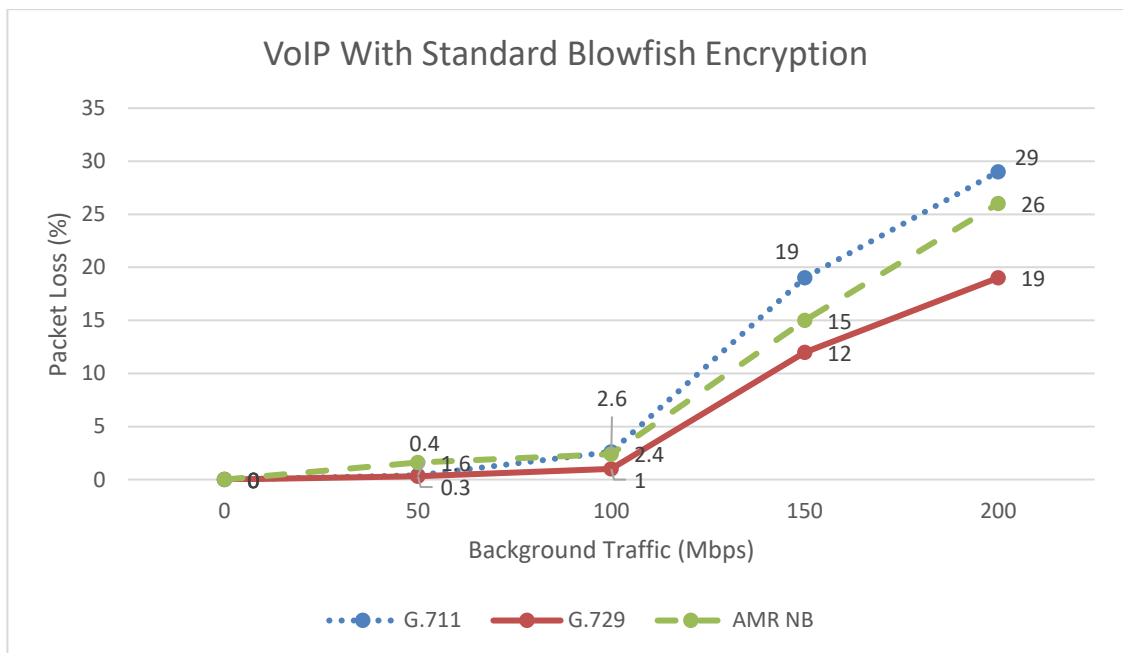
**Figure 45: Comparison of average jitter**

#### 5.2.4 PACKET LOSS RATIOS

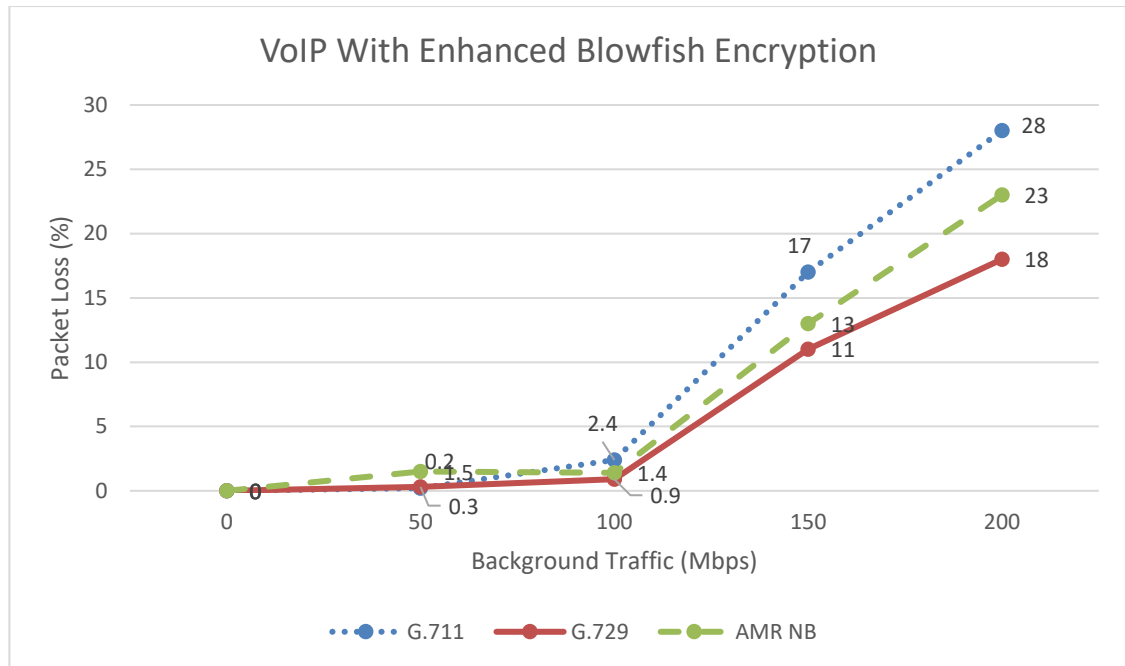
Figures 46, 47, and 48, respectively, illustrate the packet loss ratios in percentage as the background traffic was increased beginning from 0 until 50 Mbps, where packet loss was excluded from all voice codecs under all scenarios (nil encryption, as well as standard and enhanced Blowfish encryption) tested, weighing in only single call setup and network utilization that is below 50% of the allocated bandwidth. Besides, starting from 50 Mbps background traffic, packet loss below 1% was noted for all voice codecs and settings. However, when the background traffic was at 100Mbps, the G.711 voice codec for all scenarios tested suffered from a packet loss of about 2%, while the G.729 voice codec maintained the lowest for all settings. Next, at 150Mbps traffic load, the packet loss for all codecs rapidly increased between 9 and 12%, and when the maximum background traffic hit 200 Mbps, the highest 29% of packet loss had been recorded for G.711 voice codec with standard Blowfish encryption. Furthermore, a significant decrease in packet loss was noted under heavily overloaded network conditions, whereby in this network testbed, the G.711 voice codec began degrading significantly with higher ratios upon standard Blowfish encryption implementation. Meanwhile, the application of enhanced Blowfish encryption slightly decreased the packet loss ratios, in comparison to that of the standard Blowfish encryption algorithm.



**Figure 46: Packet loss ratio for voice codecs without encryption**



**Figure 47: Packet loss ratio for voice codecs with standard Blowfish encryption**



**Figure 48: Packet loss ratio for voice codecs with enhanced Blowfish encryption**

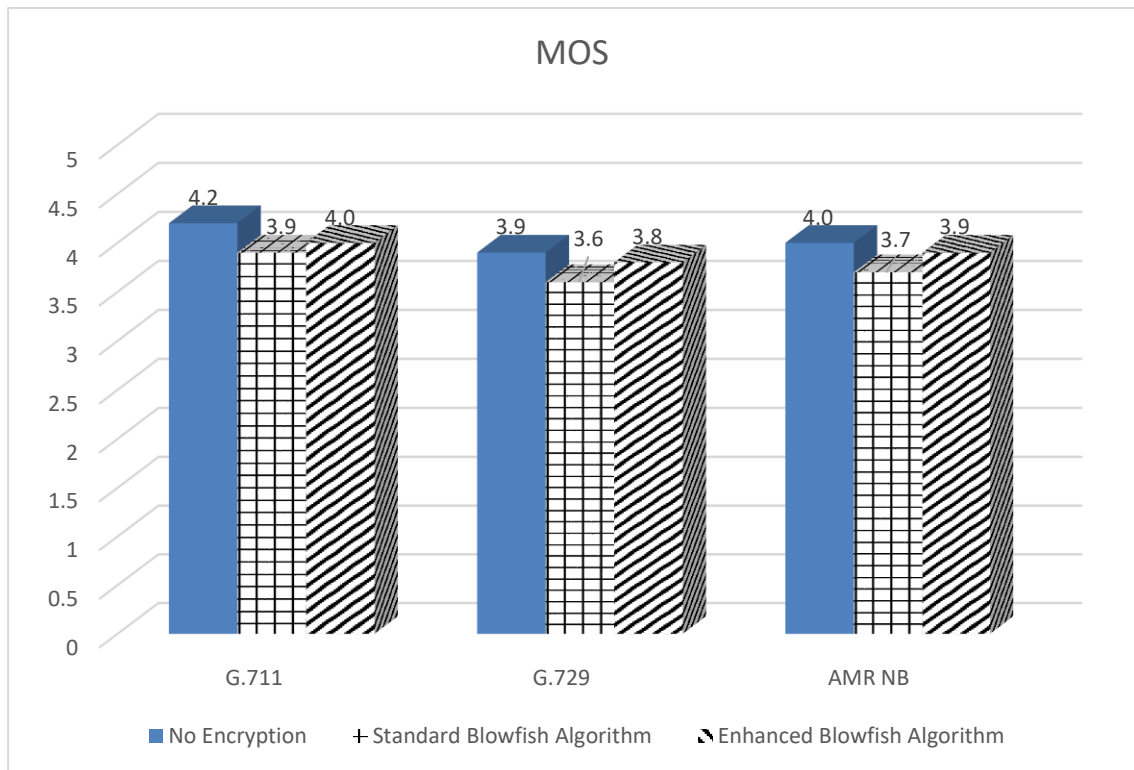
### 5.2.5 VOICE QUALITY MOS AND E MODEL VALUES

Various types of scenarios were generated by differing number of calls made, background traffic, node properties, and simulation parameter, as designed in Section 4.2, in which three different voice codecs (G.711, G729, and AMR-NB) had been compared so as to gain the best results. As such, the examination of MOS value, starting with network traffic without background traffic or security imposed, the scenarios of the testbed had been repeated with various parameters, including the implementation of both standard and enhanced Blowfish algorithms with 100Mbps of allocated bandwidth. Table 14 below displays the results of the experiments.

**Table 14: MOS values for voice codecs**

	G.711	G.729	AMR-NB
No Security	4.2	3.9	4.0
Standard Blowfish Algorithm	3.9	3.6	3.7
Enhanced Blowfish Algorithm	4.0	3.8	3.9

Table 14 proves that the codecs; G.711, G.729, and AMR-NB, resulted in varying voice qualities. Besides, the varying settings of the testbed gave varying results when encryption algorithm was implemented, as displayed in Figure 49. For instance, MOS scores for G.711 had been significantly the highest for all scenarios as the implementation of enhanced Blowfish encryption algorithm gave better voice quality, in comparison to that of the standard Blowfish algorithm. Meanwhile, G.729 provided the worst quality among the rest. Moreover, the test determined if the implementation of both encryption algorithms affected the quality of voice and their various impacts upon each voice codecs examined.



**Figure 49: MOS values for voice codecs under different scenarios**

### 5.3 VALIDATION AND VERIFICATION

The mathematical model known as E-Model for heterogeneous networks had been validated with the results obtained from the network testbed implementation, as presented in Figure 49, which referred to G.711, G.729, and AMR-NB voice codecs using network testbed. In fact, these values were translated to the E-model by using equation (1). Other than that, the results retrieved from the empirical tests for cascaded network performance had been compared with the mathematical model described in Section 2.4.2. Furthermore,

by comparing the results of both experiments and mathematical model, it was found that the model reflected a good and fair approach of the hardware/simulations model since the variances between both results were negligible. Hence, the mixed results revealed in this chapter have been considered as realistic for hardware implementation in predicting voice quality.

Moreover, by using the E-Model as a non-intrusive measurement method, as described in Section 2.4.2, the model combined the impairment caused by transmission parameters with R factor (overall transmission quality rating), where the R factor could be equivalently transformed to MOS, as depicted in Table 7 on page 57.

The factor R:

$$R = R_o - I_s - I_d - I_{e\_eff} + A \quad (1)$$

The process of encryption verification has been done by using Wireshark tcpdump software where VoIP data has been compared before and after encryption process using both algorithms in this thesis.

### 5.3.1 IMPACTS OF NETWORK DELAY, PACKET LOSS ON THE FACTOR R

Packet loss, the  $I_e$  factor referred to ITU-T Rec [49] gives  $I_d$  measured values for G.711, G.729 and AMR-NB codecs.

The results for  $I_e$  factor for G.711 which is derives from expression of the form:

$$I_e = 0 + 22 \ln (1 + 0.2 \cdot p.l) \quad (11)$$

Where 0 is typical value for G.711,  $p.l$  = packet loss in percent.

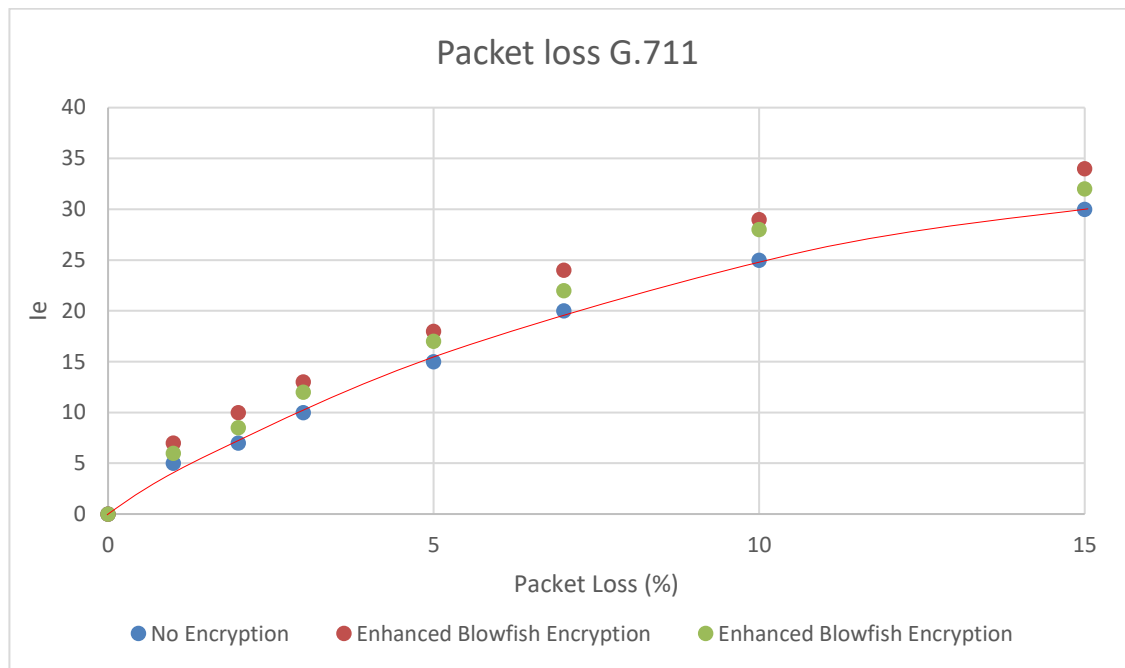
The results for  $I_e$  factor for G.729 which is derives from expression of the form:

$$I_e = 11 + 31 \ln (1 + 0.15 \cdot p.l) \quad (12)$$

The results for  $I_e$  factor for AMR-NB which is derives from expression of the form:

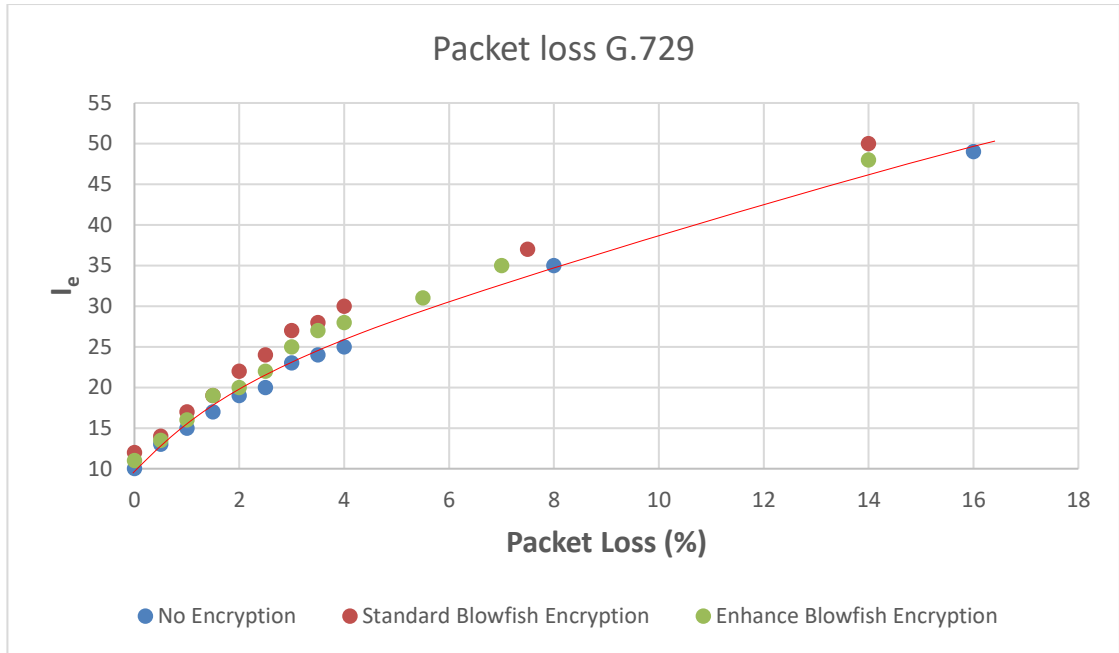
$$I_e = 5 + 33 \ln (1 + 0.1 \cdot p.l) \quad (13)$$

Figure 50 illustrates the impairment factor ( $I_e$ ) versus the packet loss for G.711 for all settings based on experimental setup that incorporated the curve fit (red line), as shown in equation (11), for voice codec without any encryption algorithm.



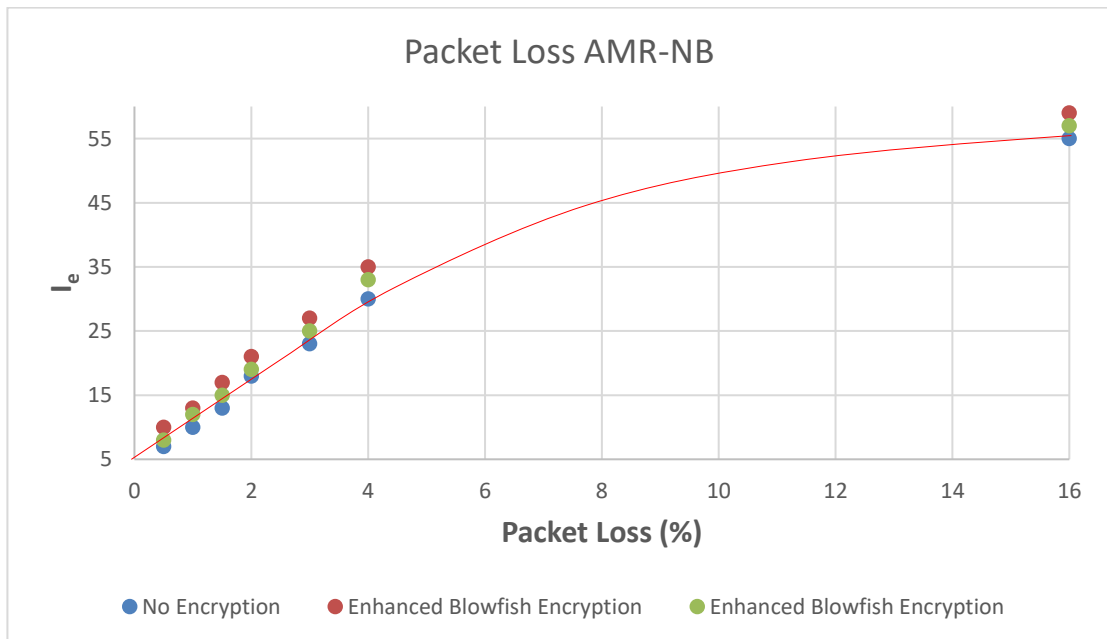
**Figure 50: Equipment impairment factor ( $I_e$ ) versus packet loss for G.711 codec**

Figure 51 presents the impairment factor ( $I_e$ ) versus the packet loss for G.729 given at all scenarios based on experimental setup that embedded curve fit (red line), as depicted in equation (12), for voice codec without any encryption algorithm.



