Department of Electronics,
Telecommunications and Informatics
University of Aveiro
2008

Department of Telecommunications
VŠB – Technical University of Ostrava
2008

Saul
Parada

<smsparada@ua.pt>

IMPACTO DE MECANISMOS DE SEGURANÇA EM REDES NA TELEFONIA IP

IMPACT OF NETWORK SECURITY ON SPEECH QUALITY

Department of Electronics,
University of Aveiro Telecommunications and Informatics
2008
VŠB – Technical University of Ostrava Department of Telecommunications
2008

Saul
Parada

<smsparada@ua.pt>

IMPACTO DE MECANISMOS DE SEGURANÇA EM REDES NA TELEFONIA IP

IMPACT OF NETWORK SECURITY ON SPEECH QUALITY

Final report presented to the University of Aveiro in partial fulfillment of the requirements for the degree of Master of Science in Engineering of Electronics and Telecommunications, developed under scientific orientation of Eng. Miroslav Vozňák, PhD., and Eng. António Nogueira, PhD., professors of the Department of Telecommunications in VŠB - Technical University of Ostrava and of the Department of Electronics, Telecommunications and Informatics in University of Aveiro, respectively.

This is dedicated to my beloved grandmother Ana Rosa Lopes da Fonseca Parada.

VŠB - Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Telecommunications

Academic year 2008/2009

Diploma Thesis Assignment

Student: **Saul Manuel Sequeira Parada**

Study Programme: N2647 Information and Communication Technology

Study Branch: 2601T013 Telecommunication Technology

Title: Vliv zabezpečení sítě na kvalitu hovoru
Impact of Network Security on Speech Quality

Description:

1. Network security techniques.
2. Speech quality evaluation in IP telephony.
3. Security used with VoIP and its implementation.
4. Comparision tests of speech quality in various secure environments.
5. Evaluation achieved results.

References:

1. KONHEIM, A.*Computer Security and Cryptography.* New Jersey: JWS, 521 p., 2007. ISBN 978-0-471-94783-7
2. HARDY,W. *VoIP Service Quality, Measuring and Evaluationg Packet-Switched Voice.* New York: McGraw Hill, 2003. ISBN 0-07-141076-7
3. NAPPA,A, BRUSCHI,D., ROZZA,A.,VOZNAK,M. *Analysis and Implementation of secure and unsecure Voice over IP environment and performance comparison using OpenSER.* Published at Universita degli studi di Milano:Technical report, 84 pages, December, 2007
4. VOZNAK, M. *Impact of OpenVPN on Speech Bandwith.* .Asszisztencia Szervező Kft. Budapest: In proceedings TSP2008, 3-4.9.2008 in Paradfurdo, Hungary. ISBN 978-963-06-5487-6

Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor: **Ing. Miroslav Vozňák, Ph.D.**

Date of issue: 30.11.2008
Date of submission: 07.05.2009

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

prof. Ing. Zdeněk Diviš, CSc.
Head of Department

prof. Ing. Ivo Vondrák, CSc.
Dean of Faculty

the jury

president

Eng. Aníbal Manuel de Oliveira Duarte, PhD.
Full Professor of the Department of Electronics, Telecommunications and Informatics in University of Aveiro

examiners committee

Eng. Rui Jorge Morais Tomaz Valadas, PhD.
Full Professor of the Department of Electrical and Computer Engineering in Instituto Superior Técnico from Technical University of Lisbon

Eng. António Manuel Duarte Nogueira, PhD.
Assistant Professor of the Department of Electronics, Telecommunications and Informatics in University of Aveiro

people involved

supervisor	Eng. Miroslav Vozňák, PhD. Assistant Professor of the Department of Telecommunications in VŠB - Technical University of Ostrava
co-supervisor	Eng. António Manuel Duarte Nogueira, PhD. Assistant Professor of the Department of Electronics, Telecommunications and Informatics in University of Aveiro

“Whatever you can do or dream you can, begin it. Boldness has genius, power and magic in it.”

Johann Wolfgang von Goethe

acknowledgments

"As we enjoy great advantages from inventions of others, we should be glad of an opportunity to serve others by any invention of ours; and this we should do freely and generously."

Benjamin Franklin

In the spirit of this dictum, I would like to express my thanks to all who have joyfully contributed to this dissertation with their own "inventions" – be it ideas, technical expertise or feedback, as well as by bearing over with me during this interesting albeit intense time.

From VŠB - Technical University of Ostrava, my supervisor Eng. Miroslav Vozňák, PhD., deserves thanks for both professional and practical guidance, as well as all the friendship; to my friend Filip Řezáč for all the precious advices and good moments spent together working. My co-supervisor Eng. António Nogueira, PhD., from University of Aveiro, deserves thanks for all the support, particularly, dealing with the exchange bureaucracy making it possible.

Last, I would like to mention the most important persons in my life, my family. To all, my sincerest thanks.

keywords VoIP, security, IPsec, TLS, Openswan, OpenVPN, AES, Triple-DES, MOS and R-factor.

abstract This dissertation is about the impact of secure network environments on speech quality of IP Telephony. Results of the analysis of voice over different unsecure and secure communication links, such as IPsec and TLS, are presented. The use of secure network environments can affect speech quality. There is also a performance comparison of different cipher algorithms and a description on how the used security mechanisms influence the final MOS and R-factor variables. The presented results are based on numerous experiments which have been performed on a real IP-based network.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

3DES	Triple Data Encryption Standard
ACELP	Algebraic Code Excited Linear Prediction
ADPCM	Adaptive Differential Pulse-Code Modulation
AES	Advanced Encryption Standard
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
B2BUA	Back-to-Back User Agent
CBC	Cipher-Block Chaining
CIA	Confidentiality, Integrity and Availability
CID	Caller ID
CNG	Comfort Noise Generation
CS-ACELP	Conjugate-Structured Algebraic Code-Excited Linear Prediction
CSRC	Contribution Source
CT	Conversation Test
DES	Data Encryption Standard
DoS	Denial of Service
DDoS	Distributed Denial of Service
DS-0	Digital Signal 0
DSA	Digital Signature Algorithm
DSP	Digital Signal Processor
DWDM	Dense Wavelength Division Multiplexing
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FTP	File Transfer Protocol
GoS	Grade of Service
GSM	Global System for Mobile communications
GPL	General Public License
HMAC	Hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	Internet Protocol security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISAKMP	Internet Security Association and Key Management Protocol
ISDN	Integrated Services Digital Network

ITU	International Telecommunication Union
ITU-T	ITU - Telecommunication standardization sector
IV	Initialization Vector
MAC	Media Access Control
MC	Multipoint Controller
MCU	Multipoint Control Unit
MD5	Message-Digest algorithm 5
MDS	Maximum Distance Separable
MDI	Media Delivery Index
MKI	Master Key Identifier
MOS	Mean Opinion Score
MP	Multipoint Processor
MP-MLQ	Multi-Pulse Maximum Likelihood Quantization
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
NBS	National Bureau of Standards
NIST	National Institute of Standards and Technology
LAN	Local Area Network
LD-CELP	Low Delay Code Excited Linear Prediction
LOT	Listening Only Test
LZO	Lempel-Ziv-Oberhumer
OCSP	Online Certificate Status Protocol
OSI	Open Systems Interconnection
PAMS	Perceptual Analysis Measurement System
PBX	Private Branch eXchange
PC	Personal Computer
PCM	Pulse-Code Modulation
PESQ	Perceptual Evaluation of Speech Quality
PKCS	Public Key Cryptography Standards
PLC	Packet Loss Concealment
POTS	Plain Old Telephone Service
PSK	Pre-Shared Key
PSQM	Perceptual Speech Quality Measurement
PSTN	Public Switched Telephone Network
QoE	Quality of Experience
QoS	Quality of Service
QoSE	Quality of Service Experience
RC#	Rivest Cipher #
RSA	Rivest Shamir Adleman
RTCP	Real-time Transport Control Protocol
RTD	Round-Trip Delay
RTP	Real-time Transport Protocol
RTT	Round-Trip Time

SCN	Switched Circuit Network
SHA	Secure Hash Algorithm
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNR	Signal-to-Noise Ratio
SPIT	Spam over Internet Telephony
SRP	Secure Remote Password protocol
SRTP	Secure Real-time Transport Protocol
SS7	Signaling System number 7
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TDM	Time-Division Multiplex
TLS	Transport Layer Security
ToS	Type of Service
TTL	Time-To-Live
WAN	Wide Area Network
UA	User Agent
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VAD	Voice Activity Detection
VoIP	Voice over Internet Protocol
VoIPsec	Voice over Internet Protocol security
VPN	Virtual Private Network

1 INTRODUCTION

1.1. Motivation

The global widespread growth of Telecommunications Networks, Internet Industry and established Telecommunications Industry has increased the interest and demand on different applications. Voice over Internet Protocol (VoIP), also referred as IP Telephony, reveals itself as an important revolutionary technology which has become a potential alternative to the traditional telephony systems over the Public Switched Telephone Network (PSTN), providing a versatile and flexible solution to speech communications.

VoIP is the family of technologies that allows IP networks to be used for voice applications, such as telephony, voice instant messaging and teleconferencing. VoIP defines a way to carry voice calls over an IP network, including the digitization and packetization of the voice streams. IP Telephony VoIP standards create a telephony system where higher-level features such as advanced call routing, voice mail and contact centers can be utilized. VoIP services convert voice into a digital signal that travels over an IP-based network [1].

One of the big misconceptions about VoIP telephony applications is that they are inherently insecure. As VoIP is a packet-switched based technology that works through IP networks, it suffers, consequently, from some IP-based networks vulnerabilities.

However, in truth, the VoIP technology family provides scores more security options than conventional telephones do. If anything is insecure, it is the old voice paradigm. IP Telephony security boils down to three risk factors: the application, the network operating systems and the infrastructure. Secured these and it is secured the VoIP network [2].

Quality of Service (QoS) is a subject of crucial importance in IP Telephony and the major issue for the VoIP implementations success.

Real-time applications, such as voice applications, have different characteristics and requirements from those of traditional data applications. Because they are real-time based, voice applications tolerate minimal variation in the amount of delay affecting delivery of their voice packets. Voice traffic is also intolerant of packet loss and jitter, both of which unacceptably degrade the quality of the voice transmission delivered to the recipient end user. To effectively transport voice traffic over IP, mechanisms are required that ensure reliable delivery of voice packets [1]. IP Telephony QoS forwards instinctively to Speech Quality.

To ensure VoIP as an acceptable replacement for standard PSTN telephony services, customers must receive the same consistently high quality of voice transmission they receive with basic telephone services and an assured privacy, i. e., security.

The implementation of various security measures can cause a marked deterioration in QoS. The use of secure network environments and mechanisms can affect Speech Quality [3].

Nowadays, the way of security implementations on IP Telephony affect QoS and, consequently, the Speech Quality is a vast field of study and it is in constant development.

1.2. Objectives

This dissertation is about the impact of secure environment on speech quality of IP Telephony.

The main objective of this dissertation is comprehend and relate how secure network environments and mechanisms can affect the Speech Quality, understanding and relating the final Mean Opinion Score (MOS) and R-factor variables, on IP Telephony. There are

presented the results of the analysis of voice over different secure communication links, such as Internet Protocol Security (IPsec) and Transport Layer Security (TLS). The using of secure network environments can affect the speech quality. There is the performance comparison of cipher algorithms and description how the used security mechanisms influence the final MOS and R-factor variables. The presented results are based on numerous of experiments which have been performed in real IP-based networks.

Another objective, as important as the first one, is to obtain the academic Master degree of Engineering of Electronic and Telecommunications.

1.3. Document Outline

The report is divided into nine chapters. This subsection is a roadmap description of the document structure and a small exhibition of what will be mentioned throughout each chapter:

Chapter **1** — INTRODUCTION: This chapter presents a brief introduction to the dissertation and its objectives. It also includes the document structure to provide an easier search.

Chapter **2** — STATE-OF-THE-ART: This chapter shows the study of the State of the Art related to VoIP test and monitor tools, security applications and network analyzers.

Chapter **3** — OVERVIEW OF VOIP: This chapter presents the VoIP protocols, describing its concepts and used technology, as well as its inherent risks and vulnerabilities.

Chapter **4** — EVALUATION OF SPEECH QUALITY IN IP TELEPHONY: This chapter presents the QoS associated to VoIP, and a description of the Audio Quality Measurement techniques and its expectations.

Chapter **5** — NETWORK SECURITY TECHNIQUES: This chapter deals about the central Internet Security issues. It is also presented a brief overview about modern cryptography, as well as security mechanisms, such as IPsec and TLS.

Chapter **6** — USED TECHNIQUES OF MEASUREMENT: In this chapter, the different paths that were taken to reach the final goals are explained step-by-step.

Chapter **7** — BANDWIDTH REQUIREMENTS: Throughout this chapter is presented and explained the equations of required bandwidth in several unsecure and secure network environments, take into account factors as the used codec, timing, payload size, number of concurrent calls and so on. A concise comparison about the bandwidth requirements between unsecure and distinct secure connections is presented.

Chapter **8** — ACHIEVED RESULTS: This chapter presents the achieved results of the explained procedures throughout chapter **6**.

Chapter **9** — CONCLUSIONS AND FUTURE WORK: This chapter presents the main conclusions about the research of how security mechanisms/implementations may affect VoIP speech quality, and about the VoIP simulation tests achieved results. It is presented a brief comparison between the expected (theoretical) and the practice obtained results for diverse circumstances. There are presented the topics for future investigation.