**Figure 51: Equipment impairment factor ( $I_e$ ) versus packet loss for G.729 codec**

Figure 52 displays the impairment factor ( $I_e$ ) versus the packet loss for AMR-NB given at all scenarios based on experimental setup, where the curve fit (red line), as shown in equation (13), had been incorporated for voice codec without any encryption algorithm.



**Figure 52: Equipment impairment factor ( $I_e$ ) versus packet loss for AMR-NB codec**

Additionally, by considering the delay for packetized voice calls, as well as the contributions from encoding-decoding, packetization, service, dejittering delay



exceeding PSTN, and delay in packet network, the value of  $I_d$  was calculated derived from E-Model (using default values), adhering to [129]

The following expressions were applied for calculation:

$$I_d = 0.65 + (0.1T_a - 15.93) \cdot \delta(T_a - 165) \quad (14)$$

Where:  $\delta(x) = 0$  for  $x < 0$ ,  $\delta(x) = 1$  for  $x > 0$

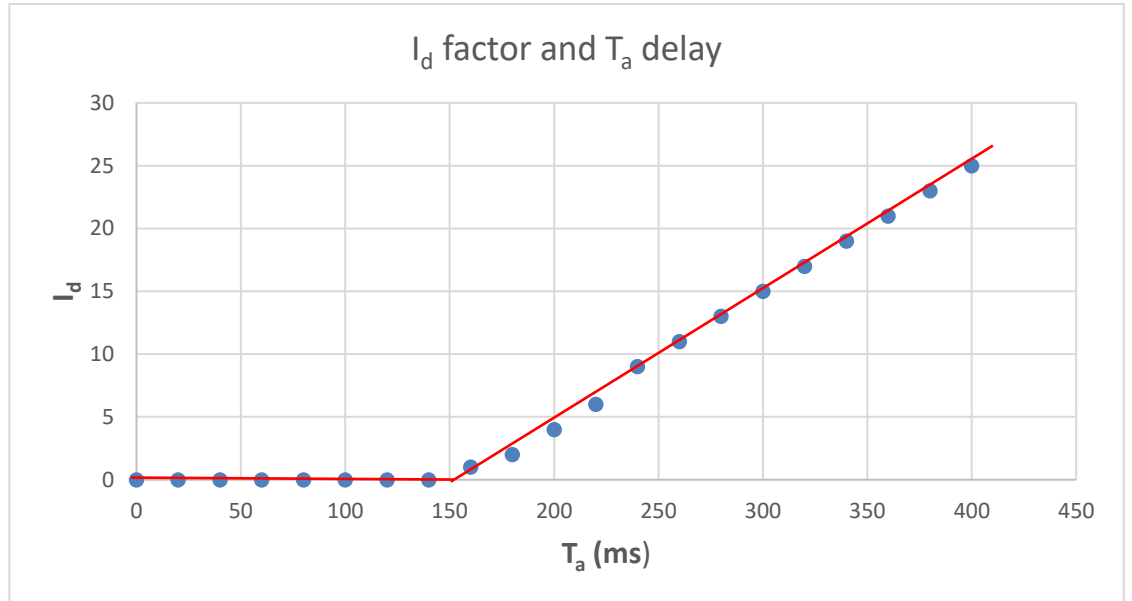
The end-to-end delay in packet connections is given by:

$$T_a = T_{enc} + T_p + T_{dec} + T_n \quad (15)$$

Where:  $T_{enc}$  = encoding delay,  $T_p$  = Packetization delay,  $T_{dec}$  = Decoding delay

$T_n$  = network delay (equivalent to sum of queuing, dejittering delay, propagation, and services [132])

Meanwhile, Figure 53 presents the correlation between delay in packet network, ( $I_d$ ) factor, and delay, ( $T_a$ ), where the results (plot) had been based on the E-Model and the curve fit (red line) derived from equation (15).



**Figure 53: Relationship between  $I_d$  factor and  $T_a$  delay**

The impacts of network delay and packet loss on factor  $R$  by using the default values depicted in [9], equation (1) was reduced to equation (16):

$$R = 94 - I_d - I_e \quad (16)$$

Substituting equation (11) – (15) into (16), the factor R:

For G.711 codec:

$$R = 93.35 - (0.1T_{n \max} - 15.90) \cdot \delta(T_{n \max} - 164.75) - 22 \ln(1 + 0.2 \cdot p.l) \quad (17)$$

For G.729 codec:

$$R = 82.35 - (0.1T_{n \max} - 15.43) \cdot \delta(T_{n \max} - 130) - 31 \ln(1 + 0.2 \cdot p.l) \quad (18)$$

For AMR-NB codec:

$$R = 88.35 - (0.1T_{n \max} - 15.00) \cdot \delta(T_{n \max} - 154.75) - 33 \ln(1 + 0.1 \cdot p.l) \quad (19)$$

Where  $T_{n \max}$  = maximum network delay.

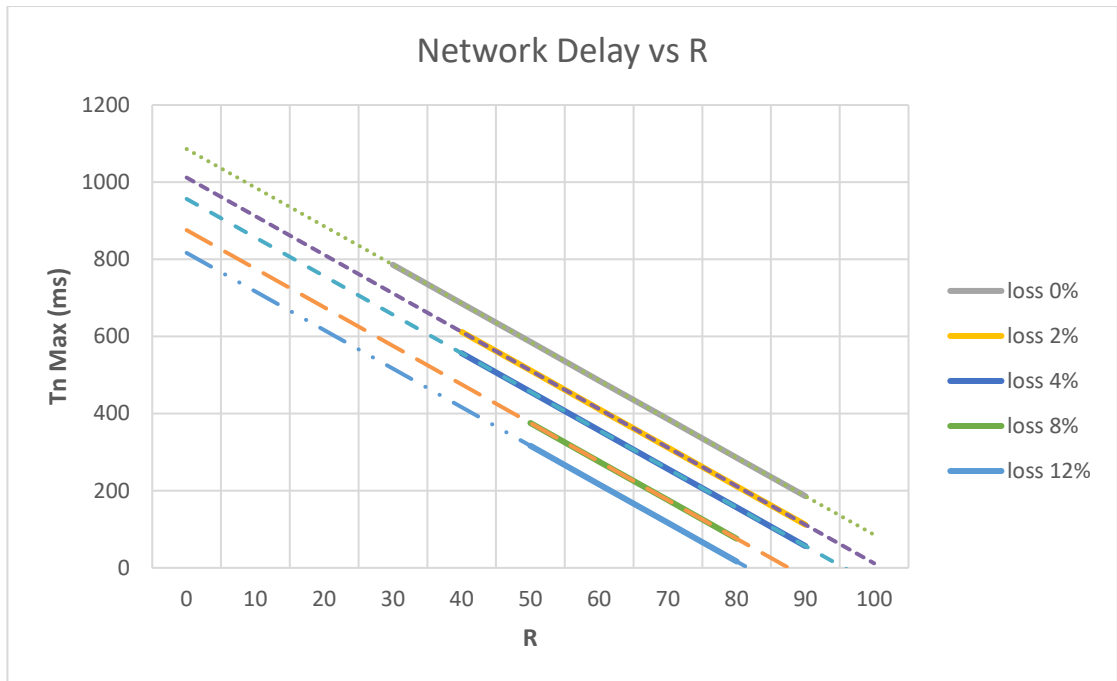
Assuming equations (17), (18), and (19), the minimum values of the sum for the encoding delay, the decoding delay, and the packetization delay are [49]:

G.711 codec: 0.25ms

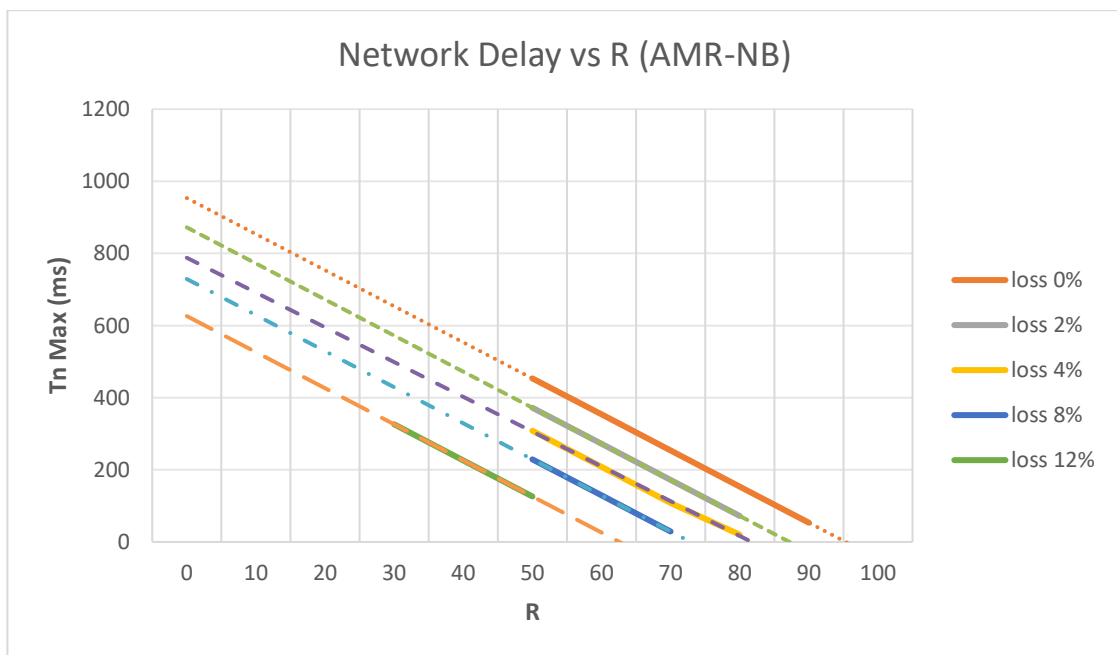
G.729 codec: 35ms

AMR-NB codec: 40ms

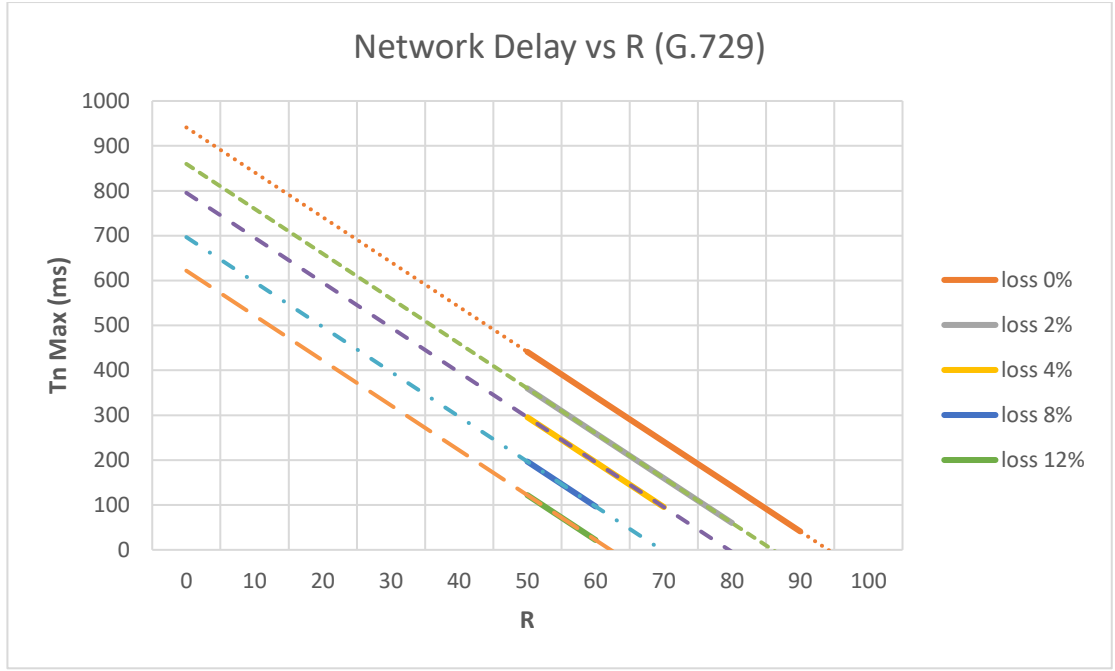
Figures 54, 55, and 56 present the correlations of network delay ( $T_{n \max}$ ) with R factor for G.711, G.729, and AMR-NB, which had been verified and validated with MOS scores in the experiments.



**Figure 54: Network delay versus factor R for G.711 codec**



**Figure 55: Network delay versus factor R for AMR-NB codec**



**Figure 56: Network delay versus factor R for G.729 codec**

The values of  $T_n$  max for R under conditions of packet loss for G.711, G.729, and AMR NB codecs are given from Figures 54 until 56, respectively, which resulted in a decrease in overall transmission quality due to increment in packet loss and delay. In fact, the voice quality achieved “Satisfied” and “Very satisfied” MOS scores if the factor R was between 80 and 90. Moreover, as user satisfaction was closely related to voice quality, hence, the results for each codec was factor  $R > 90$ , where G.711 attained a score at packet loss = 0 and  $T_n \leq 185.8$  ms; packet loss = 2% and  $T_n \leq 111.8$  ms; packet loss at 4% and  $T_n \leq 56.5$  ms. As for G.729, it reached the factor  $R > 90$  score at packet loss = 0 and  $T_n \leq 41.1$  ms. Lastly, the AMR-NB codec attained factor  $R > 90$  at packet loss 0 and  $T_n \leq 53.4$  ms. With that, the analytical expression based on the E-Model had successfully verified and validated the impacts of packet loss and delay upon voice quality.

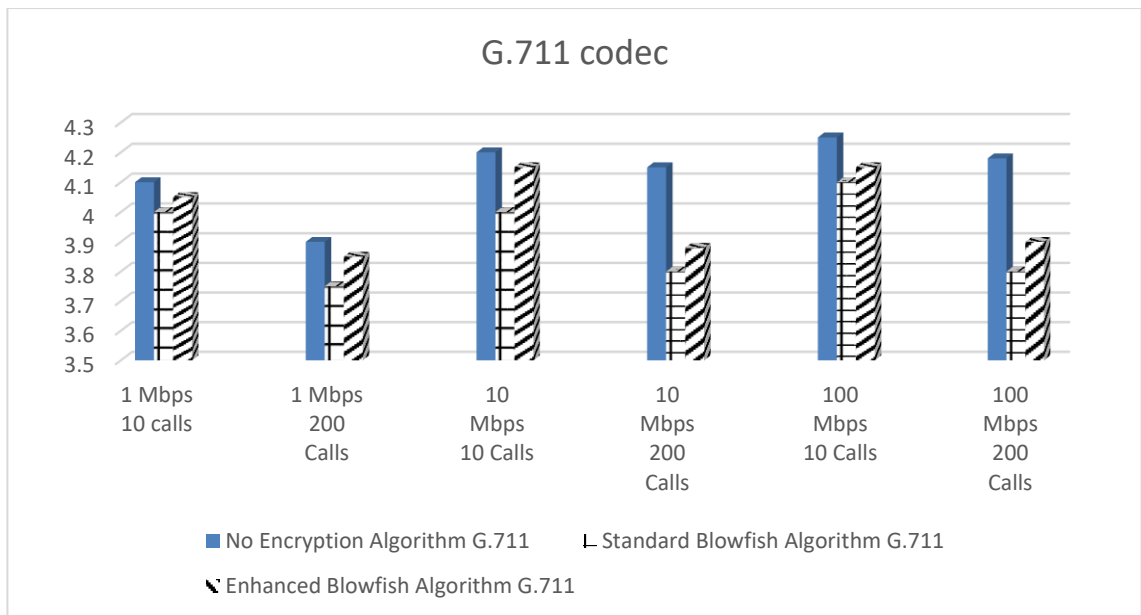
### 5.3.2 MOS SCORES

The findings for 36 experiments under various settings, as portrayed in Section 4.4, are presented in Table 15 and Figure 57 until 59. As such, all the three codecs with varied traffic and security scenarios were tested and the voice quality had been recorded, involving various bandwidths and number of calls simultaneously. As a result, G.711 recorded the highest MOS scores, followed by AMR-NB, and G.729 codecs. In addition, the MOS score significantly decreased under the imposed security, which obviously referred to the possibility and the start of both network delay and packet loss.

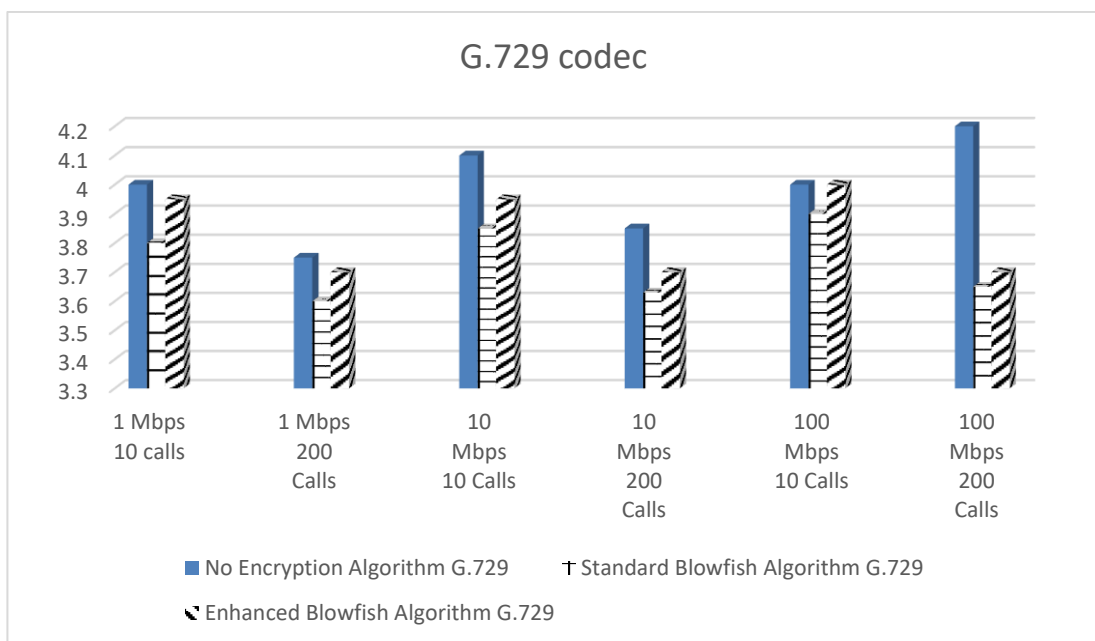
**Table 15: Comparison of MOS from G.711, G.729, and AMR-NB with and without encryption algorithms under different calls setup.**

Experiment parameter	No Encryption Algorithm			Standard Blowfish Algorithm			Enhanced Blowfish Algorithm		
	G.711	G.729	AMR-NB	G.711	G.729	AMR-NB	G.711	G.729	AMR-NB
10 calls 1 Mbps	4.10	4.00	4.05	4.00	3.80	3.95	4.05	3.95	4.00
200 Calls 1 Mbps	3.90	3.75	3.80	3.75	3.60	3.68	3.85	3.70	3.75
10 calls 10 Mbps	4.20	4.10	4.10	4.00	3.85	3.98	4.15	3.95	4.05
200 Calls 10 Mbps	4.15	3.85	3.90	3.80	3.63	3.70	3.88	3.70	3.85
10 Calls 100 Mbps	4.3	4.00	4.15	4.10	3.90	4.00	4.15	4.00	4.10
200 Calls 100 Mbps	4.2	4.00	4.05	3.80	3.65	3.75	3.90	3.70	3.80