2 STATE-OF-THE-ART

2.1. Introduction

Since the arrival of VoIP, many companies have developed and are still developing different approaches to test and monitor VoIP traffic. Most of the existing solutions don't provide a solution that combines real-world simulation with monitoring.

Concerning to network security solutions, actually, there are a few skilled and stable applications, and among them some are free.

The next subsections will briefly present some of the existing solutions for simulate, test and monitor VoIP traffic, as well as some security applications and network analyzers/packet sniffers.

2.2. VoIP Test and Monitor Solutions

IxChariot

IxChariot [4], produced by IXIA, is a test tool for simulating real-world applications to predict device and system performance under realistic load conditions. Comprised of the IxChariot Console, Performance Endpoints and IxProfile, the IxChariot offers network performance assessment and device testing by simulating hundreds of protocols across thousands of network endpoints.

IxChariot provides the ability to confidently assess the performance characteristics of any application running on wired and wireless networks. IxChariot is a reliable and useful tool to determine VoIP QoS by measuring throughput, jitter, packet loss, end-to-end delay, MOS and Media Delivery Index (MDI) variables.

Figure 1: IxChariot.

VQmon/EP

VQmon/EP [5], a product from Telchemy, is specifically designed for integration into VoIP endpoints, such as media gateways, IP phones, traditional TDM gateways and hybrid IP/TDM systems. VQmon/EP monitors voice calls and produces call quality estimates that can be reported as MOS scores and R-factors through the media path using RTCP XR (RFC 3611), end of call signalling or SNMP. VQmon/EP is small (630 Kbytes) and highly efficient (<500 instructions per second). VQmon/EP detects packet loss and jitter buffer discard events, extracts key information from DSP software and produces call quality scores and diagnostic data. VQmon generates listening and conversational quality MOS scores, R-factors and a wide range of diagnostic data, making these available through an application

programming interface (API) as raw metrics, RTCP XR and Session Initiation Protocol (SIP) QoS Report payloads. VQmon is based on the ITU E-model with many extensions to improve accuracy under time varying network conditions, wideband codecs, orthogonal impairments and signal related parameters.

SIPp

SIPp [6] is a performance testing tool for the SIP protocol, which is released under the GNU General Public License [7]. It includes a few basic SipStone user agent scenarios and establishes and releases multiple calls with the INVITE and BYE methods. It can also read XML scenario files describing any performance testing configuration. It features the dynamic display of statistics about running tests (call rate, round trip delay and message statistics), periodic CSV statistics dumps, TCP and UDP over multiple sockets or multiplexed with retransmission management, regular expressions and variables in scenario files, and dynamically adjustable call rates.

SIPp can be used to test many real SIP equipments like SIP proxies, B2BUAs, SIP media servers, SIP/x gateways, SIP PBX and so on. It is also very useful to emulate thousands of user agents calling to a SIP system.

2.3. Security Applications

Cisco Systems

Cisco Systems [8] is a multinational corporation, headquartered in San Jose, California. It designs and sells networking and communications technology and services.

Concerning to network security environment, Cisco Systems has always a word to declare. Cisco Systems provides network security solutions based on Virtual Private Networks (VPNs).

VPNs provide high levels of security through encryption and authentication technologies that protect data from unauthorized access. There are two types of Cisco encrypted VPNs: site-to-site and remote-access [9].

Site-to-Site VPNs

Site-to-site VPNs are an alternative to Frame Relay or leased-line WANs. All traffic between sites is encrypted using IP Security (IPsec). Site-to-site VPNs are also used to increase the security of other WAN technologies, such as Multiprotocol Label Switching (MPLS) and Frame Relay through data encryption and authentication.

Remote-Access VPNs

Remote-access VPNs are a flexible and cost-effective alternative to private dialup solutions; in fact, VPNs have become the primary solution for remote-access connectivity. Remote-access VPNs extend almost any data, voice or video application to remote working locations, helping to create a user experience that emulates working in the main office location. All traffic between the user desktop and the office site is encrypted. Remote-access VPNs may be deployed using Secure Sockets Layer (SSL) VPN, IPsec, or both, depending on deployment requirements

IPsec-Tools

IPsec-Tools [10] is a port of KAME's IPsec utilities to the Linux-2.6 IPsec implementation. It supports NetBSD and FreeBSD as well. IPsec-Tools includes libipsec (a library with a PF_KEY implementation), setkey (a tool for manipulating and dumping the kernel Security Policy Database and Security Association Database) and racoon (Internet Key Exchange daemon for automatically keying IPsec connections).

Openswan

Openswan [11] is a complete IPsec implementation for Linux. It supports the 2.0, 2.2, 2.4 and 2.6 Linux kernels. Openswan is a VPN implementation based on the industry standard IPSec protocol (RFC 2401).

Openswan began as a fork of the now-defunct FreeS/WAN Project [12] and continues to be released freely under the GNU GPL [7]. Unlike the FreeS/WAN Project, it is not developed exclusively for the Linux operating system.

strongSwan

strongSwan [13] is a complete IPsec implementation for Linux 2.4 and 2.6 kernels. It is a descendant of the FreeS/WAN project and continues to be released under the GNU GPL [7]. The maintainer of the strongSwan project is Andreas Steffen. The focus of the strongSwan project is on strong authentication mechanisms using X.509 public key certificates and optional secure storage of private keys on smartcards through a standardized PKCS #11 interface. It supports certificate revocation lists and the Online Certificate Status Protocol (OCSP). A unique feature is the use of X.509 attribute certificates to implement advanced access control schemes based on group memberships.

strongSwan has an easy and straightforward approach to configuration and interoperates smoothly with most other IPsec implementations.

OpenVPN

OpenVPN [14] is a full-featured open source SSL VPN solution that accommodates a wide range of configurations, including remote access, site-to-site VPNs, Wi-Fi security, and enterprise-scale remote access solutions with load balancing, failover, and fine-grained access-controls. OpenVPN is a free and open source Virtual Private Network program for creating point-to-point or server-to-multiclient encrypted tunnels between host computers. It is capable of establishing direct links between computers that are behind NAT firewalls without requiring reconfiguration. Starting with the fundamental premise that complexity is the enemy of security, OpenVPN offers a cost-effective, lightweight alternative to other VPN technologies that is well-targeted for the small and medium enterprises and enterprise markets.

The OpenVPN security model is based on SSL, the industry standard for secure communications via the internet. OpenVPN implements OSI layer 2 or 3 secure network extension using the SSL/TLS protocol, supports flexible client authentication methods based on certificates, smart cards, and/or 2-factor authentication, and allows user or group-specific access control policies using firewall rules applied to the VPN virtual interface.

OpenVPN was written by James Yonan and is published under the GNU GPL [7].

2.4. Network Analyzers

Observer

Observer [15], created by Network Instruments, is a test tool that includes many robust capabilities to optimize network availability, efficiency and performance. Observer encompasses enterprise-strength Voice over IP analysis, a time-based interface for examining up to eight Terabytes of data and the ability to pinpoint transaction delay through up to 10 conversation hops. Observer is also a multi-topology, distributed analyzer. Observer includes significant VoIP advancements. The VoIP Expert feature offers aggregate statistics for overall VoIP traffic, call summary, and quality scoring, as well as over 20 detailed per-call metrics including call status, current jitter, call setup, duration, teardown, MOS/R-factor and QoS prioritization. Observer VoIP analysis is available across multiple topologies (LAN, WAN, gigabit and 802.11a/b/g).

Iperf/JPerf

Iperf [16], developed by NLANR/DAST, is a open source tool for measuring maximum TCP and UDP bandwidth performance. Iperf allows the tuning of various parameters and UDP characteristics. Iperf has a client and server functionality, and can measure the throughput between two ends, either uni-directionally or bi-directionally. Iperf reports parameters as bandwidth, delay jitter and datagram loss. Iperf is significant as it is a standardized tool that can be run over any network and output standardized performance measurements.

JPerf [17] is an open source Java-based tool that performs all the functions as Iperf. JPerf is just a frontend to Iperf, so it performs the same tests and is interoperable with Iperf. It is even possible run JPerf as one end point and Iperf on the other.

Wireshark

Wireshark [18] is a free packet sniffer computer application. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark is a network protocol analyzer that "understands" the structure of different networking protocols and it allows the user to see all traffic being passed over the network by putting the network interface into promiscuous mode. Thus, it is able to display the encapsulation and the fields along with their meanings of different packets specified by different networking protocols.

2.5. Conclusions

After a market study for solutions capable of meeting this dissertation needs, concerning to simulation, test and monitor VoIP traffic tool, our choice drooped to the IXIA product, IxChariot, due the fact it combines a cost-effective test tool for emulating real-world applications, such as specific VoIP traffic, to predict device and system performance under realistic load conditions with the feature to determine VoIP QoS, avoiding thus the use of more than one application to perform the test and achieve the VoIP QoS variables.

Related to network secure environment, the open source Openswan and OpenVPN appear to be reliable and powerful applications, sharing the fact they are free solutions. Table **1** shows a brief comparison and overview between Openswan and OpenVPN implementation solutions. It should be noted that the SIPp tool also supports TLS.

From all the looked over network analyzers, the open source Wireshark application pleased a lot, because beyond be free is a powerful tool relatively to examine network traffic, particularly the VoIP one. Also related with the measurement of some network parameters and keeping the open source spirit, the conjugation of Iperf and JPerf tools reveals itself as a powerful mechanism to achieve variables as bandwidth, delay jitter and datagram loss.

Table 1: Openswan versus OpenVPN.

Implementation	**Openswan**	**OpenVPN**
Protocol	IPsec	SSL/TLS
Authentication	RSA, Shared Key	RSA, Shared Key
Cipher	DES, 3DES, AES	Blowfish, AES, 3DES,+
NAT	Yes	Yes

3 OVERVIEW OF VOIP

3.1. Introduction

In terms of flexibility, cost, variety of products and services, networks and solutions for communications (voice, data and images) based on packet switching and IP protocol are becoming increasingly competitive when compared to traditional offerings like Time-Division Multiplexing (TDM), Frame Relay or Asynchronous Transfer Mode (ATM). There is a growing demand for both simple access to services for Virtual Private Networks (VPNs) carrying voice, data and images. VoIP is a transmission technology for packet-switched networks, the most recent step of telephony evolution. It transports Voice over Internet Protocol packets through an IP network. Each packet may travel with a different route in the transport network, as there is no single reserved path. As a consequence of the underlying unreliable protocol, User Datagram Protocol (UDP), packets arriving at the destination may come in a different sequence than they were sent and there is no guaranteed bandwidth.

Originally, VoIP was supposed to provide only PSTN services, like voice calls, faxes or mailboxes. However, the transmission of voice over IP networks presents several additional possibilities.

3.2. Concepts

The architecture of Internet Telephony is identical to old fashioned telephone networks in many ways, but it also has some significant differences. Fundamentally, Internet Telephony runs over IP networks. The most significant consequence of having this underlying network is that it provides transparent connectivity between any devices on the network, independently of the location. Whereas devices in traditional networks are restricted by communicating with those devices to which they are directly connected, Internet Telephony can rely on an underlying infrastructure which provides these capabilities automatically.

There are four basic elements in a VoIP system:

Terminal

Terminal is a communication endpoint where a call is generated and where is terminated. It is where the end user resides and some automatic interaction is also possible (voicemail). It can be software or hardware based.

Server

Server is the system's central point. Terminals are registered here and their information (location, IP) is stored. It provides routing mechanisms for the call and sets up authentication and performing for accounting operations.

Gateway

Gateway is a border element of a VoIP network. It provides inter-operability between IP and PSTN networks.

Conference Bridge

Conference Bridge provides functionality for multi-point communication. It is separated from the server, as it demands high resource requirements.

In order for VoIP to work, these devices have to interact between them in many ways. Initially, at the source, analogue input (voice) is converted into digital signals, occurring a speech compression. Pulse-Code Modulation (PCM) can be the digital representation of that analogue signal where the magnitude of the signal is sampled regularly at uniform intervals, then quantized to a series of symbols in a numeric (usually binary) code. After converting voice packets, Real-Time Transport Protocol (RTP) is used for time stamping and content identification of User Datagram Protocol voice packets. Then, after negotiating all initial settings for transmitting, the packets are sent. However, this becomes more complex if the signalling system has to communicate with the gateway that is located between the Internet and PSTN. In case of outgoing calls, the VoIP phone captures the phone number and the IP address of the gateway. For incoming calls, the gateway has associated a telephone number with the device's IP address and forwards the calls to it. Finally, at the receiving end, packets have to be disassembled for data extraction in order to put the voice data into the device's sound card.

A concise describing of the used protocols in a VoIP call is presented in the subsection **3.3**.

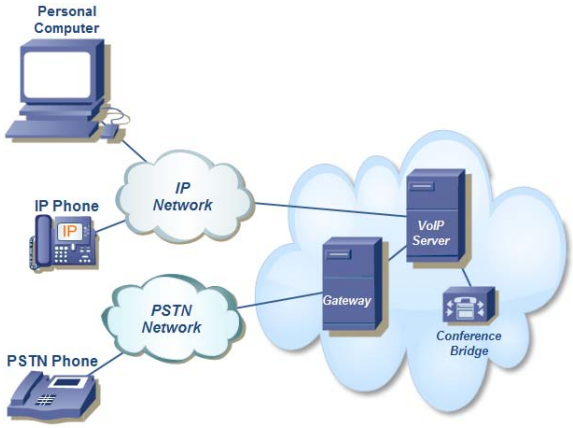

Figure 2: VoIP system scheme.

3.3. VoIP Implementation Details

This section is related to the various protocols used in VoIP systems. These break down into two main categories: protocols used for setting up calls and building a VoIP infrastructure, and protocols for transporting the speech data over the Internet. While the focus of this dissertation is on speech quality, the call setup and teardown protocols are discussed to complement.