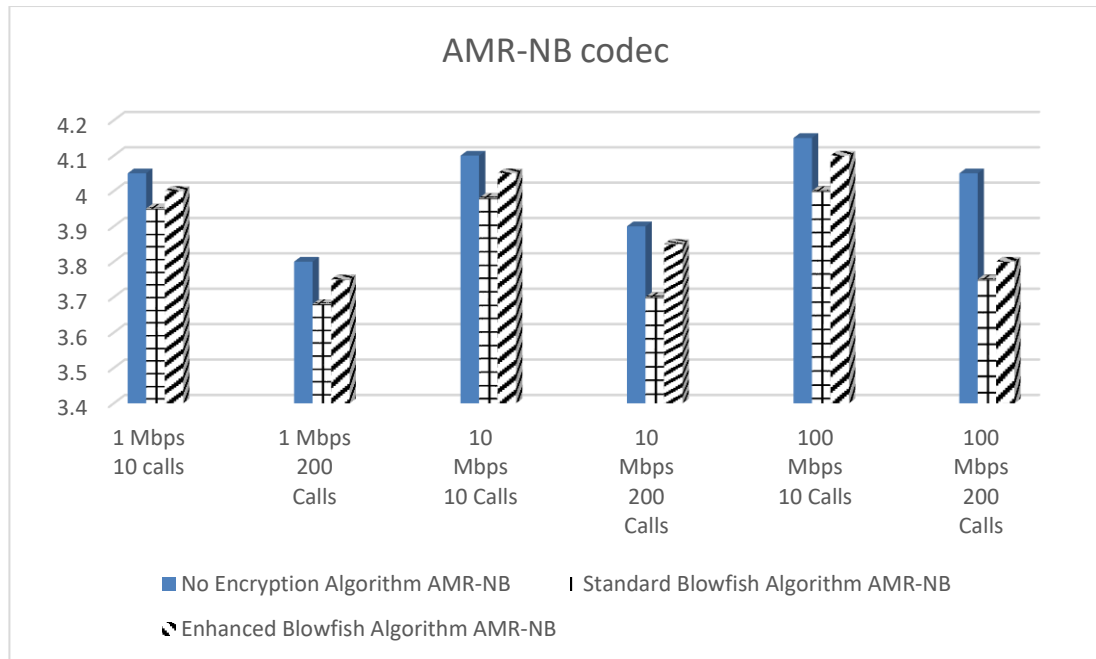
Figure 57, 58 and 59 illustrates the comparison between G.711, G.729, and AMR-NB codecs in accordance to MOS scores, where G.711 displayed the highest MOS with 100 Mbps and 10 calls setup, while the lowest MOS was recorded by G.729 codec under the implementation of standard Blowfish algorithm with 1 Mbps and 200 calls set up. Thus, the results showed that both bandwidth limitation and a hike in calls simultaneously affected the MOS scores for each voice codec. Besides, the implementation of encryption algorithms also contributed to packet delay and packet loss, which slightly reduced the quality in voice. In fact, the enhanced Blowfish algorithm exhibited better results, when compared to those of standard Blowfish algorithm, where the variance between an encrypted VoIP and one that is non-encrypted had been insignificant for voice quality degradation. Furthermore, the enhanced encryption exerted better performance than the MOS scores obtained for the standard Blowfish algorithm.



**Figure 57: Comparison of G.711 codec based on varying scenarios.**



**Figure 58: Comparison of G.729 codec based on varying scenarios.**



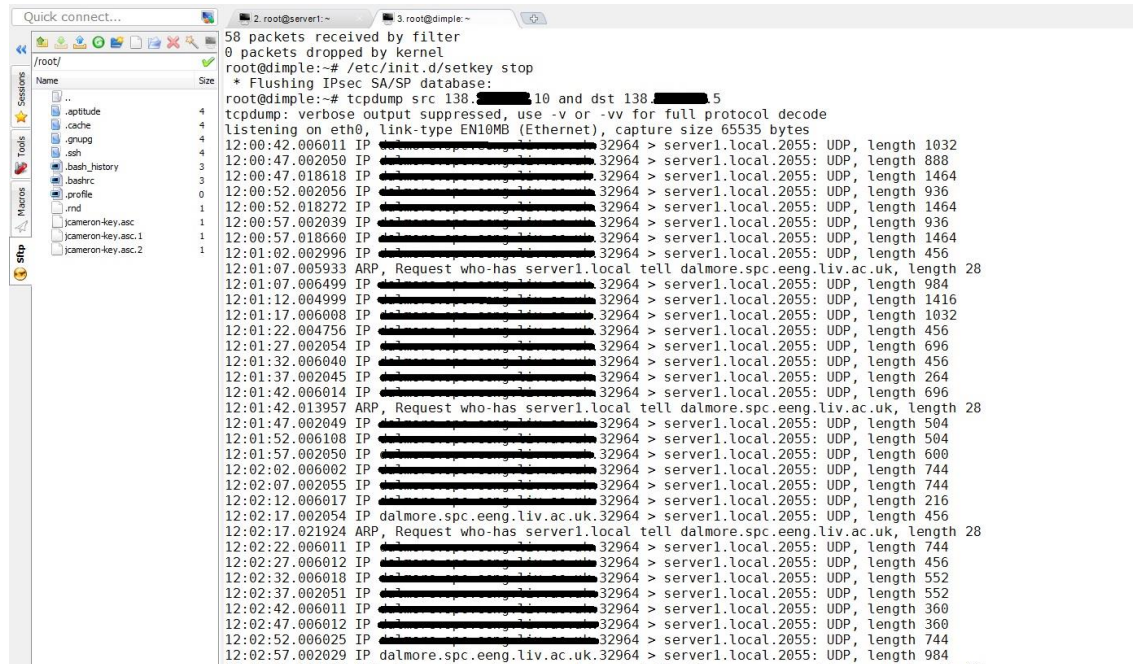
**Figure 59: Comparison of AMR-NB codec based on varying scenarios.**

From these experiments, some recommendations to enhance the voice quality of VoIP are given in the following:

- 1) G.711 displayed exceptional voice quality under two scenarios: without any encryption, and enhanced Blowfish encryption with high bandwidth
- 2) AMR-NB is strongly recommended for further usage, instead of G.711 and G.729 for cases with limited or reduction in bandwidth allocation.
- 3) The enhanced Blowfish encryption algorithm could offer adequate security as well as the standard Blowfish algorithm, besides providing better voice quality, reliability, and confidence to be used in VoIP transmission and services.

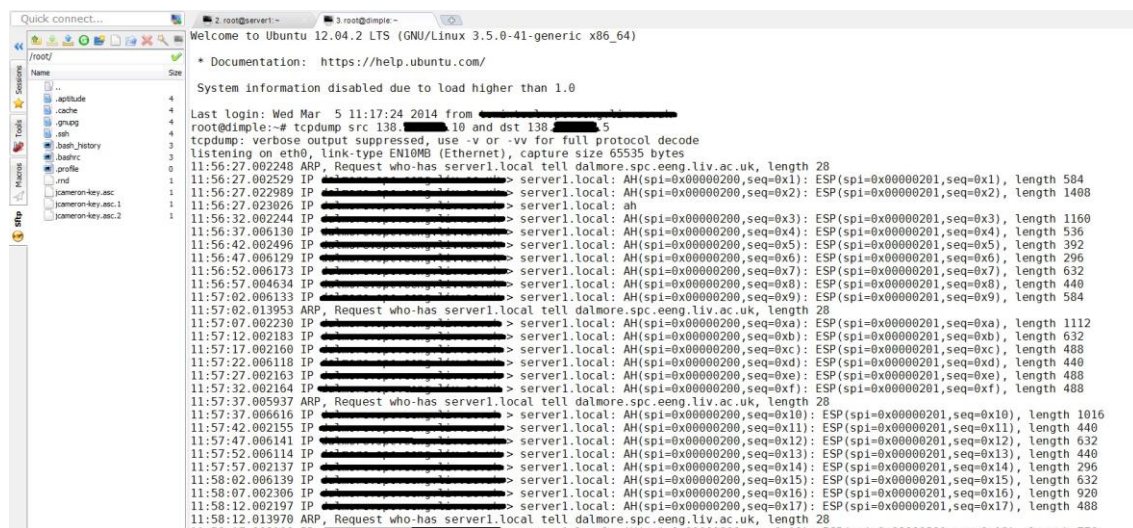
### 5.3.3 ENCRYPTIONS VERIFICATION

Figure 60 illustrates the VoIP data before the encryption process by using both encryption algorithms (Standard and enhance Blowfish algorithm) implementation in the experiments. From the captured screen, it is clear that when every packet is sent from sender and destination address, the port and protocol used in these unsecured transmission can be easily identified, manipulated, or compromised by unauthorized persons.

The image shows a Wireshark packet capture of unencrypted VoIP traffic. The left pane shows the packet list with 58 packets. The right pane shows the packet details for packet 1, which is an ARP request. The packet bytes pane shows the raw data of the ARP request. The packet list shows a series of UDP packets from 12.00.42.006011 to 12.02.57.002029, all destined to 138.10.10.5. The packet details show the Ethernet II, Internet Protocol Version 4, and User Datagram Protocol (UDP) headers. The packet bytes show the raw data of the ARP request, including the Ethernet II header, Internet Protocol Version 4 header, and User Datagram Protocol header.

**Figure 60: Unencrypted VoIP services**

Secured communication channel and transmission as shown in Figure 61 will provide confidential and protected information from being read, manipulated and jeopardized by attackers, whereas IPsec imposed in the experiments will provide data protection and verification process; compared to the condition of the VoIP data transmission before and after security implementation of the IPsec.

The image shows a Wireshark packet capture of encrypted VoIP traffic. The left pane shows the packet list with 58 packets. The right pane shows the packet details for packet 1, which is an ARP request. The packet bytes pane shows the raw data of the ARP request. The packet list shows a series of UDP packets from 11.56.27.002248 to 11.58.12.013970, all destined to 138.10.10.5. The packet details show the Ethernet II, Internet Protocol Version 4, and User Datagram Protocol (UDP) headers. The packet bytes show the raw data of the ARP request, including the Ethernet II header, Internet Protocol Version 4 header, and User Datagram Protocol header.

**Figure 61: Encrypted VoIP services**

## 5.4 SUMMARY

In this chapter, the enhanced Blowfish algorithm is proposed and had been implemented using C++ and MATLAB. The algorithms were implemented and measured with VoIP



system model by using several open source programs: OpenSwan VPN, Wireshark, Jperf, and PackETH. One of the objectives of this research is to develop and to improve the effectiveness of IPSec encryption algorithm based on standard Blowfish encryption algorithm, which had been found to display fast computation time and high encryption/decryption throughput. Moreover, both the standard and enhanced Blowfish algorithms were compared in terms of performance (throughput and execution time). Meanwhile, the implementation on IPSec VPN showed that the enhanced Blowfish algorithm exerted better performance and voice quality in VoIP services/calls setup. The findings also suggested that the enhanced blowfish algorithm contributed to lower delay, jitter, and packet loss, but better voice quality, in comparison to those portrayed by the standard Blowfish algorithm.

# CHAPTER 6

## CONCLUSION AND FUTURE WORK

This chapter, which serves as the last chapter, consists of the study conclusion based on the research findings retrieved, as well as potential research works.

### 6.1 CONCLUSION

Deploying IP telephony or VoIP has emerged as a major challenge for network designers and data network researchers. In fact, numerous challenges are present when offering security for VoIP, whereby the security imposed usually comes with another drawback, primarily because security, network performance, and voice quality appear to be conflicting requirements, for instance, initiating a security feature would have an impact upon the performance of network QoS and voice quality. With that, this study offers enhanced understanding of how the VoIP works within both the open and scalable Internet environment. By studying, searching and evaluating the literature related to existing encryption algorithm network testbed/ network simulator and voice codecs that can be implemented to VoIP services, this research provides information pertaining to VoIP performances under various network architectures and security configurations, where such information could enhance the existing security implementation, besides providing the opportunity for one to choose the most suitable encryption algorithm for implementation by related parties. Hence, after analysis and synthesis of the information gathered in the literature by identifying gaps in current knowledge, this study covers both the impact and the price needed to be paid in relation to the implementation of VoIP security dimension. Thus, a balance is sought between secured and exceptional VoIP performance services.

Encryption algorithms have played some massive roles in securing VoIP services; in which the IPSec has been employed in many applications, for example, packet encryption, Internet-based security, and security of VoIP services, where the Blowfish algorithm has emerged popular and has been widely applied. In fact, each encryption technique and algorithm have their benefits and shortcomings, as reported by countless researchers who had proven that the Blowfish algorithm exerts better performance and the strongest security via strong key size, as well as protection against

cryptanalysis. As to redesigning and developing an enhanced encryption/decryption algorithm based on an existing block cipher algorithm that minimizes computational resources without violating security requirement, therefore, in this research, enhancement of the standard Blowfish algorithm had been carried out so as to improve the performance and to maintain an acceptable security against cryptanalysis. These contributions could ensure that the VoIP services could indeed be advantageous for any broadcast network and with such security modifications, the related performance could be increased via network throughput, besides reducing end-to-end delay, minimising packet loss ratios, providing better voice quality, and maintaining an acceptable security measurement. This achievement proves that the enhanced encryption algorithms based on standard Blowfish algorithm work effectively without sacrificing the privacy of users during data/voice transmission, besides maintaining the existing security level. The proposed algorithm contributes to better end-to-end bandwidth utilisation and reduction of transmission delay. The last objective which is to improve the performance of VoIP network regarding end-to-end delay, jitter, and packet loss by using the enhanced Blowfish encryption algorithm has also been achieved where the VoIP services measured with MOS and E-Model indicated a better performance and results. Furthermore, obtaining a suitable and acceptable voice quality for VoIP without unwanted noise, delay, or packet loss (dropped sound) is necessary to ascertain satisfaction among its users.

The methodology had been employed throughout this research where the process began with designing the enhanced Blowfish algorithm, as well as the scenarios of network configurations to determine the impact of the enhanced Blowfish algorithm. Next, the existing encryption algorithms on VoIP services and the selection of voice codecs were implemented along with the encryption algorithm to measure the impact of each selection. The measurements included encryption algorithm computation cost, throughput and cryptanalysis, network QoS effectiveness (jitter, delay, packet loss), and voice quality (MOS and E-Model). Furthermore, the implementation of both standard and enhanced Blowfish algorithms had been carried out by using C++ and MATLAB programming language in order to compare and to evaluate the performance, as well as to determine the security of both algorithms using avalanche effect and correlation coefficient to study the resistance of the algorithms from cryptanalysis and the randomness of the algorithm output design. On the other hand, MOS and E-Model were employed to measure VoIP voice quality tested in network testbed and OPNET simulator,

in which the experiments identified the influential factors that affected the quality of voice for its value changed from 1 to 5, with the lowest value indicating the lowest quality of voice, while the highest value for the best voice quality. In addition, several acceptable QoS network performance parameters and MOS voice quality had been applied as a metric to evaluate the impact of implementing the suggested security techniques via enhanced Blowfish algorithm.

Throughout the simulation and the implementation stages, a total of six stages were involved, with each stage significantly contributed to the achievement of this research. The first process began with C++ programming language, which was used to simulate and to test the enhanced Blowfish algorithm, besides comparing it with the standard Blowfish algorithm, whereby modification of F function made in standard Blowfish algorithm could alter the operation of the existing S-Boxes sizes, which eventually decreased algorithm operation time. Additionally, MATLAB programming was used for cryptanalysis and security measurement, where both the avalanche effect and correlation coefficient results were compared for both algorithms. Moreover, different scenarios were set up in the network testbed experiments and OPNET network simulator so as to measure the quality of voice by using various voice codecs. Furthermore, since MOS had been relatively subjective, the tests took a long time and had been proven expensive. Therefore, the E-Model was used objectively before it was converted to MOS ratings, where the network testbed and the network simulator had been capable in accessing calls using E-Model and in providing results for comparison, verification, and validation purposes.

The encryption algorithms delivered enhanced performance at the cost of minimum computation resources, which contributed to lower execution time and higher throughput, significantly, from the implementation of these security features, in which efficient VoIP services with an acceptable or excellent voice quality performance could be achieved. Therefore, in order to achieve this, the experimental results were retrieved from the enhanced Blowfish algorithm, starting from the redesigning of the existing algorithm until comparison with the standard Blowfish, whereby the results showed that the enhanced Blowfish algorithm displayed lower execution time and higher throughput, which are significant due to the modification made to F function and S-Box randomness processes. As the performance of encryption algorithms was also related to security strength, both the algorithms were tested and evaluated by using cryptanalysis to compare

avalanche effects and correlation coefficient. Moreover, the modification of F function in the standard Blowfish algorithm affected the security strength, which led the enhanced Blowfish algorithm to exhibit less avalanche effect, but higher correlation coefficient, in comparison to those of the standard Blowfish algorithm, nonetheless, still projected high encryption quality and strong security features.

On top of that, the implementation of IPSec contributed to network overhead, which affected VoIP services in terms of network QoS (delay, jitter, packet loss) and MOS (values of voice quality). Thus, in this research, the proposed enhanced Blowfish algorithm had been quantitatively measured in terms of increase or decrease in network QoS with various voice codecs (G.711, G.729, and AMR-NB) employed as a combination of scenarios in the network testbed and the OPNET simulation to measure both VoIP E-Model and MOS voice quality. The findings showed that the enhanced Blowfish algorithm improved the performance of network QoS by decreasing delay, jitter, and packet loss, which significantly contributed to an acceptable level of voice quality. Furthermore, in the varied scenarios tested, the results showed that each codec contributed to various network delay, jitter, packet loss, and MOS scores. Furthermore, comparison between the standard and the enhanced Blowfish algorithms exemplified that the enhanced Blowfish algorithm offered the least delay, jitter, as well as acceptable packet loss ratios and MOS scores. The results in this research also had been verified and validated by using a mathematical model derived from E-Model. Other than that, by comparing the network testbed with OPNET simulation, a suitable choice of voice codec that can enhance the quality of voice had been determined. In addition, the results obtained from the present VoIP encryption revealed that better MOS scores were achieved by using G.711 voice codec under high availability of bandwidth, whereas the AMR-NB codec performed better under lower bandwidth, and G.729 recorded the lowest MOS scores under both bandwidth scenarios.

In conclusion, this thesis evidenced that the implementation of IPSec, along with enhanced Blowfish encryption algorithm, in VoIP services led to both network QoS efficiency and superior voice quality, in comparison to the standard Blowfish algorithm.

## 6.2 POTENTIAL RESEARCH WORKS

The emergence of new technologies with varying features has led to a heterogeneous VoIP network, where the end-to-end QoS and voice quality were weighed in so as to satisfy users. Although the proposed enhanced Blowfish encryption algorithm in this thesis significantly facilitated the incorporation of two networks to VoIP quality prediction, several potentials do exist to further exploit these benefits. However, it is essential to highlight here that the trade-off between the quality of network QoS, performance, and security are ongoing issues. Consequently, the following suggestions are listed as viable future research ideas:

1. Further work can investigate the impact of this algorithm upon another research domain, together with varied performance and security factors, if employed in other encryption applications.
2. Further investigation on security features focused on data integrity using hash algorithms (SHA, MD5) or authentication algorithms (Pre shared key, RSA).
3. Other available voice codecs (Narrowband, Wideband) can be investigated, where the impact of encryption algorithms can be studied and tested.
4. The E-Model also has the potential to be enhanced by modifying the mathematical model used, so as to offer higher accuracy, reliability, and enhanced standard of VoIP voice measurement.
5. The future research can analyze this proposed encryption algorithm on other network platforms, for example, wireless, zigbee, and mobile data transport, to name a few.

# REFERENCES

- [1] M. K. Carson, *Alexander Graham Bell: giving voice to the world*, Illustrate. Sterling Publishing Company, Inc., Oct, 2007.
- [2] J. Xia and S. Guha, "Voice over IP ( VoIP )," *Production*, vol. 44, no. 6405, pp. 1–33, 2010.
- [3] A. Sinaeepourfard and H. Mohamed Hussain, "Comparison of VoIP and PSTN services by statistical analysis," in *Proceedings - 2011 IEEE Student Conference on Research and Development, SCORED 2011*, 2011, pp. 459–461.
- [4] K. Rohloff, D. B. Cousins, S. Member, and D. Sumorok, "Scalable , Practical VoIP Teleconferencing With End-to-End Homomorphic Encryption," in *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, 2017, vol. 12, no. 5, pp. 1031–1041.
- [5] E. B. Setiawan, "Analisa Quality of Services (QoS) Voice over Internet Protocol (VoIP) dengan Protokol H.323 dan Session Initial Protocol (SIP)," *J. Ilm. Komput. dan Inform. ( KOMPUTA )*, vol. I, no. 2, pp. 1–8, 2012.
- [6] C. Jiang and P. Huang, "Research of monitoring VoIP voice QoS," in *Proceedings - 2011 International Conference on Internet Computing and Information Services, ICICIS 2011*, 2011, pp. 499–502.
- [7] J. A. Bergstra and C. A. Middelburg, "ITU-T Recommendation P.800.1 : Mean Opinion Score terminology," 2006.
- [8] H. Toral-Cruz, A. S. K. Pathan, and J. C. Ramirez Pacheco, "Accurate modeling of VoIP traffic QoS parameters in current and future networks with multifractal and Markov models," *Math. Comput. Model.*, vol. 57, no. 11–12, pp. 2832–2845, 2013.
- [9] J. A. Bergstra and C. A. Middelburg, "ITU-T, Recommendation G.107: The E-model, a computational model for use in transmission planning.," 2005.
- [10] P. Wuttidittachotti and T. Daengsi, "VoIP-quality of experience modeling: E-model and simplified E-model enhancement using bias factor," *Multimed. Tools Appl.*, vol. 76, no. 6, pp. 8329–8354, 2017.
- [11] H. Assem, D. Malone, J. Dunne, and P. O'Sullivan, "Monitoring VoIP call quality using improved simplified E-model," *2013 Int. Conf. Comput. Netw. Commun. ICNC 2013*, vol. 1, pp. 927–931, 2013.
- [12] M. Adel, H. Assem, B. Jennings, D. Malone, J. Dunne, and P. O'Sullivan, "Improved E-model for monitoring quality of multi-party VoIP communications," in *2013 IEEE Globecom Workshops, GC Wkshps 2013*, 2013, pp. 1180–1185.
- [13] H. Toral-Cruz, J. Argaez-Xool, L. Estrada-Vargas, and D. Torres-Rom, "An Introduction to VoIP: End-to-End Elements and QoS Parameters," *VoIP Technol.*, vol. ISBN 978-9, p. 336, 2011.
- [14] B. Goode, "Voice Over Internet Protocol (VoIP)," in *Proceedings of the IEEE*, 2002, vol. 90, no. 9, p. 51.
- [15] A. Sehgal, D. Ghosh, and C. Gandhi, "Literature Survey of VoIP Security," *Int. J. Emerg. Technol. Adv. Eng. Website www.ijetae.com*, vol. 5, no. 1, pp. 62–65, 2015.

- [16] A. Mazalek, Z. Vranova, and E. Stankova, "Analysis of the impact of IPSec on performance characteristics of VoIP networks and voice quality," in *ICMT 2015 - International Conference on Military Technologies 2015*, 2015.
- [17] A. J. Ghazali, W. Al-Nuaimy, and A. K. Nandi, "Simulation of the encryption of NetFlow packet capturing system using IPSec," *2012 5th Int. Conf. Comput. Devices Commun.*, vol. 3, pp. 1–4, Dec. 2012.
- [18] P. Backs and N. Pohlmann, "Influence of Security Mechanisms on the Quality of Services of VoIP," in *ISSE 2008 Securing Electronic Business Processes*, 2008, pp. 341–346.
- [19] P. Radmand and A. Talevski, "Impact of encryption on QoS in VoIP," *Proc. - Soc. 2010 2nd IEEE Int. Conf. Soc. Comput. PASSAT 2010 2nd IEEE Int. Conf. Privacy, Secur. Risk Trust*, pp. 721–726, 2010.
- [20] D. Butcher, X. Li, and J. Guo, "Security challenge and defense in VoIP infrastructures," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 6, pp. 1152–1162, 2007.
- [21] D. Cohen, "Specifications for the Network Voice Protocol (NVP)," 1976. [Online]. Available: <http://www.ietf.org/rfc/rfc741>.
- [22] J. Postel, "User Datagram Protocol. RFC 768 (Standard)," 1980. [Online]. Available: <https://tools.ietf.org/html/rfc768>.
- [23] V. Bajpai and J. Schönwälder, "IPv4 versus IPv6 - Who connects faster?," *Proc. 2015 14th IFIP Netw. Conf. IFIP Netw. 2015*, 2015.
- [24] "ITU-T H.323. Packet-Based Multimedia Communications Systems," 1999. [Online]. Available: <https://www.itu.int/rec/T-REC-H.323/e>.
- [25] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Proposed Standard)," 1996. [Online]. Available: <http://www.ietf.org/rfc/rfc1889.txt>.
- [26] M. J. Handley, H. Schulzrinne, E. M. Schooler, and J. Rosenberg, "Session Initiation Protocol," 1999. [Online]. Available: <https://www.ietf.org/rfc/rfc2543.txt>.
- [27] "Skype, video chat and voice call services," <https://www.skype.com/>. [Online]. Available: <https://www.skype.com/en/>.
- [28] S. Guha and N. Daswani, "An experimental study of the skype peer-to-peer voip system," *Europe*, vol. 6, pp. 5–10, 2005.
- [29] J. Postel, "Transmission Control Protocol. RFC 793 (Standard)," 1981. [Online]. Available: <http://www.ietf.org/rfc/rfc793.txt>.
- [30] J. Postel, "Internet Protocol. RFC 791 (Internet Standard)," 1981. [Online]. Available: <http://www.ietf.org/rfc/rfc791.txt>.
- [31] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard)." [Online]. Available: <http://www.ietf.org/rfc/rfc3550.txt>.
- [32] A. Davy, D. Botvich, and B. Jennings, "An Efficient Process for Estimation of Network Demand for QoS-Aware IP Network Planning," *6th IEEE Work. IP Oper. Manag.*, no. 4268, pp. 120–131, 2006.



- [33] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification. RFC 2205 (Proposed Standard)," <https://tools.ietf.org/pdf/rfc2205.pdf>, 1997. [Online]. Available: <https://tools.ietf.org/pdf/rfc2205.pdf>.
- [34] J. Wroclawski, "The Use of RSVP with IETF Integrated Services. RFC 2210," 1997. [Online]. Available: <https://tools.ietf.org/pdf/rfc2210.pdf>.
- [35] R. Braden, D. Clark, and S. . Shenker, "RFC1633: Integrated Services in the Internet Architecture: an Overview," *IETF RFC 1633*, July, 1994. [Online]. Available: <http://dl.acm.org/citation.cfm?id=RFC1633>.
- [36] S. Blake, S. Microsystems, and Z. Wang, "An Architecture for Differentiated Services. RFC 2475 (Informational)," 1998. [Online]. Available: <https://tools.ietf.org/pdf/rfc2475.pdf>.
- [37] Y. T'Joens, P. Crivellari, and B. Sales, "Layer Two Tunnelling Protocol (L2TP): ATM access network extensions. RFC 3301 (Proposed Standard)," 2002. [Online]. Available: <https://tools.ietf.org/pdf/rfc3301.pdf>.
- [38] V. Sharma and F. Hellstrand, "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery. RFC 3469 (Informational)," 2003. [Online]. Available: <https://tools.ietf.org/pdf/rfc3469.pdf>.
- [39] M. I. Ali, "Frame relay in public networks," *IEEE Communications Magazine*, pp. 72–78, 1992.
- [40] R. L. Freeman, "Asynchronous transfer mode," *Wiley-IEEE Press*, vol. 6, pp. 511–538, 2005.
- [41] C. Fineberg, "Introduction to Multiprotocol Label Switching (MPLS) Introduction to MPLS - Agenda Presenters," *MFA Forum*. pp. 1–100, 2006.
- [42] A. D. Keromytis, "A Comprehensive Survey of Voice over IP Security Research," *Ieee Commun. Surv. Tutorials*, vol. 14, no. 2, pp. 514–537, 2012.
- [43] A. A. Eskandar, M. R. Syed, and M. B. Zarei, "SIP over IP VPN : Performance Analysis," in *Proceedings on the International Conference on Internet Computing (ICOMP)*, 2014, p. 1.
- [44] S. Vuong and Kapil Kumar Singh, "Voice over IP Security," in *Network Security: Current Status and Future Directions*, no. February, C. Douligieris and D. N. Serpanos, Eds. 2007 the Institute of Electrical and Electronics Engineers, Inc, 2008, p. 384.
- [45] M. K. Ranganathan and L. Kilmartin, "Performance analysis of secure session initiation protocol based VoIP networks," *Comput. Commun.*, vol. 26, no. 6, pp. 552–565, 2003.
- [46] G. Aimes, S. Kalidindi, and M. Zekauskas, "A One-way Delay Metric for IPPM," 1999. [Online]. Available: <https://trac.tools.ietf.org/html/rfc2679>.
- [47] C. Demichelis and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)," 2002. [Online]. Available: <https://tools.ietf.org/html/rfc3393>.
- [48] G. Aimes, S. Kalidindi, and M. Zekauskas, "A One-way Packet Loss Metric for IPPM. RFC 2680 (Proposed Standard)," 1999. [Online]. Available:

<https://tools.ietf.org/pdf/rfc2680.pdf>.

- [49] ITU-T, “G.114 One-way transmission time,” *Ser. G Transm. Syst. MEDIA, Digit. Syst. NETWORKS Int. Teleph. Connect. circuits – Gen. Recomm. Transm. Qual. an entire Int. Teleph. Connect.*, pp. 1–20, 2003.
- [50] Y. Chen, T. Farley, and N. Ye, “QoS Requirements of Network Applications on the Internet,” *Information-Knowledge-Systems Manag.*, vol. 4, no. 1, pp. 55–76, 2004.
- [51] A. Lazzez, “VoIP Technology: Security Issues Analysis,” *Int. J. Emerg. Trends Technol. Comput. Sci.*, vol. 2, no. 4, pp. 1–9, 2013.
- [52] H. A. Mohammed and A. H. Ali, “Effect of some Security Mechanisms on the Qos VoIP Application using OPNET,” *Int. J. Curr. Eng. Technol.*, vol. INPRESSCO, no. ISSN 2277-4106, pp. 1626–1630, 2013.
- [53] B. Son, E. Nahm, and H. Kim, “VoIP encryption module for securing privacy,” *Multimed. Tools Appl.*, vol. 63, no. 1, pp. 181–193, 2013.
- [54] S. Gold, “Securing VoIP,” *Netw. Secur.*, vol. 2012, no. 3, pp. 14–17, 2012.
- [55] X. Wei, K. Sellal, and Y. Bouslimani, “Security implementation for a VoIP server,” *Proc. - 2012 Int. Conf. Comput. Sci. Serv. Syst. CSSS 2012*, pp. 983–985, 2012.
- [56] X. Wang and R. Zhang, *Voip Security. Vulnerabilities, Exploits, And Defenses*, vol. 81, 2011.
- [57] M. Jahanirad, A. L. N. Yahya, and R. M. Noor, “Security measures for VoIP application: A state of the art review,” *Sci. Res. Essays*, vol. 6, no. 23, pp. 4950–4959, 2011.
- [58] A. Azfar, K. K. R. Choo, and L. Liu, “Android mobile VoIP apps: a survey and examination of their security and privacy,” *Electron. Commer. Res.*, vol. 16, no. 1, pp. 73–111, 2016.
- [59] S. Ehlert, D. Geneiatakis, and T. Magedanz, “Survey of network security systems to counter SIP-based denial-of-service attacks,” *Comput. Secur.*, vol. 29, no. 2, pp. 225–243, 2010.
- [60] U. U. Rehman and A. G. Abbasi, “Security Analysis of VoIP Architecture for Identifying SIP Vulnerabilities,” in *2014 International Conference on Emerging Technologies (ICET)*, 2014, no. i, pp. 87–93.
- [61] T. Xie, G. Tu, C. Li, C. Peng, J. Li, and M. Zhang, “The Dark Side of Operational Wi-Fi Calling Services,” *2018 IEEE Conf. Commun. Netw. Secur.*, p. 1, 2018.
- [62] M. Patel, G. State, B. V. Buddhdev, and A. G. State, “Analysis of Security Threats in Voice Over Internet Protocol ( VOIP ),” *Control Theory Informatics*, vol. 3, no. 5, pp. 30–38, 2013.
- [63] J. C. Pelaez, E. B. Fernandez, and M. M. Larrondo-Petrie, “Misuse patterns in VoIP,” *Secur. Commun. Networks*, vol. 2, no. 6, pp. 635–653, 2009.
- [64] Y. P. Kosta, U. D. Dalal, and R. K. Jha, “Security comparison of wired and wireless network with firewall and Virtual Private Network (VPN),” *ITC 2010 - 2010 Int. Conf. Recent Trends Information, Telecommun. Comput.*, pp. 281–283, 2010.

- [65] M. Voznak, J. Safarik, and F. Rezac, "Threat prevention and intrusion detection in VoIP infrastructures," *Int. J. Math. Comput. Simul.*, vol. 7, no. 1, pp. 69–76, 2013.
- [66] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta, "Wireless Wakeups Revisited: Energy Management for Voip over Wi-fi Smartphones," *Proc. 5th Int. Conf. Mob. Syst. Appl. Serv.*, pp. 179–191, 2007.
- [67] H. Dhall, D. Dhall, S. Batra, and P. Rani, "Implementation of IPSec protocol," *Proc. - 2012 2nd Int. Conf. Adv. Comput. Commun. Technol. ACCT 2012*, pp. 176–181, 2011.
- [68] T. Rowan, "VPN technology: IPSEC vs SSL," *Netw. Secur.*, vol. 2007, no. 12, pp. 13–17, 2007.
- [69] O. Maurhart *et al.*, "New release of an open source QKD software : design and implementation of new algorithms , modularization and integration with IPSec," in *Qcrypt 2013*, 2013.
- [70] A. Chakrabarti and M. Govindarasu, "IP Security (IPSec)," in *Network Security: Current Status and Future Directions*, 2006, pp. 65–82.
- [71] S. Frankel, K. Kent, R. Lewkowski, A. D. Orebaugh, R. W. Ritchey, and S. R. Sharma, "Guide to IPsec VPNs," in *Special Publication (Nist SP) - 800-77*, 2005, p. 126.
- [72] D. Harkins and D. Carrel, "RFC 2409-The Internet Key Exchange (IKE)," 1998. [Online]. Available: <https://tools.ietf.org/pdf/rfc2409.pdf>.
- [73] C. Kaufman, "RFC 4306 - Internet Key Exchange (IKEv2) Protocol," 2005. [Online]. Available: <https://tools.ietf.org/pdf/rfc4306.pdf>.
- [74] S. Kent, "RFC 4302 - IP Authentication Header," 2005. [Online]. Available: <https://tools.ietf.org/pdf/rfc4302.pdf>.
- [75] S. Kent, "RFC 4303 - IP Encapsulating Security Payload (ESP)," 2005. [Online]. Available: <https://tools.ietf.org/pdf/rfc4303.pdf>.
- [76] M. Baer, R. Charlet, W. Hardaker, R. Story, and C. Wang, "RFC 4807 - IPsec Security Policy Database Configuration MIB," 2007. [Online]. Available: <https://tools.ietf.org/pdf/rfc4807.pdf>.
- [77] S. Kent and R. Atkinson, "RFC 2401 - Security Architecture for the Internet Protocol," <https://tools.ietf.org/pdf/rfc2401.pdf>, 1998. [Online]. Available: <https://tools.ietf.org/pdf/rfc2401.pdf>.
- [78] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "RFC 5995 - Internet Key Exchange Protocol Version 2 (IKEv2)," 2010. [Online]. Available: <https://tools.ietf.org/pdf/rfc5996.pdf>.
- [79] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "RFC 2408 - Internet Security Association and Key Management Protocol (ISAKMP)," 1998. [Online]. Available: <https://tools.ietf.org/pdf/rfc2408.pdf>.
- [80] S. Frankel and S. Krishnan, "RFC 6071 - IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," 2011. [Online]. Available: <https://tools.ietf.org/pdf/rfc6071.pdf>.
- [81] A. J. Ghazali, W. Al-nuaimy, A. Al-ataby, and M. A. Al-taei, "Building IPv6

- Based Tunneling Mechanisms for VoIP Security,” in *2016 13th International Multi-Conference on Systems, Signals & Devices (SSD)*, 2016, pp. 171–176.
- [82] C. Hohendorf, E. P. Rathgeb, E. Unurkhaan, and M. Tüxen, “Secure end-to-end transport over SCTP,” *J. Comput.*, vol. 2, no. 4, pp. 31–40, 2007.
  - [83] W. Chou, “Strategies to keep your VoIP network secure,” *IT Prof.*, vol. 9, no. 5, pp. 42–46, 2007.
  - [84] A. Dakur and S. Dakur, “Eavesdropping and interception security hole and its solution over VoIP service,” in *Proceedings - 2014 IEEE Global Conference on Wireless Computing and Networking, GCWCN 2014*, 2015, pp. 6–10.
  - [85] A. Wahab, R. B. Bahaweres, M. Alaydrus, Muhaemin, and R. Sarno, “Performance analysis of VoIP client with integrated encryption module,” in *1st International Conference on Communications, Signal Processing and Their Applications, ICCSPA*, 2013.
  - [86] D. Barison, R. S. Miani, G. D. Breda, L. S. Mendes, and B. B. Zarpelao, “Evaluating Quality of Encrypted VoIP Calls in a Simulation Environment,” *J. Inf. Assur. Secur.*, vol. 8, pp. 119–128, 2013.
  - [87] K. Mohamed, “QoS evaluation in VoIP software with and without Blowfish encryption module,” in *3rd International Conference on Control, Engineering & Information Technology (CEIT)*, 2015.
  - [88] A. Jain and D. Bhatnagar, “A Comparative Study of Symmetric Key Encryption Algorithms algorithms,” *IJCSN Int. J. Comput. Sci. Netw.*, vol. 3, no. 5, pp. 37–41, 2014.
  - [89] A. Passito, E. Mota, and R. Aguiuz, “Evaluating Voice Speech Quality in Networks with VPN / IP Sec,” in *13th IEEE International Conference on Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005*, 2005, pp. 161–165.
  - [90] R. Barbieri, D. Bruschi, and E. Rosti, “Voice over IPsec: analysis and solutions,” *18th Annu. Comput. Secur. Appl. Conf. 2002. Proceedings.*, 2002.
  - [91] S. S. Kolahi, Y. R. Cao, and H. Chen, “Bandwidth-IPSec security trade-off in IPv4 and IPv6 in Windows 7 environment,” in *2nd International Conference on Future Generation Communication Technologies, FGCT*, 2013, pp. 148–152.
  - [92] T. Nie and T. Zhang, “A study of DES and blowfish encryption algorithm,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, pp. 1–4, 2009.
  - [93] A. Alabaichi, R. Mahmood, F. Ahmad, and Mohammed S. Mechee, “Randomness Analysis on Blowfish Block Cipher Using ECB and CBC Modes,” *J. Appl. Sci.*, vol. 13, no. 6, pp. 768–789, 2013.
  - [94] J. W. Cornwell, “Blowfish Survey,” 1997. [Online]. Available: <https://cs.columbusstate.edu/cae-ia/studentpapers/cornwell.jason.pdf>.
  - [95] G. H. L. Krishnamurthy G.N, V. Ramaswamy, “Performance Enhancement of Blowfish Algorithm By Modifying Its Function,” in *Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications*, 2007, pp. 241–244.
  - [96] G. N. Krishnamurthy, V. Ramaswamy, G. H. Leela, and M. E. Ashalatha,