3.3.1. Call Setup Protocols

One of the most important needs for VoIP systems was a well-defined and accepted protocol for managing the VoIP infrastructure. Since VoIP devices are associated with an IP address that may change over time (if a laptop user plugs into the Internet from two different locations, for example), there must be a dynamic process of associating a particular user with an IP address. Additionally, there needs to be an established signalling process that

allows users to call each other, be connected, receive busy signals and leave voice-mail; anything users expect from traditional phones should be supported by VoIP. Finally, there should be a well-defined way to interact between VoIP and traditional PSTN telephones and networks. Several protocols have been introduced to accomplish these tasks, with the two most important ones being H.323 and Session Initiation Protocol (SIP).

H.323

H.323 is an International Telecommunication Union Telecommunication Standardization Sector (ITU-T) specification for transmitting audio, video and data across an IP-based network. When compliant with H.323, products and applications can communicate and interoperate with each other. The H.323 standard addresses call signalling and control, multimedia transport and control, and bandwidth control for point-to-point and multipoint conferences. The H series of recommendations also specifies H.320 for Integrated Services Digital Network (ISDN) and H.324 for plain old telephone service (POTS), also referred as PSTN, as transport mechanisms [19]. It was the first widely used standard for VoIP.

The H.323 standard consists of the following components and protocols:

Table 2: H.323 components and protocols [19].

Feature	Protocol
Call Signalling	H.225
Media Control	H.245
Audio Codecs	G.711, G.722, G.723, G.728, G.729
Video Codecs	H.261, H.263
Data Sharing	T.120
Media Transport	RTP/RTCP

There are four basic components in an H.323 system: terminal, gateway, gatekeeper, and multipoint control unit [2].

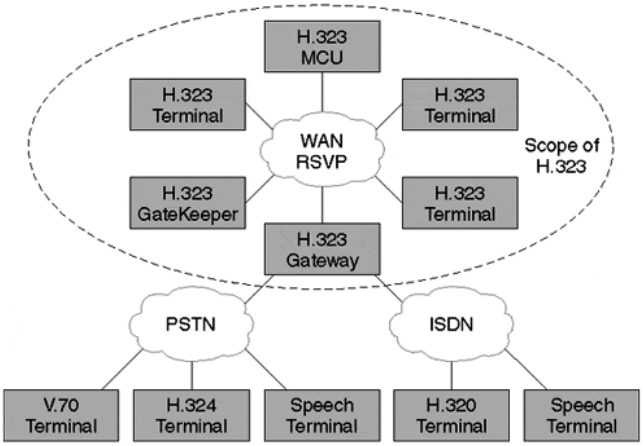

Figure 3: Elements of an H.323 network [19].

The terminal is the endpoint device that provides a user interface to the system. Terminals can be a dedicated piece of hardware known as an IP phone, a software program running on a personal computer (PC) or a normal telephone running through a network adaptor.

The H.323 gateway reflects the characteristics of a Switched Circuit Network (SCN) endpoint and H.323 endpoint. It translates between audio, video and data transmission

formats, as well as communication systems and protocols. This includes call setup and teardown on both the IP network and SCN. Gateways are maintained by a VoIP provider or by a company to allow their users to interact with other communication networks.

The third H.323 device, the gatekeeper, provides call control, bandwidth management and address translation for connections between H.323 endpoints. Calls are initiated through gatekeepers, which are responsible for translating a phone number to an IP address and also may monitor call times for billing and tracking purposes. Gatekeepers, as well, are maintained by a provider or corporation and users simply route calls through them.

The final device, known as a Multipoint Control Unit (MCU), enables three or more terminals or gateways to establish a multipoint conference. The MCU consists of a Multipoint Controller (MC) and one or more Multipoint Processor (MP). The MC supports conferences between three or more endpoints in a multipoint conference. MCs transmit the capability set to each endpoint in the multipoint conference and can revise capabilities during the conference. The MC function can be resident in a terminal, gateway, gatekeeper or MCU. The MP receives audio, video and/or data streams and distributes them to endpoints participating in a multipoint conference.

A major drawback of H.323 is that many control messages and protocols are used to initiate and terminate calls. *Figure 4* shows the control messages required to setup an H.323 call between two parties. It can be seen that H.323 relies on several other protocols including H.225 (to set up the connection) and H.245 (to determine the capabilities of each user's terminal). Additionally, connections on two separate ports are required for the setup of a single call.

Another criticism of H.323 is that the protocol is binary-encoded, i. e., it encodes information in numbers, as opposed to text. This encoding is frustrating for programmers debugging H.323 applications because it is difficult to easily observe what and where the problems are. The binary encoding also makes H.323 less extensible, as information must be contained in a very specific part of the packet with a fixed size. For these reasons many people have recently moved away from H.323 and accepted SIP as the standard for VoIP.

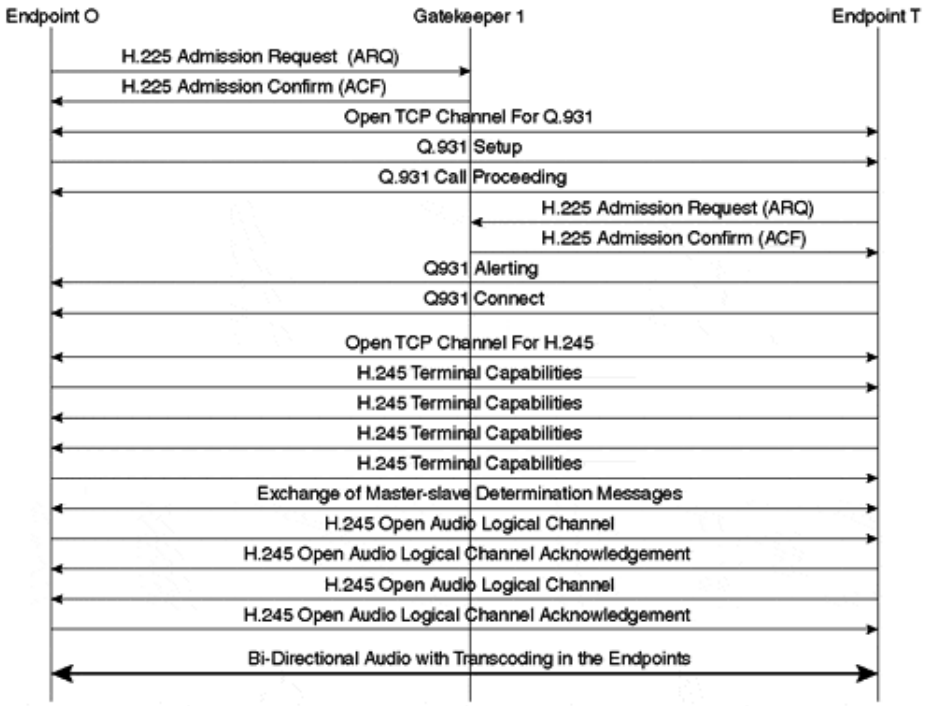

Figure 4: H.323 direct endpoint signalling (same Gatekeeper) [19].

Session Initiation Protocol

Session Initiation Protocol [20], commonly known as SIP, is the accepted standard for multimedia Internet applications, including VoIP, for the Internet Engineering Task Force (IETF). SIP is a signalling protocol that controls the initiation, modification and termination of interactive multimedia sessions. The multimedia sessions could be as diverse as audio or video calls among two or more parties, chat sessions or game sessions. SIP extensions have also been defined for instant messaging, presence and event notifications. SIP is a text-based protocol that is similar to HTTP [21] and Simple Mail Transfer Protocol [22] (SMTP) [19].

In SIP each user is associated with an address known as a uniform resource indicator (URI). The URI (sometimes called a SIP URI or SIP address) is analogous to uniform resource locators (URLs) of websites, with the key difference that they are meant to be dynamic [23]. URI's are not meant to be tied to a particular physical device, but to a logical entity that might move or exist in multiple places.

A SIP architecture has several similarities to an H.323 architecture. Two similar elements are SIP user agents (UAs) and SIP gateways.

A SIP UA is analogous to an H.323 terminal; it is a hardware or software device that allows a user to make VoIP calls using SIP. SIP UAs can take the form of dedicated hardware telephones, network adapted analogue telephones or software applications running on an Internet-enabled PC.

SIP gateways are also analogous to H.323 gateways; they provide translation between protocols. Two common gateways are SIP/PSTN gateways, which allow SIP devices to interact with traditional phone systems, and SIP/H.323 gateways, which interact SIP and H.323 devices together. Like H.323 gateways, these are part of the infrastructure backbone and are generally maintained by service providers or corporations.

SIP architecture also requires a number of SIP servers, devices that handle SIP messages and each one serves a different function. The three types of SIP servers are proxy servers, redirect servers and registration servers.

Proxy Servers

The role of a proxy server is to handle SIP messages on behalf of SIP user agents. Proxy servers usually have access to a database or location service to help determine what to do with the request. For example, a UA might try to initiate a call to a person whose SIP URI is alice@company.com. The UA sends a SIP INVITE request to a proxy and the proxy attempts to determine the IP associated with Alice's address by querying its database or location service. The proxy server then forwards the request to wherever it determines the best location for Alice is, or responds with a "not found" error message. Proxies, in general, do not create new messages, they merely forward and respond to requests as they see fit.

Redirect Servers

A redirect server is a SIP server that responds to requests, but never forwards them. Like the proxy server, it usually queries a database or location service to determine an appropriate response, but unlike the proxy it will never forward a message. Redirect servers usually inform the requesting UA of the location of someone, but leave it up to the UA to contact that location on its own.

Registration Servers

A registration server, or registrar, only accepts one type of SIP message, a SIP REGISTER request. The registrar then keeps the state of the users registered to it for a particular domain, allowing the proxy and redirect servers to query it for location information. Registration servers usually perform user authentication, although this isn't required. Authentication ensures that only valid SIP users in the registrar's domain are registered and serviced, and prevents outside users from placing and receiving calls through another domain's servers.

One advantage to SIP is that it is a text-encoded protocol. This means that the information is sent over the Internet as human-readable text. The advantage to a text-based protocol is that it is much easier to program, analyze and debug. A simple traffic sniffing tool can be used on the network to easily understand what information is being sent - a task not nearly as straightforward in H.323 systems.

Another advantage is that SIP call setup is quite simple, as can be seen in *Figure 5*. This figure, when compared to *Figure 4*, represents all the signalling required for a complete call (*Figure 4* is only an H.323 setup) and shows a call being routed through two SIP proxy servers. SIP calls only require the establishment of a single TCP connection and SIP can even bypass TCP and run entirely over UDP. In SIP, all of the codec negotiation is done in the INVITE and OK messages, bypassing the need for the series of H.245 "terminal capabilities" messages seen in *Figure 4*.

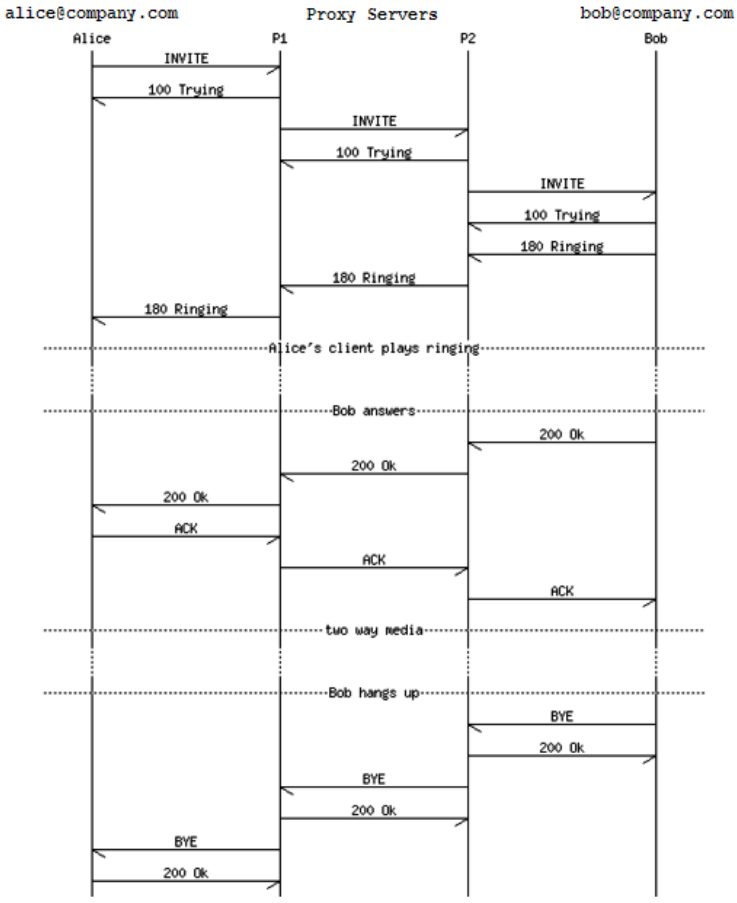

Figure 5: SIP call Setup and Takedown.

SIP versus H.323: A Comparison

H.323 is older than SIP and more considerate of legacy infrastructure, too. It defines standard procedures and best practices for interfacing with old-school telephony technology practices like the use of gateways with built-in support of legacy protocols, like ISDN. Table **3** shows a brief comparison between SIP and H.323 protocols.

Table 3: Comparison between SIP and H.323 protocols [2].