- “Performance enhancement of Blowfish and CAST-128 algorithms and Security analysis of improved Blowfish algorithm using Avalanche effect,” *Int. J. Comput. Sci. Netw. Secur. IJCSNS*, vol. 8, no. 3, pp. 244–250, 2008.
- [97] I. Alam and M. R. Khan, “Performance and Efficiency Analysis of Different Block Cipher Algorithms of Symmetric Key Cryptography,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 10, pp. 713–720, 2013.
- [98] R. K. Meyers and A. H. Desoky, “An Implementation of the Blowfish Cryptosystem,” in *IEEE International Symposium on Signal Processing and Information Technology, 2008. ISSPIT 2008.*, 2008, pp. 346–351.
- [99] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, “A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish,” in *Procedia Computer Science*, 2016, vol. 78, no. December 2015, pp. 617–624.
- [100] K. Nur, P. St, D. Darlis, and S. Si, “An implementation of data encryption for Internet of Things using blowfish algorithm on FPGA,” in *2014 2nd International Conference on Information and Communication Technology (ICoICT)*, 2014, pp. 75–79.
- [101] A. V. Mota, S. Azam, B. Shanmugam, C. Yeo, and K. Kannoorpatti, “Comparative Analysis of Different Techniques of Encryption for Secured Data Transmission,” in *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI-2017) Comparative*, 2017, pp. 231–237.
- [102] S. S. Nyit, A. Dhabhi, A. N. Nyit, and A. Dhabhi, “A Survey of Cryptographic Algorithms for IoT Devices,” *2018 IEEE Long Isl. Syst. Appl. Technol. Conf.*, pp. 1–8, 2018.
- [103] A. Nadeem and M. Y. Javed, “A Performance Comparison of Data Encryption Algorithms,” in *2005 International Conference on Information and Communication Technologies*, 2005, pp. 84–89.
- [104] A. Ramesh and Dr.A.Suruliandi, “Performance Analysis of Encryption Algorithms for Information Security Substitute bytes,” *Power Comput. Technol. (ICCPCT), 2013 Int. Conf. Circuits*, pp. 840–844, 2013.
- [105] P. Rohilla, “Blowfish Algorithm: Security and Performance,” in *World Academy of Informatics and Management Sciences, WAIMS*, 2012, vol. 1, no. 5, pp. 1–6.
- [106] M. Panda, “Performance analysis of encryption algorithms for security,” in *International Conference on Signal Processing, Communication, Power and Embedded System, SCOPES 2016 - Proceedings*, 2017, pp. 278–284.
- [107] K. Allison, K. Feldman, and E. Mick, “Blowfish Key-Expansion Key-Expansion.”
- [108] A. T. Hashim, S. M. Al-qarrawy, and J. a Mahdi, “Design and Implementation of an Improvement Of Blowfish Encryption Algorithm,” *Iraqi J. Comput. Syst. Eng. IJCCCE*, vol. 9, no. 1, 2009.
- [109] B. Schneier and P. Gutmann, “Description of the Blowfish Cipher,” 2000. [Online]. Available: <https://tools.ietf.org/pdf/draft-schneier-blowfish-00.pdf>.
- [110] T. Triyason and P. Kanthamanon, “Perceptual Evaluation of Speech Quality Measurement on Speex Codec VoIP with Tonal Language Thai,” in *Communications in Computer and Information Science*, 2012, vol. 344, pp. 181–

- [111] S. Jelassi, G. Rubino, H. Melvin, H. Youssef, and G. Pujolle, "Quality of experience of VoIP service: A survey of assessment approaches and open issues," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 2, pp. 491–513, 2012.
- [112] H. Nyquist, "Certain topics in telegraph transmission theory," *Proc. IEEE*, vol. 90, no. 2, pp. 280–305, 2002.
- [113] S. Karapantazis and F. N. Pavlidou, "VoIP: A comprehensive survey on a promising technology," *Comput. Networks*, vol. 53, no. 12, pp. 2050–2090, 2009.
- [114] O. Slavata and J. Holub, "Impact of the codec and various QoS methods on the final quality of the transferred voice in an IP network," *J. Phys. Conf. Ser.*, vol. 588, no. 1, p. 012011, 2015.
- [115] A. Vizzarri, "Analysis of VoLTE End-To-End Quality of Service using OPNET," in *2014 UKSim-AMSS 8th European Modelling Symposium*, 2014, pp. 452–457.
- [116] Z. Li, S. Zhao, S. Bruhn, J. Wang, and J. Kuang, "Comparison and optimization of packet loss recovery methods based on AMR-WB for VoIP," *Speech Commun.*, vol. 54, no. 8, pp. 957–974, 2012.
- [117] 3CX PBX and Phone System for Windows, "Codec Summary Table." [Online]. Available: <http://www.en.voipforo.com/codec/codecs.php>.
- [118] ITU-T, "Pulse code modulation of (PCM) of Voice frequencies," *International Telecommunication Union, Recommendation G.711*, 1988. [Online]. Available: <https://www.itu.int/rec/T-REC-G.711-198811-I/en>.
- [119] ITU-T, "General aspects of digital transmission systems G.729 (03/96)," 1996. [Online]. Available: <https://www.ece.cmu.edu/~ece796/documents/g729.pdf>.
- [120] K. Srinivasan and A. Gersho, "Voice activity detection for cellular networks," in *Proceedings., IEEE Workshop on Speech Coding for Telecommunications*, 1993, pp. 85–86.
- [121] J. Zhang and N. Ansari, "On assuring end-to-end QoE in next generation networks: Challenges and a possible solution," *IEEE Commun. Mag.*, vol. 49, no. 7, pp. 185–191, 2011.
- [122] A. Amin, "VoIP Performance Measurement Using QoS Parameters," in *The Second International Conference on Innovations in Information Technology*, 2005, pp. 1–10.
- [123] A. Kovac and M. Halas, "Analysis of Influence of Network Performance Parameters on VoIP Call Quality," in *Knowledge in Telecommunication Technologies and Optic: KTTO, Ostrava: VSB-Technical University of Ostrava*, 2010, pp. 26–30.
- [124] E. Antwi-Boasiako, E. Kuada, and K. Boakye-Boateng, "Role of codec selection on the performance of IPsec secured VoIP," *2016 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2016*, pp. 2508–2514, 2016.
- [125] International Telecommunication Union, "P.800: Methods for subjective determination of transmission quality," *ITU-T Recommendation*, 1996. [Online]. Available: <https://www.itu.int/rec/T-REC-P.800-199608-I/en>.

- [126] L. Li, M. Rong, and G. Zhang, "An Internet of Things QoE evaluation method based on multiple linear regression analysis," in *2015 10th International Conference on Computer Science Education (ICCSE)*, 2015, no. Iccse, pp. 925–928.
- [127] J. Q. Walker, "Assessing VoIP Call Quality Using the E-model," *NetIQ Corporation* 2002., 2002. [Online]. Available: <https://pdfs.semanticscholar.org/a041/abc34bf803d2be4ab875d814086e812c5a1b.pdf>.
- [128] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," *IEEE Int. Conf. Acoust. Speech, Signal Process. Proc. (Cat. No.01CH37221)*, vol. 2, pp. 2–5, 2001.
- [129] ITU-T, "The E-model, a computational model for use in transmission planning," *Networks*, 2014. [Online]. Available: <https://www.itu.int/rec/T-REC-G.107-201402-S/en>.
- [130] A. D. Clark, "Modeling the effects of burst packet loss and recency on subjective voice quality," *IP Teleph. Work.*, 2001.
- [131] ITU-T, "Transmission impairments due to speech processing," 2007. [Online]. Available: <https://www.itu.int/rec/T-REC-G.113-200711-I/en>.
- [132] D. De Vleeschauwer, J. Janssen, G. Petit, and F. Poppe, "Quality bounds for packetized voice transport," *Alcatel Telecommun. Rev.*, pp. 19–24, 2000.
- [133] P. Radmand, J. Singh, A. Talevski, M. Domingo-Prieto, and J. Arnedo-Moreno, "The impact of security on voip call quality," *J. Mob. Multimed.*, vol. 7, no. 1, pp. 113–128, 2011.
- [134] P. Radman, J. Singh, M. Domingo, J. Arnedo, and A. Talevski, "VoIP: making secure calls and maintaining high call quality," *Proc. 8th Int. Conf. Adv. Mob. Comput. Multimed.*, pp. 56–62, 2010.
- [135] D. Barison, R. S. Miani, B. B. Zarpelao, G. Davis Breda, and L. De Souza Mendes, "Evaluation of quality in encrypted VoIP calls," *Proc. 2012 4th Int. Conf. Comput. Asp. Soc. Networks, CASoN 2012*, pp. 175–180, 2012.
- [136] S. Sukaridhoto *et al.*, "Teknik Keamanan pada VoIP dengan Virtual Private Networking dan Kriptografi Pada Jaringan Wireless LAN 802 . 11b Serta Korelasi Terhadap Bandwidth dan Intelligibility Suara Teknik Keamanan pada VoIP dengan Virtual Private Networking dan Kriptografi Pada J," in *The 7th SITIA 2006 (Seminar on Intelligent Technology and Its Application)*, 2006, no. 031, p. 1.
- [137] R. Sadiwala and M. Sharma, "Performance Analysis of VoIP Networks Using Network SImulators - A Review," *Int. J. Comput. Netw. Commun.*, vol. 2, no. 3, pp. 1–15, 2014.
- [138] D. Akbaş and H. Gümüşkaya, "Real and OPNET modeling and analysis of an enterprise network and its security structures," *Procedia Comput. Sci.*, vol. 3, pp. 1038–1042, 2011.
- [139] M. Babu, "Performance Analysis of IPSec VPN over VoIP Networks Using OPNET," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 2, no. 9, pp. 38–44, 2012.

- [140] G. I. Salama, M. E. Shehab, a a Hafez, and M. Zaki, "Performance Analysis of Transmitting Voice over Communication Links Implementing IPsec," in *13th International Conference on Aerospace Sciences & Aviation Technology, ASAT-13*, 2009, pp. 1–12.
- [141] M. A. Ali, I. Rashid, and A. A. Khan, "Selection of VoIP CODECs for Different Networks based on QoS Analysis," *Int. J. Comput. Appl. (0975 – 8887)*, vol. 84, no. 5, pp. 38–44, 2013.
- [142] M. Sulovic and M. Hadzialic, "Dynamic Codec Selection Algorithm for VoIP," in *ICDT 2011 : The Sixth International Conference on Digital Telecommunications Dynamic*, 2011, no. c, pp. 74–79.
- [143] A. Saad, "PhD Thesis - Secure VoIP Performance Measurement," Loughborough University, 2013.
- [144] H. Mustafa and W. Xu, "End-to-End Detection of Caller ID Spoofing Attacks," in *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, 2018, vol. 15, no. 3, pp. 423–436.
- [145] G. A. Safdar, P. Sant, G. Epiphaniou, and C. Maple, "Effects of iterative block ciphers on quality of experience for Internet Protocol Security enabled voice over IP calls," *IET Inf. Secur.*, vol. 6, no. 3, pp. 141–148, 2012.
- [146] T. Daengsi, "PhD Thesis - VoIP Quality Measurement: Recommendation Of MOS and Enhanced Objective Measurement Method For Standard Thai Spoken Language.," University Of Technology North Bangkok, 2012.
- [147] M. Riza and P. Nim, "Studi dan implementasi The Blowfish Encryption Algorithma dalam Bahasa Pemrograman C++." [Online]. Available: <http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2006-2007/Makalah/Makalah0607-30.pdf>.
- [148] A. Alabaichi, F. Ahmad, and R. Mahmood, "Security analysis of blowfish algorithm," in *2nd International Conference on Informatics and Applications, ICIA 2013*, 2013, pp. 12–18.
- [149] A. A. A. El-sadek and M. M. Fouad, "Speech Encryption Applying a Modified Blowfish Algorithm," in *2014 International Conference on Engineering and Technology (ICET)*, 2014, pp. 1–6.
- [150] V. Poonia and N. S. Yadav, "Analysis of modified Blowfish algorithm in different cases with various parameters," *ICACCS 2015 - Proc. 2nd Int. Conf. Adv. Comput. Commun. Syst.*, pp. 5–9, 2015.
- [151] K. G. N and V. Ramaswamy, "Performance Analysis of Blowfish and its Modified Version using Encryption quality , Key sensitivity , Histogram and Correlation coefficient analysis," *Int. J. Recent Trends Eng. Technol.*, vol. 1, no. 2, pp. 2–5, 2009.
- [152] R. S. Cordova, R. L. R. Maata, A. S. Halibas, and R. Al-azawi, "Comparative Analysis on the Performance of Selected Security Algorithms in Cloud Computing," in *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 2017, pp. 4–7.
- [153] A.-K. Al Tamimi, "Performance Analysis of Data Encryption Algorithms." [Online]. Available: <http://www.cs.wustl.edu/~jain/cse567->



- [154] E. P. Guillen and D. A. Chacon, “VoIP Networks Performance Analysis with Encryption Systems,” vol. 16, no. 2, pp. 688–695, 2009.
- [155] “RTP ToolBox™ Software.” [Online]. Available: <http://www.gl.com/rtptoolbox.html>.
- [156] “Packeth, GUI and CLI packet generator tool for Ethernet,” <http://packeth.sourceforge.net/packeth/Home.html>. [Online]. Available: <http://packeth.sourceforge.net/packeth/Home.html>.
- [157] P. Wuttidittachotti, T. Daengsi, C. Wutiwiwatchai, A. Preechayasomboon, and S. Sukparungsee, “VoIP quality of experience: A study of perceptual voice quality from G.729, G.711 and G.722 with Thai users referring to delay effects,” in *International Conference on Ubiquitous and Future Networks, ICUFN*, 2013, pp. 401–406.
- [158] D. Choudhary and A. Kumar, “Study and Performance of AMR Codecs for GSM,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 3, no. 10, pp. 8105–8110, 2014.
- [159] S. Srivastava, S. Anmulwar, A. M. Sapkal, T. Batra, A. K. Gupta, and V. Kumar, “Comparative study of various traffic generator tools,” in *2014 Recent Advances in Engineering and Computational Sciences, RA ECS 2014*, 2014.
- [160] S. Srivastava, S. Anmulwar, A. M. Sapkal, T. Batra, A. Gupta, and V. Kumar, “Evaluation of traffic generators over a 40Gbps link,” in *2014 Asia-Pacific Conference on Computer Aided System Engineering, APCASE 2014*, 2014, pp. 43–47.
- [161] S. Hakak, F. Anwar, S. A. Latif, G. Gilkar, and M. K. Alam, “Impact of packet size and node mobility pause time on average end to end delay and Jitter in MANET’s,” in *Proceedings - 5th International Conference on Computer and Communication Engineering: Emerging Technologies via Comp-Unication Convergence, ICCCE 2014*, 2015, pp. 56–59.
- [162] S. S. Kolahi, K. Mudaliar, C. Zhang, and Z. Gu, “Impact of IPSec security on VoIP in different environments,” in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2017, pp. 979–982.
- [163] I. Q. Abduljaleel, “Speech Encryption Using Chaotic Map and Blowfish Algorithms,” pp. 68–76, 2013.
- [164] L. Breslau *et al.*, “Advances in network simulation,” *Computer (Long. Beach. Calif.)*, vol. 33, no. 5, pp. 59–67, 2000.
- [165] X. Chang, “Network simulations with OPNET,” in *Proceedings of the 1999 Winter Simulation Conference*, 1999, pp. 307–314.
- [166] SteelCentral, “Riverbed,” *Riverbed*, 2016. [Online]. Available: <http://www.riverbed.com/products/steelcentral/steelcentral-riverbed-modeler.html>.
- [167] K. Wehrle, M. Günes, and J. Gross, *Modeling and Tools for Network Simulation*, 1st ed. Springer-Verlag Berlin Heidelberg, 2010.
- [168] M. Agrawal and P. Mishra, “A Modified Approach for Symmetric Key Cryptography Based on Blowfish Algorithm,” no. 6, pp. 79–83, 2012.

- [169] R. Yegireddi and R. K. Kumar, "A survey on conventional encryption algorithms of Cryptography," *Proc. 2016 Int. Conf. ICT Business, Ind. Gov. ICTBIG 2016*, 2017.
- [170] A. Alabaichi, F. Ahmad, and R. Mahmood, "Security analysis of blowfish algorithm," in *2013 Second International Conference on Informatics & Applications (ICIA)*, 2013, pp. 12–18.
- [171] M. P. Chaudhari and N. Parmar, "Blowfish Algorithm by Modify Randomness for S-Boxes using Fuzzy Value and Apply Encryption or Decryption on Image," vol. 3, no. 6, pp. 1525–1529, 2014.
- [172] M. Anand Kumar and S. Karthikeyan, "Investigating the Efficiency of Blowfish and Rijndael (AES) Algorithms," *Int. J. Comput. Netw. Inf. Secur.*, vol. 4, no. March, pp. 22–28, 2012.
- [173] H. D. Alaa Eldin Rohiem, Salah Elagooz, "A novel approach for designing the S-box of advanced encryption standard algorithm (AES) using chaotic map," in *Proceedings of the Twenty-Second National Radio Science Conference*, 2005, no. Nrsc, pp. 455–464.
- [174] E. M. Mahmoud *et al.*, "Dynamic AES-128 with Key-Dependent S-box," *Int. J. Eng. Res. Appl.*, pp. 1662–1670, 2013.
- [175] "Openswan, IPsec implementation for Linux." [Online]. Available: <http://www.openswan.org/>.
- [176] "Wireshark network protocol analyzer." [Online]. Available: <https://www.wireshark.org/>.
- [177] A. Fernandez-Duran and J. I. Alonso, "Approach for voice quality and throughput estimation in wireless convergent networks," *Wirel. Networks*, vol. 15, no. 2, pp. 227–239, 2009.
- [178] "iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool." [Online]. Available: <https://github.com/esnet/iperf>.
- [179] R. Hassan and M. K. Sailan, "End-to-End Baseline File Transfer Performance Testbed," *Inf. Technol. J.*, vol. 10, no. 2, pp. 446–451, 2011.
- [180] "voip-ipsec." [Online]. Available: <http://www.networkstraining.com/cisco-asa-qos-for-voip-traffic/>.
- [181] F. Palmieri and U. Fiore, "Providing true end-to-end security in converged voice over IP infrastructures," *Comput. Secur.*, vol. 28, no. 6, pp. 433–449, 2009.
- [182] S. Singh Gurpreet, "A Study of Encryption Algorithms ( RSA , DES , 3DES and AES ) for Information Security," *Int. J. Comput. Appl.*, vol. 67, no. 19, pp. 33–38, 2013.
- [183] M. Zamikhaya, P. Van Der Walt, and M. S. Phatle, "Videoconferencing Solution : Through Innovation Extension to Attain and Support Pre-Alzheimer Patients in Their Daily Activities," in *eTELEMED 2014 : The Sixth International Conference on eHealth, Telemedicine, and Social Medicine*, March 23 - 27, 2014, no. 1, pp. 74–79.



# APPENDICES

# APPENDIX A

## C++ ENHANCE BLOWFISH ALGORITHM

```
//blowfish.h

#include <iostream>
#include <string.h>
#include "blowfish.h"
using namespace std;
typedef unsigned char byte;

int main()
{
    BLOWFISH bf("FEDCBA9876543210");
    string asdf = "BlowwFIshhhhhhhhhhh!";
    asdf = bf.Encrypt_CBC(asdf);
    cout << "Encrypted: " << asdf << endl;
    asdf = bf.Decrypt_CBC(asdf);
    cout << "Decrypted: " << asdf;
    return 0;
}

*/

#ifndef BLOWFISH_INCLUDED
#define BLOWFISH_INCLUDED

#include <string>

//headers needed for the CSPRNG
#ifdef _WIN32
    #include <Windows.h>
    #include <Wincrypt.h>
```

```

#else

    #include <fstream>

#endif

typedef unsigned char byte;

class BLOWFISH{

    // 16 round version
    //STANDARD: 16
    //MAXIMUM: 256
    /**MUST be an EVEN number**
    #define ROUNDS 16

    public:

        BLOWFISH(std::string hexKey);
        BLOWFISH(byte* cipherKey, int keylength);

        //TODO: string encryption functions -> base64
        std::string Encrypt_CBC(std::string data);
        byte* Encrypt_CBC(byte* data, int length, int* newlength);
        byte* Encrypt_ECB(byte* data, int length, int* newlength);
        void Encrypt_Block(byte* block, int offset = 0);

        std::string Decrypt_CBC(std::string data);
        byte* Decrypt_CBC(byte* data, int length, int* newlength);
        byte* Decrypt_ECB(byte* data, int length, int* newlength);
        void Decrypt_Block(byte* block, int offset = 0);

        void SetRandomIV();
        void SetIV(byte* newIV);
        byte* GetIV();
        bool IvSet;

    protected:

```

```

void SetupKey(byte* cipherKey, int length);

void encipher();

void decipher();

unsigned int round(unsigned int a, unsigned int b, unsigned int n);

void setblock(byte* block, int offset);

void getblock(byte* block, int offset);

static unsigned int p[];

static unsigned int s0[];

static unsigned int s1[];

static unsigned int s2[];

static unsigned int s3[];


unsigned int xl_par;

unsigned int xr_par;


byte IV[8];


byte* Crypt_ECB(byte* data, int length, int* newlength, void (BLOWFISH::*CryptBlock)(byte*, int
offset), bool decrypt);

byte* Crypt_CBC(byte* data, int length, int* newlength, void (BLOWFISH::*CryptBlock)(byte*, int
offset), bool decrypt);

byte* padData(byte* data, int length, int* paddedLength, bool decrypt, bool IvSpace);

int findPaddingEnd(byte* data, int length);

int hex2dec(char hex);

std::string byteToHex(unsigned char x);
};


BLOWFISH::BLOWFISH(std::string hexKey)
{
    IvSet = false;

    if(hexKey.length() % 2 != 0)
        throw 2;

    byte key[hexKey.length() / 2];

    for(int i = 0; i < hexKey.length() / 2; i++)

```

```

    {
        key[i] = hex2dec(hexKey[i * 2]) * 8 + hex2dec(hexKey[i * 2 + 1]);
    }
    SetupKey(key, hexKey.length() / 2);
}

int BLOWFISH::hex2dec(char hex)
{
    if('a' <= hex && hex <= 'f')
        return 10 + (hex - 'a');
    if('A' <= hex && hex <= 'F')
        return 10 + (hex - 'A');
    return hex - '0';
}

BLOWFISH::BLOWFISH(byte* cipherKey, int keyLength)
{
    IvSet = false;
    SetupKey(cipherKey, keyLength);
}

byte* BLOWFISH::Encrypt_ECB(byte* data, int length, int* newlength)
{
    return Crypt_ECB(data,length, newlength, &BLOWFISH::Encrypt_Block, false);
}

byte* BLOWFISH::Decrypt_ECB(byte* data, int length, int* newlength)
{
    return Crypt_ECB(data,length, newlength, &BLOWFISH::Decrypt_Block, true);
}

byte* BLOWFISH::Encrypt_CBC(byte* data, int length, int* newlength)
{
    return Crypt_CBC(data,length, newlength, &BLOWFISH::Encrypt_Block, false);
}

```



```
}
```

```
byte* BLOWFISH::Decrypt_CBC(byte* data, int length, int* newlength)
{
    return Crypt_CBC(data,length, newlength, &BLOWFISH::Decrypt_Block, true);
}
```

```
std::string BLOWFISH::Encrypt_CBC(std::string data)
{
    byte* binaryData = new byte[data.length()];
    for(int i = 0; i < data.length(); i++)
        binaryData[i] = data[i];
    int newlen = 0;
    byte* result = Encrypt_CBC(binaryData,data.length(), &newlen);
    std::string encoded = "";
    for(int i = 0; i < newlen; i++)
        encoded += byteToHex(result[i]);
    delete [] result;
    delete [] binaryData;
    return encoded;
}
```

```
std::string BLOWFISH::Decrypt_CBC(std::string data)
{
    if(data.length() % 2 != 0)
        throw 2;
    byte binaryData[data.length() / 2];
    for(int i = 0; i < data.length() / 2; i++)
    {
        binaryData[i] = hex2dec(data[i * 2]) * 16 + hex2dec(data[i * 2 + 1]);
    }
    int len = 0;
    byte* cryptresult = Decrypt_CBC(binaryData, data.length() / 2, &len);
    std::string result = "";
```

```

    for(int i = 0; i < len; i++)
        result += cryptresult[i];
    delete [] cryptresult;
    return result;
}

std::string BLOWFISH::byteToHex(unsigned char x)
{
    char hex[17] = "0123456789ABCDEF";
    std::string result = "";
    result += hex[x / 16];
    result += hex[x % 16];
    return result;
}

byte* BLOWFISH::padData(byte* data, int length, int* paddedLength, bool decrypt, bool IvSpace =
false)
{
    int offset = 0;
    int dataoffset = 0;
    if(decrypt)
    {
        if(length % 8 != 0) throw 8;
        *paddedLength = length;
    }
    else
    {
        //
        *paddedLength = 8 + (length % 8 == 0 ? length : length + 8 - (length % 8)) + (IvSpace ? 8 : 0);
        //pad the data to a multiple of 8 plus one block
        if(IvSpace)
            offset = 8;
    }
}

```

```

//
byte* pData = new byte[*paddedLength];
for(int i = 0; i < length; i++)
    pData[offset + i] = data[i + dataoffset];

//
for(int i = length + offset; i < *paddedLength; i++)
    pData[i] = (pData[length - 1 + offset] ^ 0xCC); //fill the padding with a character that is different
from the last character in the plaintext, so we can find the end later

return pData;
}

int BLOWFISH::findPaddingEnd(byte* data, int length)
{
    int i = length;
    while(data[i - 1] == data[length - 1]) //
    {
        i--;
    }
    return i; //
}

byte* BLOWFISH::Crypt_ECB(byte* data, int length, int* newlength, void
(BLOWFISH::*CryptBlock)(byte*, int), bool decrypt)
{
    byte* pData;
    pData = padData(data,length,newlength, decrypt); //

    for(int i = 0; i < *newlength; i+=8) //run the encryption
    {
        (this->*CryptBlock)(pData,i);
    }
}

```

```

    if(decrypt) //
    {
        *newlength = findPaddingEnd(pData,*newlength);
    }
    return pData;
}

byte* BLOWFISH::Crypt_CBC(byte* data, int length, int* newlength, void
(BLOWFISH::*CryptBlock)(byte*, int ), bool decrypt)
{
    byte* pData;
    if(!decrypt && !IvSet)
        SetRandomIV();
    IvSet = false; // don't re-use an IV
    pData = padData(data,length,newlength, decrypt, true);

    if(!decrypt)
    {
        // for(int i = 0; i < 8; i++)
        pData[i] = IV[i];
    }
    else
    {
        for(int i = 0; i < 8; i++)
            IV[i] = pData[i];
    }
    byte nextIV[8];
    for(int i = 8; i < *newlength; i+=8) //
    {
        if(!decrypt)
        {
            for(int k = 0; k < 8; k++)
                pData[k + i] ^= pData[k + i - 8]; //
        }
    }
}

```

```

else
{
    for(int k = 0; k < 8; k++)
        nextIV[k] = pData[k + i];
}
(this->*CryptBlock)(pData,i);

if(decrypt)
{
    for(int k = 0; k < 8; k++)
    {
        pData[i + k] ^= IV[k];
        IV[k] = nextIV[k];
    }
}

}

if(decrypt) //
{
    *newlength = findPaddingEnd(pData,*newlength) - 8;
    byte* noIV = new byte[*newlength];
    for(int i = 0; i < *newlength; i++)
        noIV[i] = pData[i + 8];
    delete [] pData;
    pData = noIV;
}
return pData;
}

void BLOWFISH::SetRandomIV()
{
#ifdef _WIN32
    //
    HCRYPTPROV hCryptCtx = NULL;

```

```

        CryptAcquireContext(&hCryptCtx,      NULL,      MS_DEF_PROV,      PROV_RSA_FULL,
CRYPT_VERIFYCONTEXT);

    CryptGenRandom(hCryptCtx, 8, IV);
    CryptReleaseContext(hCryptCtx, 0);
#else
    std::ifstream devRand ("/dev/urandom", std::ios::in | std::ios::binary);
    if(!devRand.read((char*)&IV,8))
    {
        throw 1;
    }
#endif
    IvSet = true;
}

void BLOWFISH::SetIV(byte* newIV)
{
    IvSet = true;
    for(int i = 0; i < 8; i++)
        IV[i] = newIV[i];
}

byte* BLOWFISH::GetIV()
{
    byte* returnIV = new byte[8];
    for(int i = 0; i < 8; i++)
        returnIV[i] = IV[i];
    return returnIV;
}

void BLOWFISH::Encrypt_Block(byte* block, int offset)
{
    setblock(block,offset);
    encipher();
    getblock(block,offset);
}

```

```

}

void BLOWFISH::Decrypt_Block(byte* block, int offset)
{
    setblock(block,offset);

    decipher();

    getblock(block,offset);
}

void BLOWFISH::setblock(byte* block, int offset)
{
    //
    xr_par = 0; xl_par = 0;
    for(int i = 0; i < 4; i++)
    {
        xl_par = (xl_par << 8) + block[offset + i];
        xr_par = (xr_par << 8) + block[4 + offset + i];
    }
}

void BLOWFISH::getblock(byte* block, int offset)
{
    //
    unsigned int xl = xl_par;
    unsigned int xr = xr_par;
    for(int i = 3; i >= 0; i--)
    {
        block[i + offset] = xl % 256;
        block[i + 4 + offset] = xr % 256;

        xr = xr >> 8;
        xl = xl >> 8;
    }
}

```

```

void BLOWFISH::SetupKey(byte* cipherKey, int length)
{
    if(length > 56)
    {
        throw 56;
    }
    byte key[length];
    for(int i = 0; i < length; i++)
        key[i] = cipherKey[i];

    int j = 0;
    unsigned int d;
    for(int i = 0; i < 18; i++)
    {
        d = (((key[j % length] * 256 + key[(j + 1) % length]) * 256 + key[(j + 2) % length]) * 256 + key[(j
+ 3) % length]);
        p[i] ^= d;
        j = (j + 4) % length;
    }

    xl_par = 0;
    xr_par = 0;

    for(int i = 0; i < 18; i+=2)
    {
        encipher();
        p[i] = xl_par;
        p[i + 1] = xr_par;
    }

    for(int i = 0; i < 256; i+=2)
    {
        encipher();
        s0[i] = xl_par;

```



```

        s0[i + 1] = xr_par;
    }

    for(int i = 0; i < 256; i+=2)
    {
        encipher();
        s1[i] = xl_par;
        s1[i+ 1] = xr_par;
    }

    for(int i = 0; i < 256; i+=2)
    {
        encipher();
        s2[i] = xl_par;
        s2[i + 1] = xr_par;
    }

    for(int i = 0; i < 256; i+=2)
    {
        encipher();
        s3[i] = xl_par;
        s3[i + 1] = xr_par;
    }

}

void BLOWFISH::encipher()
{
    xl_par ^= p[0];
    for(int i = 0; i < ROUNDS; i+=2)
    {
        xr_par = round(xr_par, xl_par, i + 1);
        xl_par = round(xl_par, xr_par, i + 2);
    }
}

```

```

xr_par ^= p[ROUNDS + 1];

unsigned int swap = xl_par;
xl_par = xr_par;
xr_par = swap;
}

void BLOWFISH::decipher()
{
    xl_par ^= p[ROUNDS + 1];
    for(int i = ROUNDS; i > 0; i -= 2)
    {
        xr_par = round(xr_par, xl_par, i);
        xl_par = round(xl_par, xr_par, i - 1);
    }
    xr_par ^= p[0];

    unsigned int swap = xl_par;
    xl_par = xr_par;
    xr_par = swap;
}

unsigned int BLOWFISH::round(unsigned int a, unsigned int b, unsigned int n)
{
    //
    unsigned int x1 = (s0[(b >> 24) % 256] ^ s1[(b >> 16) % 256]) ;
    unsigned int x2 = x1 + s3[b % 256];
    unsigned int x3 = x1 ^ p[n];
    return x3 ^ a;
}

unsigned int BLOWFISH::s0[] = {
    0xd1310ba6,0x98dfb5ac,0x2ffd72db,0xd01adfb7,0xb8e1afed,0x6a267e96,
    0xba7c9045,0xf12c7f99,0x24a19947,0xb3916cf7,0x0801f2e2,0x858efc16,

```

0x636920d8,0x71574e69,0xa458fea3,0xf4933d7e,0x0d95748f,0x728eb658,  
0x718bcd58,0x82154aee,0x7b54a41d,0xc25a59b5,0x9c30d539,0x2af26013,  
0xc5d1b023,0x286085f0,0xca417918,0xb8db38ef,0x8e79dcb0,0x603a180e,  
0x6c9e0e8b,0xb01e8a3e,0xd71577c1,0xbd314b27,0x78af2fda,0x55605c60,  
0xe65525f3,0xaa55ab94,0x57489862,0x63e81440,0x55ca396a,0x2aab10b6,  
0xb4cc5c34,0x1141e8ce,0xa15486af,0x7c72e993,0xb3ee1411,0x636fbc2a,  
0x2ba9c55d,0x741831f6,0xce5c3e16,0x9b87931e,0xafd6ba33,0x6c24cf5c,  
0x7a325381,0x28958677,0x3b8f4898,0x6b4bb9af,0xc4bfe81b,0x66282193,  
0x61d809cc,0xfb21a991,0x487cac60,0x5dec8032,0xef845d5d,0xe98575b1,  
0xdc262302,0xeb651b88,0x23893e81,0xd396acc5,0x0f6d6ff3,0x83f44239,  
0x2e0b4482,0xa4842004,0x69c8f04a,0x9e1f9b5e,0x21c66842,0xf6e96c9a,  
0x670c9c61,0xabd388f0,0x6a51a0d2,0xd8542f68,0x960fa728,0xab5133a3,  
0x6eef0b6c,0x137a3be4,0xba3bf050,0x7efb2a98,0xa1f1651d,0x39af0176,  
0x66ca593e,0x82430e88,0x8cee8619,0x456f9fb4,0x7d84a5c3,0x3b8b5ebe,  
0xe06f75d8,0x85c12073,0x401a449f,0x56c16aa6,0x4ed3aa62,0x363f7706,  
0x1bfedf72,0x429b023d,0x37d0d724,0xd00a1248,0xdb0fead3,0x49f1c09b,  
0x075372c9,0x80991b7b,0x25d479d8,0xf6e8def7,0xe3fe501a,0xb6794c3b,  
0x976ce0bd,0x04c006ba,0xc1a94fb6,0x409f60c4,0x5e5c9ec2,0x196a2463,  
0x68fb6faf,0x3e6c53b5,0x1339b2eb,0x3b52ec6f,0x6dfc511f,0x9b30952c,  
0xcc814544,0xaf5ebd09,0xbee3d004,0xde334afd,0x660f2807,0x192e4bb3,  
0xc0cba857,0x45c8740f,0xd20b5f39,0xb9d3fbdb,0x5579c0bd,0x1a60320a,  
0xd6a100c6,0x402c7279,0x679f25fe,0xfb1fa3cc,0x8ea5e9f8,0xdb3222f8,  
0x3c7516df,0xfd616b15,0x2f501ec8,0xad0552ab,0x323db5fa,0xfd238760,  
0x53317b48,0x3e00df82,0x9e5c57bb,0xca6f8ca0,0x1a87562e,0xdf1769db,  
0xd542a8f6,0x287effc3,0xac6732c6,0x8c4f5573,0x695b27b0,0xbcca58c8,  
0xe1ffa35d,0xb8f011a0,0x10fa3d98,0xfd2183b8,0x4afcb56c,0x2dd1d35b,  
0x9a53e479,0xb6f84565,0xd28e49bc,0x4bfb9790,0xe1ddf2da,0xa4cb7e33,  
0x62fb1341,0xcee4c6e8,0xef20cada,0x36774c01,0xd07e9efe,0x2bf11fb4,  
0x95dbda4d,0xae909198,0xeaad8e71,0x6b93d5a0,0xd08ed1d0,0xafc725e0,  
0x8e3c5b2f,0x8e7594b7,0x8ff6e2fb,0xf2122b64,0x8888b812,0x900df01c,  
0x4fad5ea0,0x688fc31c,0xd1cff191,0xb3a8c1ad,0x2f2f2218,0xbe0e1777,  
0xea752dfe,0x8b021fa1,0xe5a0cc0f,0xb56f74e8,0x18acf3d6,0xce89e299,  
0xb4a84fe0,0xfd13e0b7,0x7cc43b81,0xd2ada8d9,0x165fa266,0x80957705,  
0x93cc7314,0x211a1477,0xe6ad2065,0x77b5fa86,0xc75442f5,0xfb9d35cf,

```

0xebcdaf0c,0x7b3e89a0,0xd6411bd3,0xae1e7e49,0x00250e2d,0x2071b35e,
0x226800bb,0x57b8e0af,0x2464369b,0xf009b91e,0x5563911d,0x59dfa6aa,
0x78c14389,0xd95a537f,0x207d5ba2,0x02e5b9c5,0x83260376,0x6295cfa9,
0x11c81968,0x4e734a41,0xb3472dca,0x7b14a94a,0x1b510052,0x9a532915,
0xd60f573f,0xbc9bc6e4,0x2b60a476,0x81e67400,0x08ba6fb5,0x571be91f,
0xf296ec6b,0x2a0dd915,0xb6636521,0xe7b9f9b6,0xff34052e,0xc5855664,
0x53b02d5d,0xa99f8fa1,0x08ba4799,0x6e85076a};