Function/ Characteristic	SIP	H.323
Endpoint discovery and admission	SIP REGISTER methods	SSL/ RAS Protocol
Call setup and teardown	SIP INVITE methods	H.225 Protocol
Capabilities negotiation, codec selection, and media session port selection	Session Definition Protocol	H.245 Protocol
Packetization and sound sample transmission	RTP/RTCP Protocol	RTP/RTCP Protocol
Streaming of recorded audio and video	RTSP	None recommended
Frame encoding	Text (ASCII & similar)	ASN.1
Messaging approach	HTTP-like	ISDN-like / Q.931
SoftPBX call path is called a...	Proxy	Gatekeeper-routed
Call-routing reference device is called a...	Registrar	Gatekeeper
Independent call path is called...	Redirect	Directed signaling
PSTN interface approach	None recommended	H.323 gateway
Encryption of signalling messages	TLS/TCP	None recommended
Endpoint identification	SIP URI, email address, E.164 address, or alias	E.164 address
Connections through firewalls	Gatekeeper/softPBX call path	Proxy/softPBX call path
Multiplexed trunks	Yes	Yes
UDP port number	5060/5061	1503,1720,1731

Others

SIP and H.323 are not the only protocols that have been proposed for IP Telephony [2]. Others include Megaco [24], MGCP [25] and Skinny. While these protocols are used in some areas today, currently, SIP and H.323 are the standards for VoIP, so a detailed discussion of these protocols is not provided here.

3.3.2. Transport Protocols

While SIP and H.323 perform call setup and takedown signalling, they do not provide any means of actually transmitting information over the Internet, nor do they handle the media streams containing the actual media data. For this task, several existing protocols are employed, including IP, UDP, TCP and RTP. Both SIP and H.323, for example, must run over either TCP or UDP. Additionally, the audio data of the conversation uses RTP running over UDP for delivery. Finally, all of these protocols run over IP. An explanation of what these different transport protocols provide is found below.

Internet Protocol

Internet Protocol (IP) is used to route packets across a data network, such as the Internet. It provides connectionless and best-effort packet delivery, meaning that packets might be lost, delayed, arrive out of sequence or contain errors.

Each location on the network is associated with a particular address, known as an IP address. The current most-employed standard for IP addresses is the IP version 4 [26] (IPv4), in which addresses consist of four numbers between 0 and 255, separated by periods. 123.45.67.8 is an example of an IPv4 address, as is 11.0.0.255. A new version of IP, IP version 6 [27] (IPv6) was developed by the IETF to allow for longer addresses, because the IP address space is rapidly running out [28]. The increased address space provided by IPv6 is necessary; for example, in case of every telephone in the world will have a unique IP address. IPv6 is an important emerging technology, however, is largely outside of the scope of this dissertation and all performed experiments were using the most common IPv4 protocol.

IP addresses are assigned by the Internet Assigned Number Association [23] (IANA) and they are globally unique. This ensures consistency; packets destined for a particular unicast address should always arrive at one and only one unique machine. Any IP packet with a specific destination IP address, sent to any router on the Internet, should find its way to that IP device.

User Datagram Protocol

User Datagram Protocol [29] (UDP) provides a small step up into the complexity of IP. It is also a connectionless and best-effort protocol, meaning that packets can be lost or dropped, but it provides a checksum, allowing errors to be detected. Additionally, UDP specifies not only an IP address, but also a port. Certain communications run over specific, well-known ports (for example, HTTP uses port 80 and SIP uses port 5060). In this way, UDP allows multiple logical channels of communication to exist between two computers. In the case of VoIP, SIP could be handling the call setup over port 5060, while the actual data streams are travelling over some other (usually determined at call-setup) port.

Transmission Control Protocol

Transmission Control Protocol [30] (TCP) also runs over IP, but provides the reliability and guaranteed delivery of data that IP and UDP do not. It does this by using acknowledgements and sequence numbers. Every TCP packet that is sent has a header representing which bytes of the overall stream of data the packet represents. This header allows the receiver to recognize when information is missing and ask the sender to resend. It also allows the receiver to reconstruct information in the correct order, even though packets may arrive out of order.

TCP is a good transport protocol to use when guaranteed and reliable delivery is required. Most traffic, including World Wide Web, e-mail and File Transfer Protocol (FTP) use TCP. A problem with TCP, however, is that it can make things slow. Retransmissions takes time and in live streaming applications, like VoIP, a delay as small as a second can severely detract the conversation quality. Additionally, TCP has problems with long latencies, since it drastically reduces the transmission rate when packets are lost. For these reasons, TCP is generally not used for media streams, although it can be used for call-setup. As mentioned previously, currently, SIP and H.323 support delivery over both TCP and UDP.

Real-time Transport Protocol

VoIP and other media streaming applications are an exclusive type of traffic with very specific requirements. More than any other service these applications are extremely time-sensitive [19]. Delays larger than a few fractions of a second are simply unacceptable. Fortunately, these applications do not need guaranteed delivery of every packet. Audio and video applications can usually interpolate a value for a missing data point and end-users

likely will not notice a loss in quality. A final requirement of these applications is that ordering should be preserved. Moreover, many voice and video codecs use differences rather than absolute values to encode data and a reordering of information can make it completely useless. To summarize, VoIP requires a quick delivery of data along with ordering information, but doesn't require guaranteed delivery. Real-time Transport Protocol [31] (RTP) was developed to meet these characteristics.

RTP runs over UDP and, therefore, is a connectionless, best-effort protocol. This ensures the fastest possible delivery over the Internet and also means that some packets will be dropped or arrive out of order. In addition to the UDP checksum, which alerts the receiver if the data in the packet contains errors, it also provides several multimedia-specific features.

One of these features is sequencing. As mentioned above, sequencing is important in streaming applications. Typical VoIP applications actually hold on to packets for a certain amount of time (known as the jitter buffer) before delivering them to the end-user. This allows some packets that arrive late or out of order to be correctly placed in the stream, fact that has been shown that drastically increase conversation quality [32]. The sequencing feature of RTP allows this to be accomplished.

In addition to sequencing, RTP also provides other valuable media-specific features. It has a built-in field to represent the used codec according to a defined list of standard codecs that is kept by IANA [33]. It also provides a timestamp when the packet is created (used by RTCP), and a marker bit, which is usually used to signal the start of a new audio stream, or a special type of packet.

RTP runs together with RTCP (Real-Time Control Protocol). RTCP is a protocol in which two endpoints communicate things as delay, packet loss and jitter to each other over the network. Some applications may use RTCP to renegotiate a media connection depending on the integrity of the channel. For example, if bandwidth seems to be creating excessive delays and packet losses, a new codec can be used that represents a lower voice quality, but requires less bandwidth.

Figure 6 presents a RTP packet; below can be found a brief explanation concerning to some specific fields of the packet and its functionality.

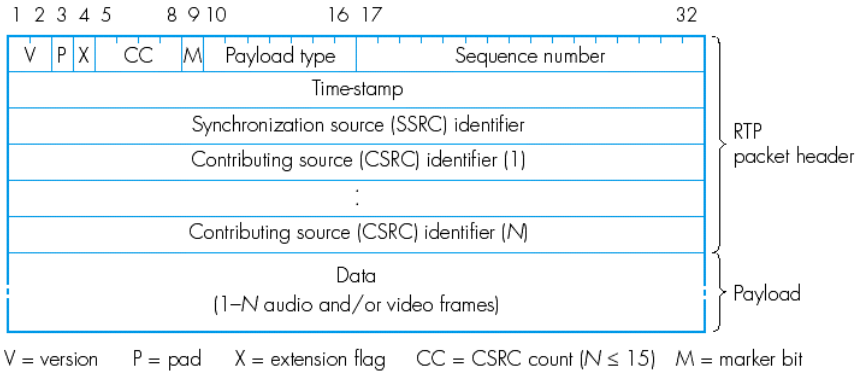

Figure 6: RTP packet.

V, Version (2 bits)

RTP version number. Always set to 2.

P, Padding (1 bit)

If set, this packet contains one or more additional padding bytes at the end which are not part of the payload. The last byte of the padding contains a count of how many padding

bytes should be ignored. Padding may be needed by some encryption algorithms with fixed block sizes or for carrying several RTP packets in a lower-layer protocol data unit.

X, Extension (1 bit)

If set, the fixed header is followed by exactly one header extension.

CC, CSRC count (4 bits)

The number of Contribution Source identifiers that follow the fixed header.

M, Marker (1 bit)

The interpretation of the marker is defined by a profile. It is intended to allow significant events such as frame boundaries to be marked in the packet stream. A profile may define additional marker bits or specify that there is no marker bit by changing the number of bits in the payload type field.

Payload Type (7 bits)

Identifies the format of the RTP payload and determines its interpretation by the application. A profile specifies a default static mapping of payload type codes to payload formats. Additional payload type codes may be defined dynamically through non-RTP means. An RTP sender emits a single RTP payload type at any given time; this field is not intended for multiplexing separate media streams.

Sequence Number (16 bits)

The sequence number increments by one for each RTP data packet sent and may be used by the receiver to detect packet loss and to restore packet sequence. The initial value of the sequence number is random (unpredictable).

Time-stamp (32 bits)

The timestamp reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations. The resolution of the clock must be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter. The clock frequency is dependent on the format of data carried as payload and is specified statically in the profile or payload format specification that defines the format or may be specified dynamically for payload formats defined through non-RTP means. If RTP packets are generated periodically, the nominal sampling instant as determined from the sampling clock is to be used, not a reading of the system clock.

SSRC, Synchronization source identifier (32 bits)

Identifies the synchronization source. The value is chosen randomly, with the intent that no two synchronization sources within the same RTP session will have the same SSRC. Although the probability of multiple sources choosing the same identifier is low, all RTP implementations must be prepared to detect and resolve collisions. If a source changes its

source transport address, it must also choose a new SSRC to avoid being interpreted as a looped source.

CSRC, Contributing source identifier (32 bits)

An array of 0 to 15 CSRC elements identifying the contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field. If there are more than 15 contributing sources, only 15 may be identified. CSRC identifiers are inserted by mixers, using the SSRC identifiers of contributing sources.

Secure Real-time Transport Protocol

Real-time Transport Protocol [34] (SRTP) is a profile of the Real-time Transport Protocol, which can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the control traffic for RTP, RTCP [35].

SRTP is very suitable for VoIP applications, especially those involving low bit rate voice codecs (G.729, for example), since it can be used with header compression and has no significant impact on Quality of Service [36].

SRTP is the security layer which resides between the RTP/RTCP application layer and the transport layer, generating SRTP packets from the RTP/RTCP stream and forwarding these to the receiver. Similarly, it also transforms incoming SRTP packets to RTP/RTCP packets and passes these up the stack. The cryptographic state information associated with each SRTP stream is termed the cryptographic context. It must be maintained by both the sender and receiver of SRTP streams. If there are several SRTP streams present within a given RTP session, separate cryptographic contexts must be maintained for each. A cryptographic context includes any session key (a key directly in encryption/message authentication) and the master key (a securely exchanged random bit string used to derive session keys), as well as other working session parameters.

While SRTP does not define a precise mechanism to implement key exchange, it does provide for several features which make key management easier and heighten overall key security.

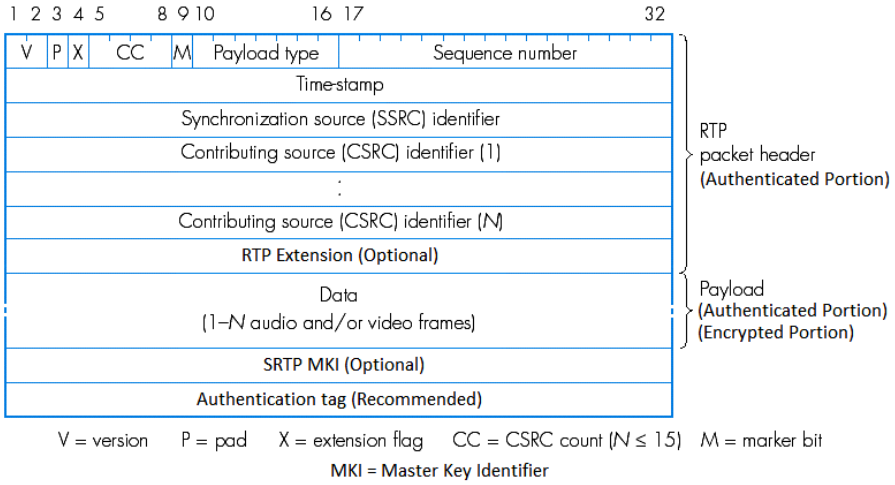

Figure 7: An RTP packet with SRTP additions.

As *Figure 7* displays, SRTP encrypts only the payload (i.e., the media) for confidentiality. The authentication algorithm protects the integrity of the entire original RTP packet. The

optional Master Key Identifier (MKI) and the recommended authentication tag are the only SRTP additions to the original RTP packet.

The SRTP MKI identifies which master key was used to derive the session keys currently in use for the encryption and/or authentication of the current packet. While sometimes not used, a typical MKI is 4 bytes in length and is used in a system that may employ multiple key exchanges. The authentication tag is used to carry message authentication data, and is also of a configurable length, but is usually either 4 or 10 bytes long.

Authentication ensures that attackers can neither modify packets in the stream nor insert (forge) additional packets. The authentication operation is performed after the encryption operation and protects the entire RTP packet. Since the sequence number is part of this protection, the authentication tag provides protection against replay.

3.4. Audio Codecs

A codec is a device or program capable of performing encoding and decoding on a digital data stream or signal. Various types of codecs are used to encode and decode, or compress and decompress data that would otherwise use large amounts of bandwidth on WAN links. Codecs are especially important on lower-speed serial links where every bit of bandwidth is needed and utilized to ensure network reliability [1].

Coding techniques are standardized by the ITU. The ITU G-series codecs are among the most popular standards for VoIP applications. These codecs use many types of encoding algorithms, which encode the digital signal for transmission in order to save bandwidth. Among all the encoding algorithms, the most used ones are Pulse-Code Modulation (PCM), Adaptive Differential Pulse-Code Modulation (ADPCM), Low Delay Code Excited Linear Prediction (LD-CELP), Multi-Pulse Maximum Likelihood Quantization (MP-MLQ), Algebraic Code Excited Linear Prediction (ACELP) and Conjugate-Structured Algebraic Code-Excited Linear Prediction (CS-ACELP).

Following table, Table **4**, presents a list and a brief information of the most used audio codecs:

Table 4: Audio codecs information [37].

Name	Bit Rate [Kbps]	Codec Sample Size [Bytes]	Codec Sample Interval [ms]	Encoding Algorithm	Mean Opinion Score [MOS]
G.711	64	80	10	PCM	4.10
G.726	32	20	5	ADPCM	3.85
G.728	16	10	5	LD-CELP	3.61
G.729	8	10	10	CS-ACELP	3.92
G.723.1	6.3	24	30	MP-MLQ	3.90
G.723.1	5.3	20	30	ACELP	3.80

Explanation of Terms

Codec Bit Rate

Based on the codec, this is the number of bits per second that need to be transmitted to deliver a voice call.

Codec Sample Size

Based on the codec, this is the number of bytes captured by the Digital Signal Processor (DSP) at each codec sample interval.

Codec Sample Interval

This is the sample interval at which the codec operates.

Mean Opinion Score (MOS)

Mean Opinion Score mark, for a particular codec is the average mark given by a panel of auditors listening to several recorded samples (voice samples, music samples, voice with background noise, etc.). This score range from 1 (worst) to 5 (best) [38].

3.5. VoIP Risks and Vulnerabilities

As VoIP is an IP-based technology that utilises the Internet, it also inherits all associated IP vulnerabilities. The impact of these Internet-borne attacks is then multiplied by the VoIP architecture as it adds a number of additional weaknesses, which require further work to secure and maintain. Furthermore, as with adding any new service to an inadequately secured environment, is like piercing holes in an already-leaky boat.

The VoIP risks and vulnerabilities boils down into the threats inherited from IP, associated with VoIP, and lastly, specific to VoIP [39]:

Risks and Vulnerabilities Inherited from IP

Poor Architectural Design

Poor or inadequate architecture can lead to ongoing difficulties in the operation and security of a VoIP system. Firewalls are particularly vulnerable areas in a VoIP network as they require additional ports to be opened to facilitate VoIP traffic. Non VoIP-aware firewalls may lack dynamic interaction with VoIP so they simply leave a range of ports continually open for call activity.

PBX Hosts & Gateways

Most service interceptions and eavesdropping attacks will usually require the compromise of a PBX as a means of network access. A compromised host or gateway can facilitate this by capturing voice packets to reveal information on all calls, call duration and call parameters. This information will permit the mapping of VoIP and possibly the supporting data networks.

Replay Attacks

A replay attack can be mounted against a VoIP network by retransmitting a legitimate session so that the recipient device reprocesses the data. The basis of a replay attack is to capture a valid packet, which can then be replayed into the network. This generally causes the target network to respond and provide more traffic to capture, eventually providing

enough information to move to packet spoofing and masquerading or simply finding an entry point into the target network for eavesdropping.

Risks and Vulnerabilities Associated with VoIP

Packet Spoofing & Masquerading

Packet spoofing uses IP packets with a false source address. A major risk associated with packet spoofing and masquerading is identity theft. A man-in-the-middle spoofing attack, as shown in *Figure 8*, can be launched when a person makes a call, which includes sensitive information. As a result of the attack they may speak to the intended recipient, however, their call is being monitored by malicious users.

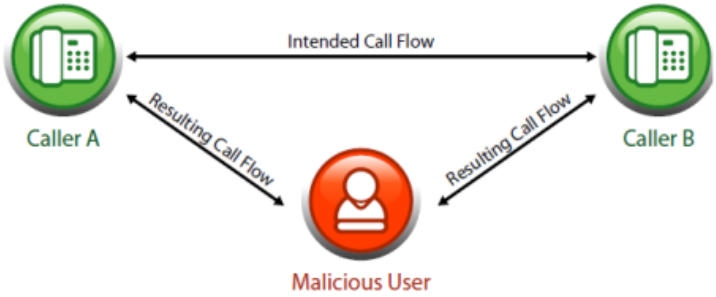

Figure 8: Resultant call flow from a man-in-the-middle spoofing attack [39].

Fuzzing

Fuzzing is a legitimate method of testing software systems for bugs and is accomplished by providing an application with semi-valid input to see what its reaction will be. This technique can be employed to exploit vulnerabilities in a target VoIP system and is achieved by sending messages so that the target system will assume the sent content is valid. In reality, the message is "broken" or "fuzzed", thus causing various failures to occur when the target system attempts to parse or process it. Resultant failures can include application delays, information leaks and system crashes.

Reconnaissance Attacks

Reconnaissance Attacks are a form of intelligence gathering where networks are probed to ascertain their vulnerabilities. Methods used to achieve this include call walking and port scanning and are the first action undertaken by an attacker when attempting to penetrate a network. A successful probe would determine the behaviour of the network's equipment, users and services that might be available to be exploited or disrupted. This information could then be used to launch a focused attack against the network.

Denial of Service (DoS)

DoS and Distributed Denial of Service (DDoS) attacks occur when a malicious user deliberately sends an exceedingly large amount of random messages to one or more VoIP end-points from either a single location (DoS) or from multiple locations (DDoS), as shown in *Figure 9*. Multiple locations are achieved through the use of zombies (compromised machines that could be woken upon request and used for malicious purposes). The DoS attack is successful when the amount of incoming messages exceeds the processing capacity

of the target system, thereby exhausting system resources and thus, denying services to the end-users.

VoIP systems are especially vulnerable to DoS and DDoS attacks because of the high fundamental requirement that they place on QoS. Therefore less traffic or network disruption is required for a DoS attack to be successful when compared to mounting a DoS attack against a data network. A further consideration is needed where VoIP and data share the same network. Here the data network could also be subject to the same DoS attack.

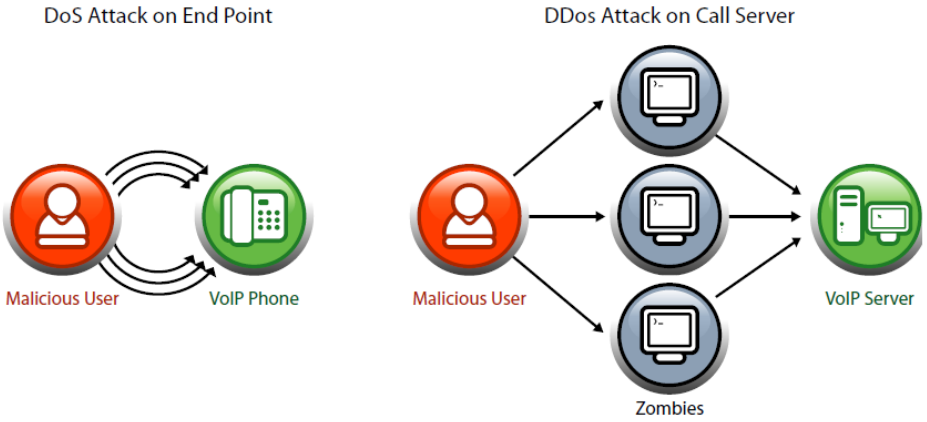

Figure 9: DoS and DDoS attacks launched against a SIP infrastructure [39].

Risks and Vulnerabilities Specific to VoIP

CID Spoofing

One type of masquerading is based on the manipulation of Caller ID (CID), which is used to identify the caller before answering and is known as CID spoofing. The CID is based on reported information from different carrier switches and is specified by the switch administrator in a VoIP environment. This allows an attacker to spoof their CID information with a text string or phone number he specify and could be used to give credibility to various malicious users undertaking social engineering attacks.

In addition to this, the option for CID privacy (i.e. the ability to obscure your phone number from the CID display) is not possible with VoIP, since the phone number is included in the SIP and H.323 header. This allows any attacker with an IP packet sniffer, such as Wireshark [18], to discover the remote caller's phone number, even if their number has been marked as private by their service provider.

Phone Impersonation

Phone impersonation occurs due to the weak authentication process attributed to VoIP. There are two major contributors that consolidate this fact. Firstly, there is a limited human interface available for VoIP phones, limiting users to the selection of a numeric PIN for their password in lieu of a strong password based on the entire ASCII character set. Secondly, and this is related to the SIP standard, the authentication mechanism is based on the MD5 algorithm. An attacker who can sniff the entire SIP authentication exchange cannot observe the password sent in plain text, but can observe enough information to mount an offline dictionary attack against the password. The combination of these weaknesses allows passwords to be easily obtained by an attacker and then used to impersonate a phone or user.

Eavesdropping

Eavesdropping is the unauthorized interception of voice packets or RTP media streams, and the decoding of signalling messages. It is a relatively simple attack. Wireshark [18] is an example of a tool that can be used to capture VoIP traffic and reconstruct VoIP conversations.

Call Hijacking & Redirection

Call hijacking and redirection occurs when a call intended for one user is redirected to another. To achieve this, an attacker only needs to have knowledge of the user's authentication credentials in order to impersonate and receive all calls intended for that user. Methods including spoofing of a node, man-in-the-middle attacks and manipulation of call requests using signalling response codes make call hijacking and redirection relatively easy to instigate. Further to this, VoIP features including call forwarding and "follow-me" also help facilitate the ability to route calls to specific phone numbers.

VoIP Spam

VoIP SPAM or Spam over Internet Telephony (SPIT) is the unsolicited and unwanted bulk messages broadcast over VoIP to particular end users. Not only could this be extremely annoying, it also has the potential to be rather costly where for example, calls are forwarded to mobile phones. Another issue arises with SPIT, which is the fact that high-volume bulk calls routed over IP are very difficult to trace and have the inherent capacity for fraud, unauthorised resource use, and privacy violations. Voice mail bombing is a form of SPIT where multiple voice mail messages flood voice mail boxes. This attack could result in service disruption or a denial of service attack.

4 EVALUATION OF SPEECH QUALITY IN IP TELEPHONY

4.1. Introduction

In traditional telephony, quality of service for each and every phone call is guaranteed by the constant availability of dedicated bandwidth. Whenever a channel or "loop" is established across the network, the bandwidth allocated to that channel is steadfast and unchanging. Most digitally encoded call paths on the PSTN use the same codec, G.711, so transcoding is not necessary. Almost no processing bottlenecks will be found on the PSTN, and since the system isn't generally packet-based, there is almost never degradation in perceived call quality as a result of congestion. If the PSTN and Signalling System number 7 (SS7) cannot establish a full-bandwidth path through the network, the call just does not get connected and the caller hears a busy tone. The designers of the PSTN felt a breakdown of connectivity would be preferable to a breakdown of quality.

Of course, packet networks work the other way. When bandwidth availability drops, as more packets are sent on the network, throughput slows. Until a certain breaking point, bandwidth availability can be compromised while still allowing data through; the transmission just slows down. Some applications tolerate congestion and slow throughput better than others. The more tolerance an application has, the higher its error budget is said to be.

Slowness of transmission latency is the enemy of Voice over IP and one of the key contributors to failure with the technology. Aside from careful network design and bandwidth provisioning, which are factors in building any IP network, there is an elegant solution to the latency problem, one that allows local and end-to-end guarantees of bandwidth and prioritization of real-time traffic over less sensitive traffic [2].

4.2. QoS Concepts

Because of the nature of IP networking, voice packets sent via IP are subject to certain transmission problems. Conditions present in the network might introduce problems such as echo, jitter or delay. These problems must be addressed with QoS mechanisms. The clarity, be it "cleanliness" and "crispness", of the audio signal is of utmost importance.

The following factors can affect clarity, and consequently, speech quality [1]:

Fidelity

Fidelity is the degree to which a system, or a portion of a system, accurately reproduces at its output the essential characteristics of the signal impressed upon its input, or the result of a prescribed operation on the signal impressed upon its input. The bandwidth of the transmission medium almost always limits the total bandwidth of the spoken voice.

Echo

Echo is a result of electrical impedance mismatches in the transmission path. Echo is always present, even in traditional telephony networks, but at a level that cannot be detected by the human ear. The two components that affect echo are amplitude (loudness of the echo) and delay (the time between the spoken voice and the echoed sound). Echo can be controlled by using suppressors or cancellers.

Jitter

Jitter is variation in the arrival of coded speech packets at the far end of a VoIP network. The varying arrival time of the packets can cause gaps in the recreation and playback of the voice signal. These gaps are undesirable. Delay is induced in the network by variation in the routes of individual packets, contention or congestion. Variable delay can be solved by using dejitter buffers.

Delay

Delay is the time between the spoken voice and the arrival of the electronically delivered voice at the far end. Delay results from multiple factors, including distance (propagation delay), coding, compression, serialization and buffers.

Packet Loss

Voice packets might be dropped under various conditions, such as an unstable network, network congestion or too much variable delay in the network. Lost voice packets are not recoverable, resulting in gaps in the conversation that are perceptible to the user.

Side Tone

Side tone is the purposeful design of the telephone that allows the speakers to hear their spoken audio in the earpiece. Without side tone, the speaker is left with the impression that the telephone instrument is not working.