```

unsigned int BLOWFISH::s1[] = {

```

0x4b7a70e9,0xb5b32944,0xdb75092e,0xc4192623,0xad6ea6b0,0x49a7df7d,
0x9cee60b8,0x8fedb266,0xecaa8c71,0x699a17ff,0x5664526c,0xc2b19ee1,
0x193602a5,0x75094c29,0xa0591340,0xe4183a3e,0x3f54989a,0x5b429d65,
0x6b8fe4d6,0x99f73fd6,0xa1d29c07,0xfe830f5,0x4d2d38e6,0xf0255dc1,
0x4cdd2086,0x8470eb26,0x6382e9c6,0x021ecc5e,0x09686b3f,0x3ebaefc9,
0x3c971814,0x6b6a70a1,0x687f3584,0x52a0e286,0xb79c5305,0xaa500737,
0x3e07841c,0x7fdeae5c,0x8e7d44ec,0x5716f2b8,0xb03ada37,0xf0500c0d,
0xf01c1f04,0x0200b3ff,0xae0cf51a,0x3cb574b2,0x25837a58,0xdc0921bd,
0xd19113f9,0x7ca92ff6,0x94324773,0x22f54701,0x3ae5e581,0x37c2dad6,
0xc8b57634,0x9af3dda7,0xa9446146,0x0fd0030e,0xecc8c73e,0xa4751e41,
0xe23cd99,0x3bea0e2f,0x3280bba1,0x183eb331,0x4e548b38,0x4f6db908,
0x6f420d03,0xf60a04bf,0x2cb81290,0x24977c79,0x5679b072,0xbcaf89af,
0xde9a771f,0xd9930810,0xb38bae12,0xdccf3f2e,0x5512721f,0x2e6b7124,
0x501adde6,0x9f84cd87,0x7a584718,0x7408da17,0xbc9f9abc,0xe94b7d8c,
0xec7aec3a,0xdb851dfa,0x63094366,0xc464c3d2,0xef1c1847,0x3215d908,
0xdd433b37,0x24c2ba16,0x12a14d43,0x2a65c451,0x50940002,0x133ae4dd,
0x71dff89e,0x10314e55,0x81ac77d6,0x5f11199b,0x043556f1,0xd7a3c76b,
0x3c11183b,0x5924a509,0xf28fe6ed,0x97f1fbfa,0x9ebafb2c,0x1e153c6e,
0x86e34570,0xae96fb1,0x860e5e0a,0x5a3e2ab3,0x771fe71c,0x4e3d06fa,
0x2965dcb9,0x99e71d0f,0x803e89d6,0x5266c825,0x2e4cc978,0x9c10b36a,
0xc6150eba,0x94e2ea78,0xa5fc3c53,0x1e0a2df4,0xf2f74ea7,0x361d2b3d,
0x1939260f,0x19c27960,0x5223a708,0xf71312b6,0xebadfe6e,0xeac31f66,
0xe3bc4595,0xa67bc883,0xb17f37d1,0x018cff28,0xc332ddef,0xbe6c5aa5,
0x65582185,0x68ab9802,0xeecea50f,0xdb2f953b,0x2aef7dad,0x5b6e2f84,
0x1521b628,0x29076170,0xecdd4775,0x619f1510,0x13cca830,0xeb61bd96,

```

```

0x0334fe1e,0xaa0363cf,0xb5735c90,0x4c70a239,0xd59e9e0b,0xcbaade14,
0xeccc86bc,0x60622ca7,0x9cab5cab,0xb2f3846e,0x648b1eaf,0x19bdf0ca,
0xa02369b9,0x655abb50,0x40685a32,0x3c2ab4b3,0x319ee9d5,0xc021b8f7,
0x9b540b19,0x875fa099,0x95f7997e,0x623d7da8,0xf837889a,0x97e32d77,
0x11ed935f,0x16681281,0x0e358829,0xc7e61fd6,0x96dedfa1,0x7858ba99,
0x57f584a5,0x1b227263,0x9b83c3ff,0x1ac24696,0xcdb30aeb,0x532e3054,
0x8fd948e4,0x6dbc3128,0x58ebf2ef,0x34c6ffea,0xfe28ed61,0xee7c3c73,
0x5d4a14d9,0xe864b7e3,0x42105d14,0x203e13e0,0x45eee2b6,0xa3aaabea,
0xdb6c4f15,0xfac4fd0,0xc742f442,0xef6abb5,0x654f3b1d,0x41cd2105,
0xd81e799e,0x86854dc7,0xe44b476a,0x3d816250,0xcf62a1f2,0x5b8d2646,
0xfc8883a0,0xc1c7b6a3,0x7f1524c3,0x69cb7492,0x47848a0b,0x5692b285,
0x095bbf00,0xad19489d,0x1462b174,0x23820e00,0x58428d2a,0x0c55f5ea,
0x1dadf43e,0x233f7061,0x3372f092,0x8d937e41,0xd65fecf1,0x6c223bdb,
0x7cde3759,0xcbee7460,0x4085f2a7,0xce77326e,0xa6078084,0x19f8509e,
0xe8efd855,0x61d99735,0xa969a7aa,0xc50c06c2,0x5a04abfc,0x800bcadc,
0x9e447a2e,0xc3453484,0xfdd56705,0x0e1e9ec9,0xdb73dbd3,0x105588cd,
0x675fda79,0xe3674340,0xc5c43465,0x713e38d8,0x3d28f89e,0xf16dff20,
0x153e21e7,0x8fb03d4a,0xe6e39f2b,0xdb83adf7 };

```

```

unsigned int BLOWFISH::s2[] = {

```

```

0xe93d5a68,0x948140f7,0xf64c261c,0x94692934,0x411520f7,0x7602d4f7,
0xbcf46b2e,0xd4a20068,0xd4082471,0x3320f46a,0x43b7d4b7,0x500061af,
0x1e39f62e,0x97244546,0x14214f74,0xbf8b8840,0x4d95fc1d,0x96b591af,
0x70f4ddd3,0x66a02f45,0xbfb09ec,0x03bd9785,0x7fac6dd0,0x31cb8504,
0x96eb27b3,0x55fd3941,0xda2547e6,0xabca0a9a,0x28507825,0x530429f4,
0x0a2c86da,0xe9b66dfb,0x68dc1462,0xd7486900,0x680ec0a4,0x27a18dee,
0x4f3ffea2,0xe887ad8c,0xb58ce006,0x7af4d6b6,0xaace1e7c,0xd3375fec,
0xce78a399,0x406b2a42,0x20fe9e35,0xd9f385b9,0xee39d7ab,0x3b124e8b,
0x1dc9faf7,0x4b6d1856,0x26a36631,0xae397b2,0x3a6efa74,0xdd5b4332,
0x6841e7f7,0xca7820fb,0xfb0af54e,0xd8feb397,0x454056ac,0xba489527,
0x55533a3a,0x20838d87,0xfe6ba9b7,0xd096954b,0x55a867bc,0xa1159a58,
0xcca92963,0x99e1db33,0xa62a4a56,0x3f3125f9,0x5ef47e1c,0x9029317c,
0xfdf8e802,0x04272f70,0x80bb155c,0x05282ce3,0x95c11548,0xe4c66d22,
0x48c1133f,0xc70f86dc,0x07f9c9ee,0x41041f0f,0x404779a4,0x5d886e17,

```

```

0x325f51eb,0xd59bc0d1,0xf2bcc18f,0x41113564,0x257b7834,0x602a9c60,
0xdff8e8a3,0x1f636c1b,0x0e12b4c2,0x02e1329e,0xaf664fd1,0xcad18115,
0x6b2395e0,0x333e92e1,0x3b240b62,0xeebeeb922,0x85b2a20e,0xe6ba0d99,
0xde720c8c,0x2da2f728,0xd0127845,0x95b794fd,0x647d0862,0xe7ccf5f0,
0x5449a36f,0x877d48fa,0xc39dfd27,0xf33e8d1e,0x0a476341,0x992eff74,
0x3a6f6eab,0xf4f8fd37,0xa812dc60,0xa1ebddf8,0x991be14c,0xdb6e6b0d,
0xc67b5510,0x6d672c37,0x2765d43b,0xcdcd0e804,0xf1290dc7,0xcc00ffa3,
0xb5390f92,0x690fed0b,0x667b9ffb,0xcdeb7d9c,0xa091cf0b,0xd9155ea3,
0xbb132f88,0x515bad24,0x7b9479bf,0x763bd6eb,0x37392eb3,0xcc115979,
0x8026e297,0xf42e312d,0x6842ada7,0xc66a2b3b,0x12754ccc,0x782ef11c,
0x6a124237,0xb79251e7,0x06a1bbe6,0x4bfb6350,0x1a6b1018,0x11caedfa,
0x3d25bdd8,0xe2e1c3c9,0x44421659,0x0a121386,0xd90cec6e,0xd5abea2a,
0x64af674e,0xda86a85f,0xebbf9e88,0x64e4c3fe,0x9dbc8057,0xf0f7c086,
0x60787bf8,0x6003604d,0xd1fd8346,0xf6381fb0,0x7745ae04,0xd736fcc,
0x83426b33,0xf01eab71,0xb0804187,0x3c005e5f,0x77a057be,0xbde8ae24,
0x55464299,0xbf582e61,0x4e58f48f,0xf2ddfa2,0xf474ef38,0x8789bdc2,
0x5366f9c3,0xc8b38e74,0xb475f255,0x46fcd9b9,0x7aeb2661,0x8b1ddf84,
0x846a0e79,0x915f95e2,0x466e598e,0x20b45770,0x8cd55591,0xc902de4c,
0xb90bace1,0xbb8205d0,0x11a86248,0x7574a99e,0xb77f19b6,0xe0a9dc09,
0x662d09a1,0xc4324633,0xe85a1f02,0x09f0be8c,0x4a99a025,0x1d6efe10,
0x1ab93d1d,0x0ba5a4df,0xa186f20f,0x2868f169,0xdc7da83,0x573906fe,
0xa1e2ce9b,0x4fcd7f52,0x50115e01,0xa70683fa,0xa002b5c4,0x0de6d027,
0x9af88c27,0x773f8641,0xc3604c06,0x61a806b5,0xf0177a28,0xc0f586e0,
0x006058aa,0x30dc7d62,0x11e69ed7,0x2338ea63,0x53c2dd94,0xc2c21634,
0xbbcbee56,0x90bcb6de,0xebfc7da1,0xce591d76,0x6f05e409,0x4b7c0188,
0x39720a3d,0x7c927c24,0x86e3725f,0x724d9db9,0x1ac15bb4,0xd39eb8fc,
0xed545578,0x08fca5b5,0xd83d7cd3,0x4dad0fc4,0x1e50ef5e,0xb161e6f8,
0xa28514d9,0x6c51133c,0x6fd5c7e7,0x56e14ec4,0x362abfce,0xddc6c837,
0xd79a3234,0x92638212,0x670efa8e,0x406000e0};

```

```

unsigned int BLOWFISH::s3[] = {

```

```

0x3a39ce37,0xd3faf5cf,0xabc27737,0x5ac52d1b,0x5cb0679e,0x4fa33742,
0xd3822740,0x99bc9bbe,0xd5118e9d,0xbf0f7315,0xd62d1c7e,0xc700c47b,
0xb78c1b6b,0x21a19045,0xb26eb1be,0x6a366eb4,0x5748ab2f,0xbc946e79,

```

0xc6a376d2,0x6549c2c8,0x530ff8ee,0x468dde7d,0xd5730a1d,0x4cd04dc6,  
0x2939bbdb,0xa9ba4650,0xac9526e8,0xbe5ee304,0xa1fad5f0,0x6a2d519a,  
0x63ef8ce2,0x9a86ee22,0xc089c2b8,0x43242ef6,0xa51e03aa,0x9cf2d0a4,  
0x83c061ba,0x9be96a4d,0x8fe51550,0xba645bd6,0x2826a2f9,0xa73a3ae1,  
0x4ba99586,0xef5562e9,0xc72fef3,0xf752f7da,0x3f046f69,0x77fa0a59,  
0x80e4a915,0x87b08601,0x9b09e6ad,0x3b3ee593,0xe990fd5a,0x9e34d797,  
0x2cf0b7d9,0x022b8b51,0x96d5ac3a,0x017da67d,0xd1cf3ed6,0x7c7d2d28,  
0x1f9f25cf,0xadf2b89b,0xad6b472,0x5a88f54c,0xe029ac71,0xe019a5e6,  
0x47b0acfd,0xed93fa9b,0xe8d3c48d,0x283b57cc,0xf8d56629,0x79132e28,  
0x785f0191,0xed756055,0xf7960e44,0xe3d35e8c,0x15056dd4,0x88f46dba,  
0x03a16125,0x0564f0bd,0xc3eb9e15,0x3c9057a2,0x97271aec,0xa93a072a,  
0x1b3f6d9b,0x1e6321f5,0xf59c66fb,0x26dcf319,0x7533d928,0xb155fdf5,  
0x03563482,0x8aba3cbb,0x28517711,0xc20ad9f8,0xabcc5167,0xccad925f,  
0x4de81751,0x3830dc8e,0x379d5862,0x9320f991,0xea7a90c2,0xfb3e7bce,  
0x5121ce64,0x774f3e32,0xa8b6e37e,0xc3293d46,0x48de5369,0x6413e680,  
0xa2ae0810,0xdd6db224,0x69852dfd,0x09072166,0xb39a460a,0x6445c0dd,  
0x586cdecf,0x1c20c8ae,0x5bbef7dd,0x1b588d40,0xccd2017f,0x6bb4e3bb,  
0xdda26a7e,0x3a59ff45,0x3e350a44,0xbcb4cdd5,0x72eacea8,0xfa6484bb,  
0x8d6612ae,0xbf3c6f47,0xd29be463,0x542f5d9e,0xaec2771b,0xf64e6370,  
0x740e0d8d,0xe75b1357,0xf8721671,0xaf537d5d,0x4040cb08,0x4eb4e2cc,  
0x34d2466a,0x0115af84,0xe1b00428,0x95983a1d,0x06b89fb4,0xce6ea048,  
0x6f3f3b82,0x3520ab82,0x011a1d4b,0x277227f8,0x611560b1,0xe7933fdc,  
0xbb3a792b,0x344525bd,0xa08839e1,0x51ce794b,0x2f32c9b7,0xa01fbac9,  
0xe01cc87e,0xbcc7d1f6,0xcf0111c3,0xa1e8aac7,0x1a908749,0xd44fbd9a,  
0xd0dade3b,0xd50ada38,0x0339c32a,0xc6913667,0x8df9317c,0xe0b12b4f,  
0xf79e59b7,0x43f5bb3a,0xf2d519ff,0x27d9459c,0xbf97222c,0x15e6fc2a,  
0x0f91fc71,0x9b941525,0xfae59361,0xceb69ceb,0xc2a86459,0x12baa8d1,  
0xb6c1075e,0xe3056a0c,0x10d25065,0xcb03a442,0xe0ec6e0e,0x1698db3b,  
0x4c98a0be,0x3278e964,0x9f1f9532,0xe0d392df,0xd3a0342b,0x8971f21e,  
0x1b0a7441,0x4ba3348c,0xc5be7120,0xc37632d8,0xdf359f8d,0x9b992f2e,  
0xe60b6f47,0x0fe3f11d,0xe54cda54,0x1edad891,0xce6279cf,0xcd3e7e6f,  
0x1618b166,0xfd2c1d05,0x848fd2c5,0xf6fb2299,0xf523f357,0xa6327623,  
0x93a83531,0x56cccd02,0xacf08162,0x5a75ebb5,0x6e163697,0x88d273cc,  
0xde966292,0x81b949d0,0x4c50901b,0x71c65614,0xe6c6c7bd,0x327a140a,

```

0x45e1d006,0xc3f27b9a,0xc9aa53fd,0x62a80f00,0xbb25bfe2,0x35bdd2f6,
0x71126905,0xb2040222,0xb6cbcf7c,0xcd769c2b,0x53113ec0,0x1640e3d3,
0x38abbd60,0x2547adf0,0xba38209c,0xf746ce76,0x77afa1c5,0x20756060,
0x85cbfe4e,0x8ae88dd8,0x7aaf9b0,0x4cf9aa7e,0x1948c25c,0x02fb8a8c,
0x01c36ae4,0xd6ebe1f9,0x90d4f869,0xa65cdea0,0x3f09252d,0xc208e69f,
0xb74e6132,0xce77e25b,0x578fdfe3,0x3ac372e6};

```

```

unsigned int BLOWFISH::p[] = {

```

```

0x243f6a88,0x85a308d3,0x13198a2e,0x03707344,0xa4093822,0x299f31d0,
0x082efa98,0xec4e6c89,0x452821e6,0x38d01377,0xbe5466cf,0x34e90c6c,
0xc0ac29b7,0xc97c50dd,0x3f84d5b5,0xb5470917,0x9216d5d9,0x8979fb1b,

//
//
0xb83acb02, 0x2002397a, 0x6ec6fb5b, 0xffcf4ddd, 0x4cbf5ed1, 0xf43fe582,
0x3ef4e823, 0x2d152af0, 0xe718c970, 0x59bd9820, 0x1f4a9d62, 0xe7a529ba,
0x89e1248d, 0x3bf88656, 0xc5114d0e, 0xbc4cee16, 0x034d8a39, 0x20e47882,
0xe9ae8fbd, 0xe3abdc1f, 0x6da51e52, 0x5db2bae1, 0x01f86e7a, 0x6d9c68a9,
0x2708fcd9, 0x293cbc0c, 0xb03c86f8, 0xa8ad2c2f, 0x00424eeb, 0xcacab452d,
0x89cc71fc, 0xd59c7f91, 0x7f0622bc, 0x6d8a08b1, 0x834d2132, 0x6884ca82,
0xe3aacbf3, 0x7786f2fa, 0x2cab6e3d, 0xce535ad1, 0xf20ac607, 0xc6b8e14f,
0x5eb4388e, 0x775014a6, 0x656665f7, 0xb64a43e4, 0xba383d01, 0xb2e41079,
0x8eb2986f, 0x909e0ca4, 0x1f7b3777, 0x2c126030, 0x85088718, 0xc4e7d1bd,
0x4065ffce, 0x8392fd8a, 0xaa36d12b, 0xb4c8c9d0, 0x994fb0b7, 0x14f96818,
0xf9a53998, 0xa0a178c6, 0x2684a81e, 0x8ae972f6, 0xb8425eb6, 0x7a29d486,
0x551bd719, 0xaf32c189, 0xd5145505, 0xdc81d53e, 0x48424eda, 0xb796ef46,
0xa0498f03, 0x667deede, 0x03ac0ab3, 0xc497733d, 0x5316a891, 0x30a88fcc,
0x9604440a, 0xceeb893a, 0x7725b82b, 0x0e1ef69d, 0x302a5c8e, 0xe7b84def,
0x5a31b096, 0xc9ebf88d, 0x512d788e, 0x7e4002ee, 0x87e02af6, 0xc358a1bb,
0x02e8d7af, 0xdf9fb0e7, 0x790e942a, 0x3b3c1aba, 0xc6ffa7af, 0x9df796f9,
0x321bb994, 0x0174a8a8, 0xed22162c, 0xcff1bb99, 0xdaa8d551, 0xa4d5e44b,
0xecdde3ec, 0xa80dc509, 0x0393eef2, 0x72523d31, 0xd48e3a1c, 0x224eb65e,
0x6052c3a4, 0x2109c32f, 0x052ee388, 0xed9f7ea9, 0x91c62f97, 0x77b55ba0,
0x150cbca3, 0x3aec6525, 0xdf318383, 0x43a9ce26, 0x9362ad8b, 0x0134140b,
0x8df5cf81, 0x1e9ff559, 0x167f0564, 0x3812f4e0, 0x588a52b0, 0xcbb8e944,

```



```

0xef5b16a3, 0x73c4eda1, 0x7dfcfeea, 0xf54bcbbe, 0x8773e3d2, 0xc531dcd0,
0x55c46729, 0x52774f3a, 0x57ca6bc0, 0x467d3a3b, 0x24778425, 0xb7991e9a,
0xdd825c26, 0xe452c8ee, 0xfcacde1e, 0x84833af3, 0x61211d03, 0x1732c131,
0xccadb247, 0xe606be8c, 0x712b39f1, 0x88b4ef39, 0x3a9fcdc5, 0xc5755169,
0x1ff6994f, 0x39829cb0, 0x11016573, 0x3343cbeb, 0x61d3d0b4, 0x44f30aef,
0xa8ae7375, 0x2a3a1c9d, 0xb4b70914, 0xd6ab250c, 0x853b7328, 0x495f948f,
0xd2a4ed8e, 0x6cf751e4, 0xc320bb75, 0xd9caa0b3, 0x8ba56262, 0x4e84b03f,
0xeea8076e, 0x74a07fe5, 0x8039e00c, 0x36ffdaf8, 0x03731358, 0xb9e671b9,
0xdac4ce1c, 0xb25b10ed, 0x4dd3d5b1, 0xfc2b480, 0x4634f579, 0x25eac400,
0xa9ac55ea, 0x728932df, 0x06041d05, 0x5d31f502, 0xc539c2e3, 0x2b89d9db,
0x5bcc0a98, 0xc05bfd6f, 0x1b250622, 0x2e21be0e, 0x60973b04, 0xecd54a67,
0xb54fe638, 0xa6ed6615, 0x981a910a, 0x5d92928d, 0xac6fc697, 0xe73c63ad,
0x456edf5f, 0x457a8145, 0x51875a64, 0xcd3099f1, 0x69b5f18a, 0x8c73ee0b,
0x5e57368f, 0x6c79f4bb, 0x7a595926, 0xaab49ec6, 0x8ac8fcfb, 0x8016cbdb,
0x8bbc1f47, 0x6982c711, 0x85c7da7a, 0x58811477, 0xcd67fad1, 0xd764d9b4,
0xc8102950, 0x5cd09da5, 0x1bb1f147, 0x95167d80, 0x0367046d, 0xaf1daca1,
0xa2247b23, 0x11301a54, 0x791d99c6, 0x7a4fb7cf, 0x277449a4, 0x09e57492,
0x35c9a57e, 0x5e7f500a, 0xb9a62a8a, 0xd5242a6b, 0xa1337859, 0x9cda3346,
0x14874047, 0x4328ba08, 0xeb81d51f, 0x3248896a, 0x8007d85d, 0x0f6e8dda,
0x8250bdaf, 0xce2ee042, 0x897ee022, 0x5f003612, 0x3ba18f90, 0x26314076,
0x7824035a, 0x3b57e2d5, 0x8e78aed1, 0xe90dc600
};