Background Noise

Background noise is the low-volume audio that is heard from the far-end connection. Certain bandwidth-saving technologies can eliminate background noise altogether, such as voice activity detection (VAD). When this technology is implemented, the speaker audio path is open to the listener, while the listener audio path is closed to the speaker. The effect of VAD is often that speakers think the connection is broken because they hear nothing from the other end. Therefore, VAD is often combined with comfort noise generation (CNG) to prevent the illusion that the call has been disconnected.

4.3. Telephony QoS

In the field of telephony, Quality of Service was defined in the ITU standard X.902 as "a set of quality requirements on the collective behaviour of one or more objects". Quality of Service comprises requirements on all the aspects of a connection, such as service response time, loss, signal-to-noise ratio, cross-talk, echo, interrupts, frequency response, loudness levels and so on. A subset of Telephony QoS is Grade of Service (GoS) requirements, which comprises aspects of a connection relating to capacity and coverage of a network, for example guaranteed maximum blocking probability and outage probability [40].

QoS in IP Telephony is normally used as a quality measure, with many alternative definitions, rather than referring to the ability to reserve resources. Quality of Service sometimes refers to the level of quality of service, i.e., the guaranteed service quality. High QoS is often confused with a high level of performance or achieved service quality, for example high bit rate, low latency and low bit error probability.

An alternative and disputable definition of QoS, used especially in telephony and streaming video services, is a metric that reflects or predicts the subjectively experienced quality, for example, the Quality of Experience (QoE), the "user perceived performance" [41], the "degree of satisfaction of the user", the "number of happy customers" or the Mean Opinion Score. In this context, QoS is the cumulative effect on subscriber satisfaction of all imperfections affecting the service. This definition includes the application and the human in the assessment and demands an appropriate weighting of diverse objective measures.

Quality of Service Experience (QoSE) on the other hand, is the actual measure of user's experience with an operator in terms of delivered quality with or without reference to what is being promised. This differs from QoS (defined as not taking user experience into consideration) as the QoSE is defined only in the context of user experience and from QoE which might be an objective measure of the subjective user experience.

4.4. Audio Quality Measurement

A multitude of information is contained in the speech signal typical of everyday communication, as it may happen, for example, during a telephone conversation. Although context and perceptual mechanisms, as well as, adaptation of the communication behaviour yield reliable comprehension even in unfavourable conditions, deteriorations of spectral and/or temporal cues of the speech signal degrade the contained information: be it the linguistic information (the actual message) or paralinguistic information regarding the speaker identity and others. Even if comprehension is not at stake, other factors, such as annoyance, modify the user's appreciation of a particular connection. In this context, the term speech quality is typically used, extending the comprehension-oriented view on speech to additional factors governing speech perception [42].

There are many aspects to voice service quality, but by far the most visible and important is speech quality. Speech quality, also known as clarity, refers to the clearness of a speaker's voice as perceived by a listener. It is the result of the fidelity of speech signal reproduction in a VoIP network. Speech quality is a one-way phenomenon; that is, it is experienced in one direction: from speaker to listener; and it affects listening quality [43].

Several methods can be used to determine signal quality [1], including the following:

- Subjective Mean Opinion Score (MOS) Method;
- Perceptual Speech Quality Measurement (PSQM);
- Perceptual Evaluation of Speech Quality (PESQ).

Subjective MOS Method

Subjective MOS method for determination of transmission quality is difined in ITU-T Recommendation P.800 [44].

MOS is a scoring system for voice quality. A MOS score is generated when listeners evaluate pre-recorded sentences that are subject to varying conditions, such as compression algorithms. Listeners then assign the sentences values, based on a scale from 1 through 5, where 1 is the worst and 5 is the best. The sentence used for English-language MOS testing is, "Nowadays, a chicken leg is a rare dish". This sentence is used, because it contains a wide range of sounds found in human speech, such as long vowels, short vowels, hard sounds and soft sounds.

The test scores are then averaged to a composite score. The test results are subjective, because they are based on the opinions of the listeners. The tests are also relative because a score of 3.8 from one test cannot be directly compared to a score of 3.8 from another test.

Therefore, must be established a baseline for all tests, such as the same used codec, so the scores can be normalized and compared directly.

Table **5** presents the average user opinion related to the respective MOS ratings.

Table 5: MOS ratings [45].

User Opinion	MOS
Very Satisfied	4.1 – 5.0
Satisfied	3.7 – 4.1
Some Users Satisfied	3.4 – 3.7
Many Users Dissatisfied	2.9 – 3.4
Nearly All Users Dissatisfied	2.4 – 2.9
Not Recommended	1.0 – 2.4

PSQM

PSQM is an automated method of measuring speech quality "in service", or as the speech happens. PSQM software usually resides with IP call management systems, which are sometimes integrated into SNMP systems. The measurement is made by comparing the original transmitted speech to the resulting speech at the far end of the transmission channel. PSQM systems are deployed as in-service components. The PSQM measurements are made during real conversation on the network. This automated testing algorithm has over 90 percent accuracy compared to subjective listening tests such as MOS. Scoring is based on a scale from 0 through 6.5, where 0 is the best and 6.5 is the worst. Because it was originally designed for circuit-switched voice, PSQM does not take into account the jitter or delay problems experienced in packet-switched voice systems.

PESQ

PESQ was specifically developed to be applicable to end-to-end voice-quality testing under real network conditions, like VoIP, PSTN, Integrated Services Digital Network (ISDN) and Global System for Mobile Communication (GSM). PESQ was developed by combining the two advanced speech quality measures PSQM+ and Perceptual Analysis Measurement System (PAMS).

PESQ can take into account codec errors, filtering errors, jitter problems and delay problems that are typical in a VoIP network. It combines the best of the PSQM method along with a method called PAMS. PESQ scores range from 1 (worst) through 4.5 (best), with 3.8 considered acceptable quality. PESQ is meant to measure only one aspect of voice quality. The effects of two-way communication, such as loudness loss, delay, echo and side tone, are not reflected in PESQ scores.

Voice-Quality Measurement Comparison

Early quality measurement methods, such as MOS and PSQM, were designed before widespread acceptance of VoIP technology. PESQ was designed to address the shortcomings of MOS and PSQM.

MOS uses subjective testing in which the average opinion of a group of test users is calculated to create the MOS score. This method is both time-consuming and expensive and might not provide consistent results between groups of testers.

PSQM and PESQ use objective testing in which an original reference file sent into the system is compared with the impaired signal that came out. These testing methods provide

an automated test mechanism that does not rely on human interpretation for result calculations.

However, PSQM was originally designed for circuit-switched networks and does not take into account the effects of jitter and packet loss.

PESQ measures the effect of end-to-end network conditions, including codec processing, jitter and packet loss. Therefore, PESQ is the preferred method of testing voice quality in an IP network.

Table **6** offers a comparison of the features offered by MOS, PSQM, and PESQ.

Table 6: Quality measurement techniques comparison [1].

Feature	MOS	PSQM	PESQ
Test Method	Subjective	Objective	Objective
End-to-end packet loss test	Inconsistent	No	Yes
End-to end jitter test	Inconsistent	No	Yes

4.5. ITU-T E-Model

The model currently recommended by the ITU–T for network planning is the so-called E-model (ITU–T Recommendations G.107 and G.108 [43]). It is mainly empirical by nature and was developed on the basis of large amounts of auditory test data (both Listening Only Tests and Conversation Tests). The E-model takes the impairments due to talker echo and transmission delay into consideration and can hence be applied to predicting quality in a conversational situation. It was compiled as a combination of different quality prediction models in the framework of the European Telecommunications Standards Institute. The E-model is widely used to estimate quality during the planning of networks and also during their operation (monitoring). The model has been subject to extensive validation, and has been found to deliver realistic quality estimates for individual and within limits also for combined degradations [42].

There are a few advantages to the E-model over perceptual models. Unlike PESQ, the E-model takes absolute delays into account when determining voice quality. This is impossible to do in a perceptual model that only compares end-point signals. Additionally, the E-model is broken up into different terms that each represent a different aspect of the connection that contributes to conversation quality. This makes it easier to determine what is causing a particular communication channel to be deficient. Finally, the E-model was designed with the knowledge of VoIP systems in mind and includes terms for codec and packet loss. This makes it more easily applied to VoIP systems than the other models [46].

A primary output of the E-model is the Rating Factor, R, also known as the "R-factor". The R-factor is calculated based on adding assumed impairment factors in a call path. It is not based on any measured parameters of an actual network [43]. R-factor is used to express quality.

In this model, the overall transmission rating R is defined by the equation (**1**):

$$R = R_0 - I_s - I_d - I_{e,eff} + A \quad (1)$$

The R-factor for the overall transmission rating ranges from 0 to 100 (100 corresponding to optimum and 0 to worst quality). The classes of degradations are represented as follows:

- $\mathbf{R_0}$ describes the speech quality due to the basic signal-to-noise ratio, as it results from the speech level and the levels of the different noise sources on the line, be it line noise or room noise at send or receive side;

- The Simultaneous Impairment Factor, $\mathbf{I_s}$, accounts for the effect of degradations simultaneous to the transmitted speech signal, such as signal-correlated noise or excessive speech levels;
- The Delayed Impairment Factor, $\mathbf{I_d}$, stands for the impairments that are delayed with respect to the transmitted speech signal, such as transmission delay or echo;
- The Effective Equipment Impairment Factor, $\mathbf{I_{e,eff}}$, comprises the Equipment Impairment Factor $\mathbf{I_e}$, which quantifies the speech quality impairment due to low-bit-rate coding. To include the effect of packet loss, the loss-free $\mathbf{I_e}$ has recently been extended to the loss-dependent $\mathbf{I_{e,eff}}$.
- The Advantage Factor **A** quantifies the advantage of access related to a particular system. For example, a user of a mobile service may be more tolerant towards a degraded channel than a user of a wire-line network, owing to the advantage of mobility.

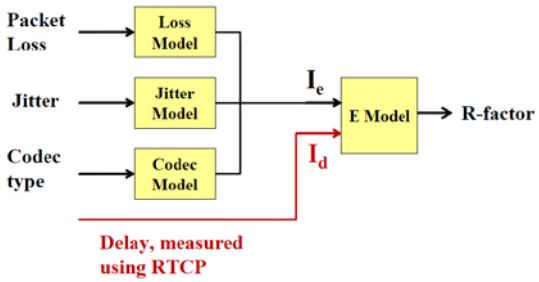

Figure 10: The factors affecting the computation of the E-model and the R-factor [47].

If the elimination of the influence of Signal-to-noise ratio (SNR) and the echo signal attenuation is assumed, the calculation of the R-factor is simplified (**2**) according to the values of the ITU-T Recommendation G.107 [48], which are $\mathbf{R_0}$=94,7688, $\mathbf{I_S}$=1,4136 and **A**=0.

$$R=R_0-I_s-I_d-I_{e,eff} = 93{,}3553 - I_d-I_{e,eff} \quad (\mathbf{2})$$

In order to achieve the parameters $\mathbf{I_{e,eff}}$ and $\mathbf{I_d}$ should be used the following equations (**3**)(**4**) [49]:

$$I_{e,eff}=I_e+(95{,}0-I_e)\cdot\frac{P_{pl}}{\frac{Ppl}{BrustR}+B_{pl}} \quad (\mathbf{3})$$

$$I_d=\begin{cases} 0{,}0267\cdot d\,, & d<125\,ms \\ 0{,}1194\cdot d-15{,}8760\,, & 175\text{ ms}\le d\le 400\text{ ms}\end{cases} \quad (\mathbf{4})$$

where $\mathbf{P_{pl}}$ is the loss probability in percentage, $\mathbf{B_{pl}}$ the codec packet-loss robustness factor and $\mathbf{d_{[ms]}}$ the delay in milliseconds. The $\mathbf{B_{pl}}$ and $\mathbf{I_e}$ variables can be found on the ITU-T Recommendation G.113 [50].

However, for some coders, the subjective impairment due to burst packet loss can be reflected using the so-called Burst Ratio, **BurstR**, which partly captures the "burstiness" of a specific loss distribution (**5**) [50]:

$$BurstR=\frac{\text{Average length of observed bursts in an arrival sequence}}{\text{Average length of burs ts expected for the network under "random " loss}} \quad (\mathbf{5})$$

when packet loss is random **BurstR**=1; and when packet loss is bursty **BurstR** > 1.

The packet loss distribution can be modelled using a Markov process. A multi-state Markov Model is used to measure the distribution of lost or discarded packets or frames, and to divide the call into "bursts" and "gaps". For packet loss distributions corresponding to a 2-state Markov model with transition probabilities **p** between a "found" and a "loss" state, and **q** between the "loss" and the "found" state, the Burst Ratio can be calculated as [3]:

$$\mathrm{BurstR} = \frac{1}{p+q} \quad (6)$$

For its default settings (ITU–T Recommendation G.107), corresponding to a clean ISDN channel operated with handset telephones, the current version of the E-model predicts a transmission rating of R = 93.2 [42].

The transmission rating factor R predicted by the E-model can be converted to a MOS. The relation between R-factor and MOS is depicted in *Figure 11* and *Figure 12*.

Figure 11: Relation between R-factor and MOS [51].