```

```
#endif // BLOWFISH_INCLUDED
```

## IPSEC VPN (OPENSWAN) ENHANCE BLOWFISH ALGORITHM:

```
/master/lib/libcrypto/libblowfish/bf_locl.h
```

```
#ifndef HEADER_BF_LOCL_H
```

```
#define HEADER_BF_LOCL_H
```

```
#undef c21
```

```

#define c2l(c,l)  (l=((unsigned long)((c)++))) , \

                l|=((unsigned long)((c)++))<< 8L, \
                l|=((unsigned long)((c)++))<<16L, \
                l|=((unsigned long)((c)++))<<24L)

/*
*/

#undef c2ln
#define c2ln(c,l1,l2,n)
{ \

    c+=n; \
    l1=l2=0; \
    switch (n) { \
        case 8: l2=((unsigned long)((--(c))))<<24L; \
        case 7: l2|=((unsigned long)((--(c))))<<16L; \
        case 6: l2|=((unsigned long)((--(c))))<< 8L; \
        case 5: l2|=((unsigned long)((--(c)))); \
        case 4: l1=((unsigned long)((--(c))))<<24L; \
        case 3: l1|=((unsigned long)((--(c))))<<16L; \
        case 2: l1|=((unsigned long)((--(c))))<< 8L; \
        case 1: l1|=((unsigned long)((--(c)))); \
    } \
}

#undef l2c
#define l2c(l,c)  ((c)+)=(unsigned char)(((l) )&0xff), \
                *((c)+)=(unsigned char)(((l)>> 8L)&0xff), \
                *((c)+)=(unsigned char)(((l)>>16L)&0xff), \
                *((c)+)=(unsigned char)(((l)>>24L)&0xff))

/*
*/

#undef l2cn
#define l2cn(l1,l2,c,n)  { \

```

```

        c+=n; \
switch (n) { \
case 8: *--(c)=(unsigned char)(((l2)>>24L)&0xff); \
case 7: *--(c)=(unsigned char)(((l2)>>16L)&0xff); \
case 6: *--(c)=(unsigned char)(((l2)>> 8L)&0xff); \
case 5: *--(c)=(unsigned char)(((l2)  )&0xff); \
case 4: *--(c)=(unsigned char)(((l1)>>24L)&0xff); \
case 3: *--(c)=(unsigned char)(((l1)>>16L)&0xff); \
case 2: *--(c)=(unsigned char)(((l1)>> 8L)&0xff); \
case 1: *--(c)=(unsigned char)(((l1)  )&0xff); \
        } \
}

/* NOTE - c is not incremented as per n2l */
#define n2ln(c,l1,l2,n)    { \
        c+=n; \
        l1=l2=0; \
switch (n) { \
case 8: l2=((unsigned long)(*--(c))))  ; \
case 7: l2|=((unsigned long)(*--(c)))<< 8; \
case 6: l2|=((unsigned long)(*--(c)))<<16; \
case 5: l2|=((unsigned long)(*--(c)))<<24; \
case 4: l1=((unsigned long)(*--(c))))  ; \
case 3: l1|=((unsigned long)(*--(c)))<< 8; \
case 2: l1|=((unsigned long)(*--(c)))<<16; \
case 1: l1|=((unsigned long)(*--(c)))<<24; \
        } \
}

/*
*/

#define l2nn(l1,l2,c,n)
{ \
        c+=n; \

```

```

switch (n) { \
case 8: *--(c)=(unsigned char)(((12) )&0xff); \
case 7: *--(c)=(unsigned char)(((12)>> 8)&0xff); \
case 6: *--(c)=(unsigned char)(((12)>>16)&0xff); \
case 5: *--(c)=(unsigned char)(((12)>>24)&0xff); \
case 4: *--(c)=(unsigned char)(((11) )&0xff); \
case 3: *--(c)=(unsigned char)(((11)>> 8)&0xff); \
case 2: *--(c)=(unsigned char)(((11)>>16)&0xff); \
case 1: *--(c)=(unsigned char)(((11)>>24)&0xff); \
        } \
}

#undef n2l
#define n2l(c,l)    (l=((unsigned long)(*((c)++)))<<24L, \
                    l|=((unsigned long)(*((c)++)))<<16L, \
                    l|=((unsigned long)(*((c)++)))<< 8L, \
                    l|=((unsigned long)(*((c)++))))

#undef l2n
#define l2n(l,c)    ((*((c)++)=(unsigned char)(((l)>>24L)&0xff), \
                    *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                    *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                    *((c)++)=(unsigned char)(((l) )&0xff))

/* This is actually a big endian algorithm, the most significant byte
 * is used to lookup array 0 */

#if defined(BF_PTR2)

/*
*/

#define BF_ENC(LL,R,KEY,Pi) (\
    LL^=KEY[Pi], \
    t= KEY[BF_ROUNDS+2 + 0 + ((R>>24)&0xFF)], \

```

```

        t+= KEY[BF_ROUNDS+2 + 256 + ((R>>16)&0xFF)], \
        t^= KEY[BF_ROUNDS+2 + 512 + ((R>>8 )&0xFF)], \
        t+= KEY[BF_ROUNDS+2 + 768 + ((R    )&0xFF)], \
        LL^=t \
    )

#elif defined(BF_PTR)

#ifndef BF_LONG_LOG2
#define BF_LONG_LOG2 2    /* default to BF_LONG being 32 bits */
#endif

#define BF_M (0xFF<<BF_LONG_LOG2)
#define BF_0 (24-BF_LONG_LOG2)
#define BF_1 (16-BF_LONG_LOG2)
#define BF_2 ( 8-BF_LONG_LOG2)
#define BF_3 BF_LONG_LOG2 /* left shift */

/*
*/

#define BF_ENC(LL,R,S,P) ( \
    LL^=P, \
    LL^= (((BF_LONG *)((unsigned char *)&(S[ 0]))+(R>>BF_0)&BF_M))+ \
        *(BF_LONG *)((unsigned char *)&(S[256]))+(R>>BF_1)&BF_M))^ \
        *(BF_LONG *)((unsigned char *)&(S[512]))+(R>>BF_2)&BF_M))^ \
        *(BF_LONG *)((unsigned char *)&(S[768]))+(R<<BF_3)&BF_M))) \
    )

#else

/*
*/

#define BF_ENC(LL,R,S,P) ( \
    LL^=P, \

```

```

LL^=(( S[ ((int)(R>>24)&0xff)] + \
        S[0x0100+((int)(R>>16)&0xff)])^ \
        S[0x0200+((int)(R>> 8)&0xff)])+ \
        S[0x0300+((int)(R  )&0xff)]&0xffffffffL \
    )
#endif

#endif

```

# APPENDIX B

## NETWORK TESTBED SPECIFICATIONS

**Table 16: Device Specification**

No	Item/type	Operating System /software	Specification
1	Softphone/ Client #1	Microsoft Windows 10 Pro 64, Version 2016  Ekiga softphone 4.0.1	Intel (R) Core™ i7-3632QM CPU, 2.20GHz, 4.00GB of DDR4 RAM and 1TB Hard disk
2	Softphone/ Client #2	Microsoft Windows 10 Pro, Version 2016  Ekiga softphone 4.0.1	Intel® Core™ i7-7700HQ. 8 GHz, up to 3.8 GHz CPU, 8.00GB of DDR4 RAM and 1TB Hard disk
3	RTP ToolBox	Microsoft Windows 10 Pro, Version 2016 RTP Packet Testing & Simulation Tools Ver 4.10.13	7th Generation Intel® Core™ i5-7400 Processor (6M Cache, up to 3.5 GHz) 8.00GB of DDR4 RAM and 1TB Hard disk
4	Wireshark IPerf/JPerf	Linux Ubuntu 16.04.2 LTS AMD64 Distro type : Debian Codename: Xenial Xerus Wireshark Win64-2.0.13 Iperf 2.0.5	Intel® Core™ i7-7700T (2.9 GHz, up to 3.8 GHz with Intel® Turbo Boost Technology, 8 MB cache, 4 cores) 8.00GB of DDR4 RAM and 2TB Hard disk
5	PackETH	Linux Ubuntu 16.04.2 LTS AMD64 Distro type : Debian Codename: Xenial Xerus PackETH 1.8.1 (Linux) Wireshark Win64-2.0.13	7th Generation Intel® Core™ i5-7400 Processor (6M Cache, up to 3.5 GHz) 4.00GB of DDR4 RAM and 1TB Hard disk
6	Router #1 (ISP)	Cisco 2911 Integrated Services Router IOS : c2900-universalk9- mz.SPA.152-2.T.bin	Memory : DDR2 ECC DRAM 512MB, Compact Plash Memory : 256MB
7	Router #2 (VLAN1)	Linux Ubuntu 16.04.2 LTS AMD64 Distro type : Debian Codename: Xenial Xerus	7th Generation Intel® Core™ i3-7100 Processor (3M Cache, up to 3.9 GHz) 4.00GB of DDR4 RAM and 1TB Hard disk
8	Router #3 (VLAN2)	Linux Ubuntu 16.04.2 LTS AMD64 Distro type : Debian Codename: Xenial Xerus	7th Generation Intel® Core™ i3-7100 Processor (3M Cache, up to 3.9 GHz) 4.00GB of DDR4 RAM and 1TB Hard disk
9	Switch #1 (VLAN1)	Cisco Catalyst 2960-S IOS : c2960-lanbasek9-mz.150- 1.SE3.bin	Cisco Catalyst 2960S-24PD-L 24 Ports, 2 Uplinks, Flash memory 64MB, DRAM 128MB
10	Switch #2 (VLAN2)	Cisco Catalyst t 2960-S IOS : c2960-lanbasek9- mz.150-1.SE3.bin	Cisco Catalyst 2960S-24PD-L 24 Ports, 2 Uplinks, Flash memory 64MB, DRAM 128MB

**Table 17: Network IP Addresses**

No	Item/type	IP Addresses	Port Interface	Description
1	Softphone/	192.168.10.20/24	GbE1-g1	VLAN1 to port1 Switch #1
2	Softphone/ Client #2	192.168.20.20/24	GbE2-g1	VLAN2 to port2 Switch #2
3	RTP ToolBox	192.168.10.30/24	eth0-g3	VLAN1 to port3 Switch #1
4	Wireshark	192.168.10.5/24	eth1-g4	VLAN2 to port4 Switch #1
5	Router #1 (ISP)		eth0-g16 eth0-g13	R1 to VLAN2 Client 1 to VLAN2
6	Router #2 (VLAN1)	192.168.20.1/24	eth1-eth4	R2 to R3
7	Router #3 (VLAN2)	192.168.10.1/24	eth0-g17	R2 to VLAN3
8	Switch #1 (VLAN1)	192.168.20.10/24	eth0-eth0	TR* to R3
9	Switch #2 (VLAN2)	192.168.10.10/24	eth2-g7 eth0-g8 eth2-g9 eth1-g10	R1 to VLAN1 Wireshark to VLAN1 RTP tools to VLAN1 TCPdump to VLAN1

**Table 18: Site-to-site IPSec tunnel**

Interface	peer	local-ip	tunnel	Description
eth0 on R2	10.10.2.1	10.10.1.1	tunnel1	esp-group ESP-IPIP ike-group IKE-IPIP local-subnet 10.0.5.10/32 remote-subnet 10.0.6.2/32 pre-shared-secret B10wf1\$h
eth0 on R1	10.10.1.1	10.10.2.1	tunnel1	esp-group ESP-IPIP ike-group IKE-IPIP local-subnet 10.0.6.2/32 remote-subnet 10.0.5.10/32 pre-shared-secret B10wf1\$h



# APPENDIX C

## OPNET CONFIGURATIONS & FIGURES



Figure 62: Application Profile Configuration

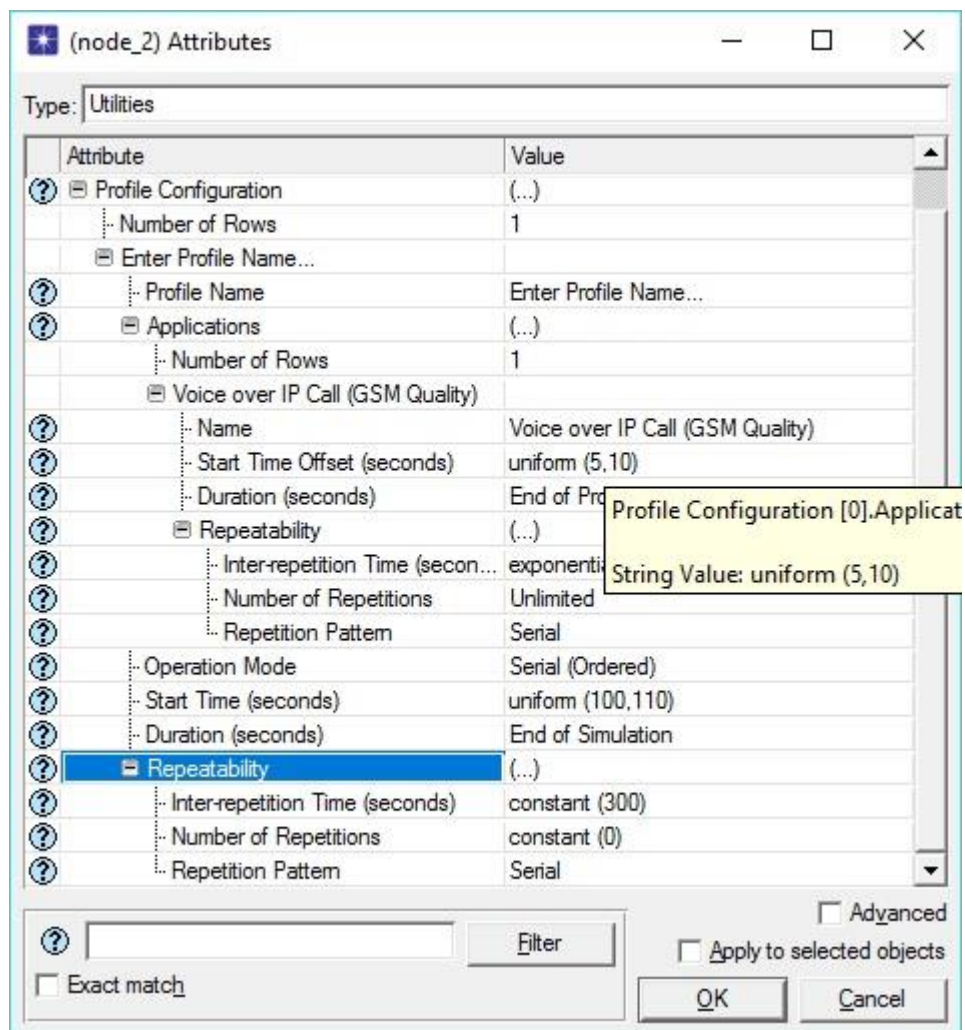
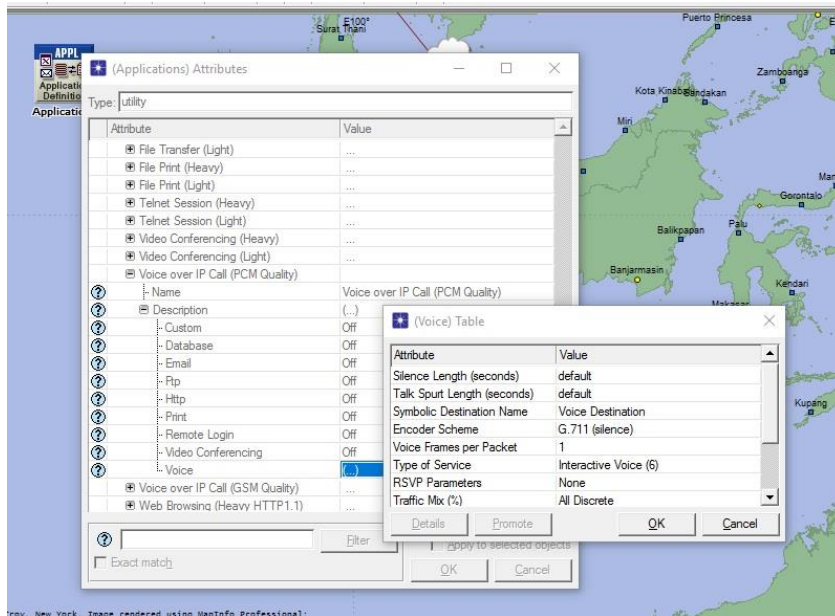


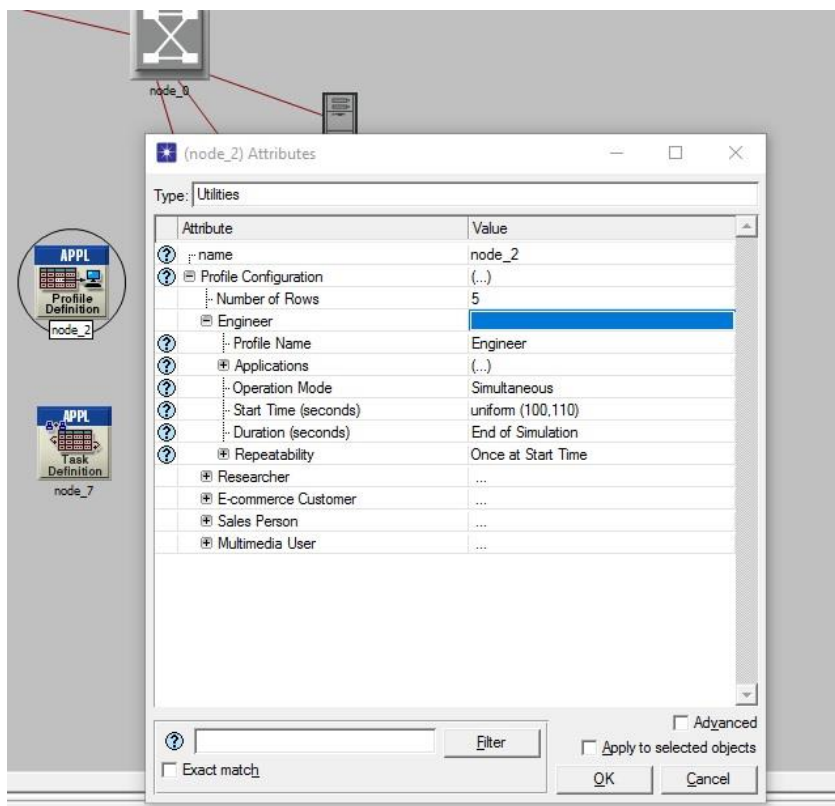
Figure 63: Application Configuration Attributes



**Figure 64: Codec Scheme**



**Figure 65: Profile Configuration**



**Figure 66: Profile Attributes**