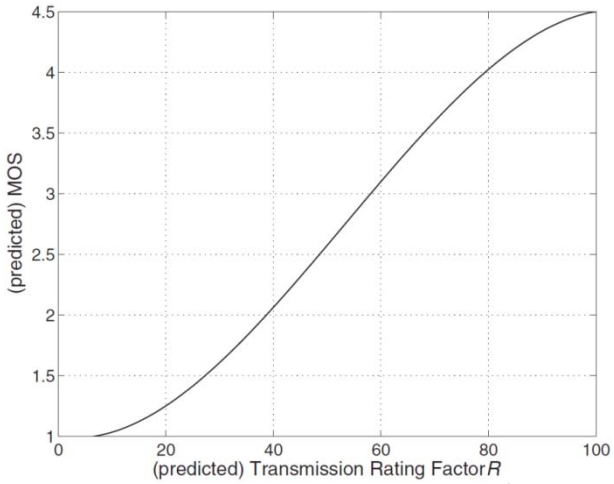

Figure 12: Relation between MOS and R-factor, according to ITU–T Rec. G.107 (2005) [42].

The ITU-T presented E-model has several advantages. There is no required a-priori knowledge of the transport network technical characteristics that is used to transmit voice traffic, no need to use the original encoded samples that are transported over the internet and, is not necessary to use a large number of end-users with diverse characteristics [47].

4.6. Audio Quality Expectations

Customer expectations regarding voice quality exist at two levels. The first is the PSTN, where the expectation is for "toll quality" voice with a MOS rating of 4.0 or more. And indeed, most PSTN calls achieve an excellent MOS 4.3 rating - about as good as it gets for narrowband speech. The second level has been established by the cellular phone network, where the expectation is considerably lower (MOS around 3.2 – 3.5). Of course, cellular customers willingly tolerate this "fair" voice quality for the convenience of mobility [45].

VoIP networks generally compete with the former and to be competitive with the PSTN, VoIP services should strive for a MOS of 4.0 or more. This is easier said than done, naturally, because the PSTN has three design characteristics that together maximize voice quality:

- The use of dedicated Time-Division Multiplexing circuits to eliminate contention and minimize latency end-to-end in all connections;

- The use of Pulse Code Modulation, specified in ITU-T Recommendation G.711 at a full 64 kbps (the PSTN's standard DS-0 circuit), for its ability to achieve a MOS rating of 4.4;
- Painstaking measures to remove all impairments that degrade voice quality, including eliminating hybrid and acoustic echo, reducing noise or adding "comfort" noise and normalizing signal levels for consistent volume; all of which serve to maximize intelligibility

As a result of these and other provisions, circuit-based networks enjoy exceptionally high voice quality. But the PSTN infrastructure is very expensive.

4.7. Bandwidth and Effective Bandwidth

In any network, the obvious first concern is whether the network is available for use. Since a network can be broken down into nodes and links between nodes where traffic flows, the quest for an available network boils down to the availability of each node and the availability of each path between the nodes.

As in data networks, bandwidth congestion can cause packet loss and a host of other QoS problems. Thus, proper bandwidth reservation and allocation is essential to VoIP quality. One of the great attractions of VoIP is data and voice sharing the same wires, fact which is also a potential headache for implementers who must allocate the necessary bandwidth for both networks in a system normally designed for one. Congestion of the network causes packets to be queued, which in turn contributes to the latency of the VoIP system. Low bandwidth can also contribute to non-uniform delays (jitter), since packets will be delivered in spurts when a window of opportunity opens up in the traffic.

Bandwidth usage varies significantly across a WAN, so a complex methodology is needed to estimate required bandwidth usage. One provided methodology is based on an analysis of the aggregate bandwidth needed in terms of the amount of traffic and its rate of flow [52].

Methods for reducing the bandwidth usage of VoIP include RTP header compression and VAD. RTP compression condenses the media stream traffic, so less bandwidth is used. However, an inefficient compression scheme can cause latency or voice degradation, causing an overall downturn in QoS. VAD prevents the transmission of empty voice packets (i.e., when a user is not speaking, their device does not simply send out white noise). However, by definition VAD will contribute to jitter in the system by causing irregular packet generation.

Adding security constraints significantly increases the bandwidth usage, causing more latency and jitter, thereby degrading the overall QoS of the network. In addition, these requirements do not explicitly take into account the heterogeneous data flow over the network. Since voice and data streams are sharing the same finite bandwidth and data streams tend to contain much larger packets than VoIP, significant amounts of data can congest the network and prevent voice traffic from reaching its destination in a timely fashion. For this reason, most new hardware devices deployed on networks support QoS for VoIP. These devices, such as routers and firewalls, make use of the IP protocol's Type of Service (ToS) bits to send VoIP traffic through before less time urgent data traffic.

Not only is the available bandwidth of the system affected by the introduction of security measures, but in addition the effective bandwidth of the VoIP system is significantly depreciated. Effective bandwidth is defined as the percentage of bandwidth carrying actual data with regard to the total bandwidth used [53]. The introduction of IPsec or other forms of encryption, as TLS, results in a much larger header to payload ratio for each packet and this reduces the effective bandwidth as the same number of packets (but larger sized) are

used to transport the same amount of data. The consequences of this reduction include decreased throughput and increased latency.

Following table, Table **7**, presents a brief information of the used bandwidth for the most used audio codecs:

Table 7: Cisco CallManager bandwidth calculations [37].

Name & Bit Rate (Kbps)	Voice Payload [Bytes]	Voice Payload [seconds]	Packets Per Second [PPS]	Bandwidth Ethernet [Kbps]
G.711 (64 Kbps)	160	20	50	87.2
G.726 (32 Kbps)	80	20	50	55.2
G.728 (16Kbps)	60	30	34	31.5
G.729 (8 Kbps)	20	20	50	31.2
G.723.1 (6.3 Kbps)	24	30	24	21.9
G.723.1 (5.3 Kbps)	20	30	24	20.8

4.8. Overcome QoS Issues

The key to overcome QoS issues, as latency and bandwidth congestion, is speed. By definition, faster throughput means reduced latency and probabilistically reduces the chances of severe bandwidth congestion. Thus, every facet of network traversal must be completed quickly in VoIP. The latency often associated with tasks in data networks will not be tolerated. Chief among these latency producers that must improve performance are firewall/NAT traversal and traffic encryption/decryption. Traditionally, these are two of the most effective ways to secure networks. However, they are also two of the greatest contributors to network congestion and throughput delay. Inserting traditional firewall and encryption products into a VoIP network is not feasible, particularly when VoIP is integrated into existing data networks.

5 NETWORK SECURITY TECHNIQUES

5.1. Introduction

Information is important for any person, company or organization. Damage or misuse of information may result in disaster to a user or the entire company. In addition, the appearance of internet introduces opportunities for unauthorized person to access the information.

Confidentiality (Privacy), Integrity and Availability (CIA) are used to evaluate system security [54]. In general, Internet Security can be divided into these three central issues. They are the main requirements of information security.

Confidentiality or Privacy

Confidentiality means keeping the information private. This means protection of data privacy, prevent unauthorized access to data, resource or service, where only authorized people can access the data, service or resource. A user should be able to control who has access to information he wants to keep private. Confidentiality is also related with privacy which means individual personal information or similar. Confidentiality is the target being attacked most often.

Confidentiality in the context of VoIP means that a person who is not involved in the conversation should not be able to determine who is talking to whom or what is being said.

Integrity

Integrity is one of the most important aspects in network security; it is used frequently when considering security. Integrity means keeping the data in its original form without any modification. Integrity has three goals: ensure the information in the original form, prevent unauthorized modification and prevent incorrect modification by accident.

Integrity basically means that whatever information we receive is guaranteed to be what we think it is. A malicious attacker cannot modify the information without us knowing and cannot pretend to be someone he is not.

Integrity in VoIP systems implies that the person we are talking to, is who they claim to be, and their words cannot be altered in transit. Integrity is also important in communicating with VoIP servers that may route or track our calls. A second, more subtle aspect of integrity is that the conversation should not be able to be replayed at some date as though it is live. This aspect is known as replay protection.

Availability

Availability means the required information, service, resource or device is always available when it is needed by an authorized user. It also means that an attacker should not be able to prevent a person from using a system. A compromise of availability is usually accomplished through a DoS attack. DoS attacks can be specifically targeted at a single point or can take down an entire system.

For VoIP systems to maintain availability it should be impossible for attackers to prevent a single person from using VoIP. A less stringent definition of availability implies that attackers should not be able to take down crucial components of the VoIP architecture (SIP Servers, for example).

Defending availability is the most difficult security task to accomplish because of the large number, distributed location and shared functionality of the network devices that participate in a single call. Additionally, any DoS attack on the Internet could potentially affect VoIP users.

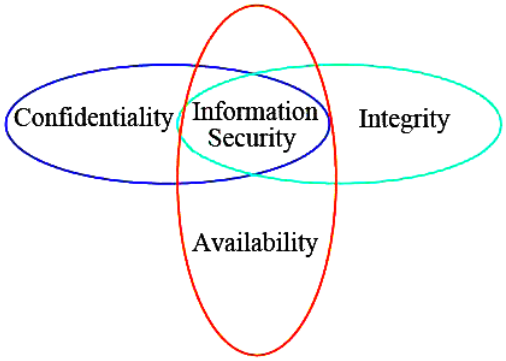

Figure 13: Relationship between CIA and Information Security [55].

IP Telephony has plenty vulnerabilities that needs to be protected, fact that have been discussed in the section **3.5**; the standard protection tool is cryptology.

The security elements Confidentiality, Integrity and Availability are accomplished through cryptography.

Cryptography does not just provide privacy; it also provides security. Using cryptography we can ensure that we are talking to whom or what we intend, whether it is a person or an ATM machine. We can ensure that no one else is eavesdropping on us and that no one else is pretending to be us. By encrypting data, we prevent information leakage. We protect against manipulation of our data stream. The security works both ways. We can trust them and they can trust us. Security gives us integrity [56].

5.2. Cryptography

Cryptography is the study of mathematical techniques for all aspects of information security. Cryptanalysis is the complementary science concerned with the methods to defeat these techniques. Cryptology is the study of cryptography and cryptanalysis. Key features of information security information include confidentiality or privacy, data integrity, authentication and non-repudiation. Each of these aspects of message security can be addressed by standard methods in cryptography. Besides exchange of messages, tools from cryptography can be applied to sharing an access key between multiple parties so that no one person can gain access to a vault by any two of them can [57].

The goal of cryptography is to make it possible for two people to exchange a message in such a way that other people cannot understand the message. There is no end to the number of ways this can be done.

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient. Decryption is the process of transforming encrypted information so that it is intelligible again. A cryptographic algorithm, also called a cipher, is a mathematical function used for encryption and/or decryption. In most cases, two related functions are employed, one for encryption and the other for decryption. In cryptography, a block cipher is a symmetric-key cipher which operates on fixed-length groups of bits, termed blocks, with an unvarying transformation. When encrypting, a block cipher might take, for example, a 128-bit block of plain-text as input and output a corresponding 128-bit block of cipher-text.

With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, but on a number called a key that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple. Decryption without the correct key is very difficult and in some cases impossible for all practical purposes.

The modern field of cryptography can be divided into several areas. The chief ones are Symmetric-key encryption and Public-key (asymmetric) encryption [58]. These two types of encryption are used, often in conjunction, to provide a variety of security functions for network and information security.

5.2.1. Symmetric-key Encryption

Encryption algorithms that use the same key for encrypting and for decrypting information are called symmetric-key algorithms. The symmetric-key is also called a secret-key because it is kept as a shared secret between the sender and receiver of the information. Otherwise, the confidentiality of the encrypted information is compromised due the major vulnerability of the secret-key algorithm, the need for sharing the secret-key.

A basic symmetric-key encryption and decryption example can be seen in *Figure 14*.

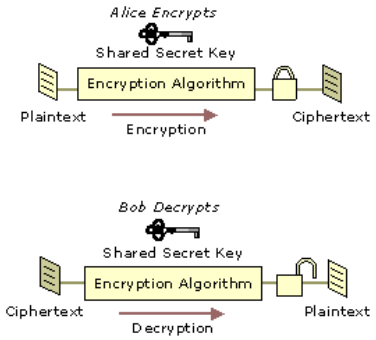

Figure 14: Encryption and decryption with a Symmetric-key [59].

Symmetric keys are commonly used by security protocols as session keys for confidential online communications. For example, the Transport Layer Security and Internet Protocol security protocols use symmetric session keys with standard encryption algorithms to encrypt and decrypt confidential communications between parties. Different session keys are used for each confidential communication session and session keys are sometimes renewed at specified intervals.

Symmetric keys are also commonly used by technologies that provide bulk encryption of persistent data, such as e-mail messages and document files.

Cryptography-based security technologies use a variety of symmetric-key encryption algorithms to provide confidentiality. Some examples of popular and well-respected symmetric algorithms include Advanced Encryption Standard (AES), Blowfish, CAST, Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA), Rivest Cipher # (RC#), Serpent and Twofish.

Strength of the symmetric-key encryption depends on the size of the used key.

Due the scope of this dissertation be concerned with the effect of security mechanisms on speech quality, related to the cipher algorithms were adopted the AES and Triple-DES (3DES), a cipher algorithm that applies DES three times. Among all the available cipher modes of operation, the choice of the used one dropped to the Cipher-Block Chaining (CBC).

A brief analyze of these two distinct cipher algorithms, as well as the used cipher mode of operation is presented below.

Advanced Encryption Standard

In 1997, the National Institute of Standards and Technology (NIST) called for submissions for a new standard to replace the aging DES. The contest terminated in November 2000 with the selection of the Rijndael cryptosystem as the AES [57].

The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. The Rijndael cryptosystem operates on 128-bit blocks, arranged as 4 × 4 matrices with 8-bit entries, with key sizes of 128, 192 and 256 bits, respectively.

The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plain-text into the final output of cipher-text. Each round consists of several processing steps, including one that depends on the encryption-key. A set of reverse rounds are applied to transform cipher-text back into the original plain-text using the same encryption key.

The algorithm consists of multiple iterations of a round cipher, each of which is the composition of the following four basic steps [57]:

- SubBytes transformation: this step is a nonlinear substitution, given by an S-box (look up table), designed to resist linear and differential cryptanalysis;
- ShiftRow transformation: provides a linear mixing for diffusion of plaintext bits;
- MixColumn transformation: provides a similar mixing as in the ShiftRow step;
- AddRoundKey transformation: bitwise XOR with the round key.

The SubBytes step

In the SubBytes step, each byte in the array is updated using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF(2^8) (Galois Field), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points and also any opposite fixed points.

The ShiftRows step

The ShiftRows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For the block of size 128 bits and 192 bits the shifting pattern is the same. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. In the case of the 256-bit block, the first row is unchanged and the shifting for second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectively – although this change only applies for the Rijndael cipher when used with a 256-bit block, which is not used for AES.

The MixColumns step

In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher. Each column is treated as a polynomial over GF(2^8) and is then multiplied modulo $x^4 + 1$ with a fixed polynomial

$c(x) = 3x^3 + x^2 + x + 2$. The MixColumns step can also be viewed as a multiplication by a particular Maximum Distance Separable (MDS) matrix in Finite field.

The AddRoundKey step

In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

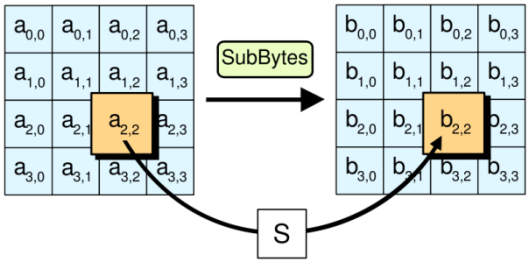

Figure 15: SubBytes Step [60].

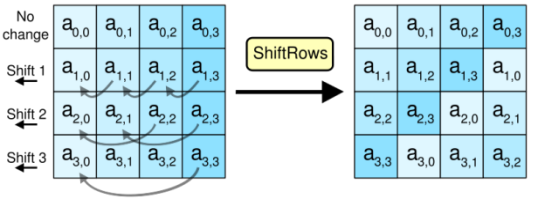

Figure 16: ShiftRows Step [60].

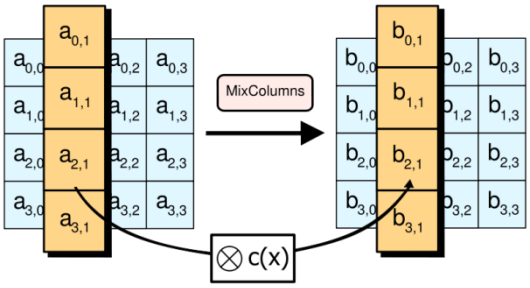

Figure 17: MixColumns Step [60].

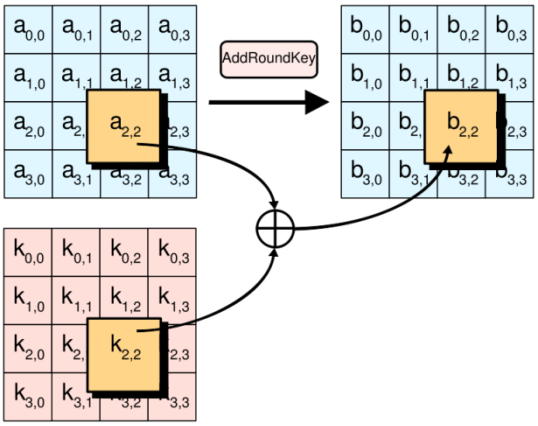

Figure 18: AddRoundKey Step [60].

Data Encryption Standard

In 1972 the NBS (National Bureau of Standards), now known as NIST, initiated a program for data protection and wanted as part of it an encryption algorithm that could be standardized. They put out a request for such an algorithm. In 1974, IBM responded with a design based on their "Lucifer" algorithm. This design would eventually evolve into the DES. DES has a key-length of 56 bits and a block-length of 64 bits. It consists of 16 rounds of what is called a "Feistel network" [61].

DES takes a 64-bit block of plain-text as input and outputs a 64-bit block of cipher-text. Since it always operates on blocks of equal size and it uses both permutations and substitutions in the algorithm, DES is both a block cipher and a product cipher. DES has 16 rounds, meaning the main algorithm is repeated 16 times to produce the cipher-text. It has been found that the number of rounds is exponentially proportional to the amount of time required to find a key using a brute-force attack. So as the number of rounds increases, the security of the algorithm increases exponentially.

The DES algorithm's overall structure is presented in *Figure 19*. There are 16 identical stages of processing, termed rounds. There is also an initial and final permutation,

designated IP (initial permutation) and FP (final permutation), which are inverses (IP "undoes" the action of FP and vice versa).

Before the main rounds, the block is divided into two 32-bit halves and processed alternately; this criss-crossing is known as the Feistel scheme. The Feistel structure ensures that decryption and encryption are very similar processes – the only difference is that the subkeys are applied in the reverse order when decrypting. The rest of the algorithm is identical. This greatly simplifies implementation, particularly in hardware, as there is no need for separate encryption and decryption algorithms.

The ⊕ symbol denotes the XOR operation. The Feistel Function, F-function, scrambles half a block together with some of the key. The output from the F-function is then combined with the other half of the block, and the halves are swapped before the next round. After the final round, the halves are not swapped; this is a feature of the Feistel structure which makes encryption and decryption similar processes.

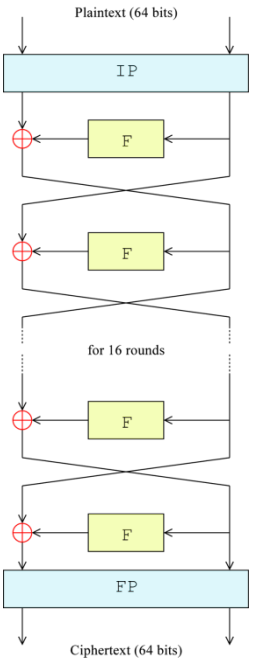

Figure 19: The overall Feistel structure of DES [62].

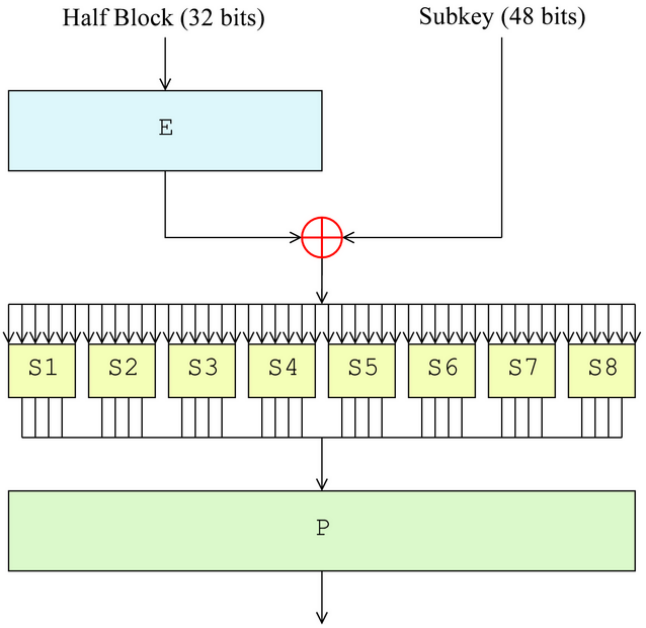

Figure 20: The Feistel function (F-function) of DES [62].

Triple-DES

In cryptography, Triple-DES, often written 3DES, is the common name for the Triple Data Encryption Algorithm. It is so named because it applies the DES cipher algorithm three times to each data block.

Triple-DES provides a relatively simple method of increasing the key size of DES to protect against brute-force attacks, without requiring a completely new block cipher algorithm.

Cipher-Block Chaining Mode

Cipher-Block Chaining mode of operation was invented by IBM in 1976 [63]. In the CBC mode, each block of plain-text is XORed with the previous cipher-text block before being encrypted. This way, each cipher-text block is dependent on all plain-text blocks processed

up to that point. Also, to make each message unique, an initialization vector (IV) must be used in the first block.

The cipher-text blocks are initialized with a randomly chosen message which may be transmitted openly, i.e., the security of the cryptosystem is based on the secrecy of the key, not on the secrecy of the IV [57].

CBC with random initial vector, is the most popular block-cipher mode of operation, used pervasively in practice [61]. Its main drawbacks are that encryption is sequential (i.e., it cannot be parallelized) and that the message must be padded to a multiple of the cipher block size. One way to handle this last issue is through the method known as cipher-text stealing.

A one-bit change in a plain-text affects all following cipher-text blocks. A plain-text can be recovered from just two adjacent blocks of cipher-text. As a consequence, decryption can be parallelized and a one-bit change to the cipher-text causes complete corruption of the corresponding block of plain-text and inverts the corresponding bit in the following block of plaintext.

Figures 21 and *22* show an example of encryption and decryption of the CBC mode of operation.

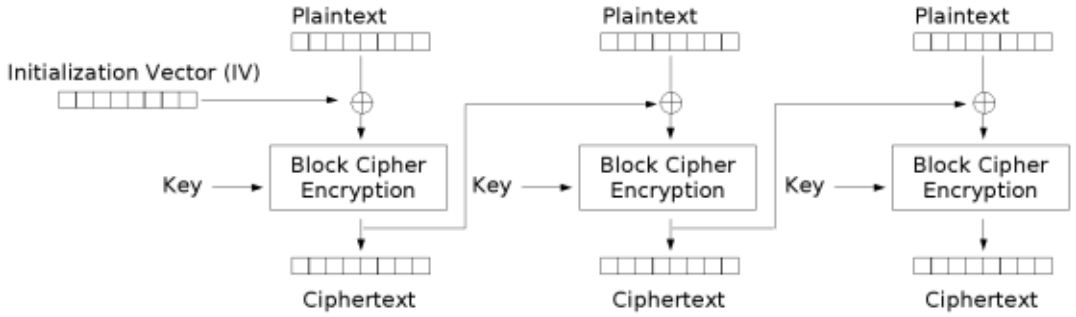

Figure 21: Cipher-Block Chaining mode encryption [64].

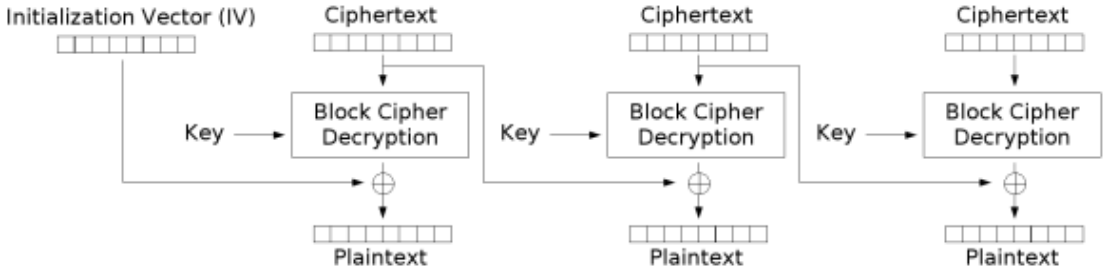

Figure 22: Cipher-Block Chaining mode decryption [64].

5.2.2. Public-key Encryption

Encryption algorithms that use different keys for encrypting and decrypting information are most often called Public-key algorithms, but are sometimes also called Asymmetric-key algorithms. Public-key encryption requires the use of both, a private-key, a key that is known only to its owner, and a public-key, a key that is available to and known to other entities on the network. The two keys are different but complementary in function. Information that is encrypted with the public-key can be decrypted only with the corresponding private-key of the set.

Figure 23 shows a basic example of encryption and decryption with asymmetric keys.

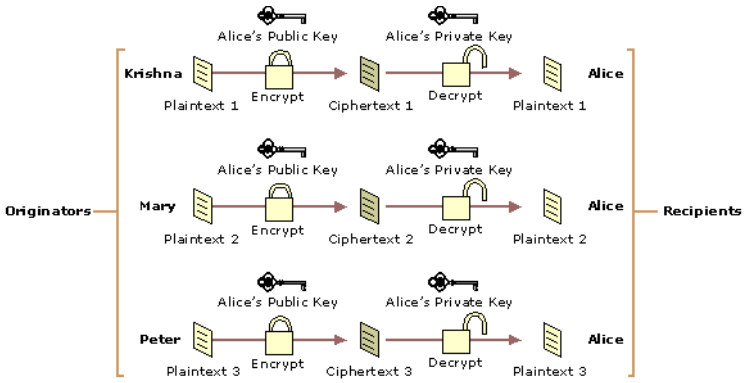

Figure 23: Encryption and decryption with a Asymmetric-key [59].

Nowadays, Public-key encryption plays an increasingly important role in providing strong and scalable security on intranets and on the Internet. However, Public-key encryption is much slower than Symmetric-key encryption, because it places a much heavier computational load on computer processors than Symmetric-key encryption. Due to the fact that symmetric ciphers are considerably faster processing, symmetric-key encryption is more suitable for IP Telephony.

Public key encryption is commonly used to perform the following functions [59]:

- Encrypt symmetric secret keys to protect the symmetric keys during the exchange over the network or while being used, stored or cached by operating systems;
- Create digital signatures to provide authentication and non-repudiation for online entities;
- Create digital signatures to provide data integrity for electronic files and documents.

The Diffie-Hellman and RSA algorithms are among the most widely used high quality public-key algorithms. Others include the Cramer-Shoup cryptosystem, ElGamal encryption and various elliptic curve techniques.

In addition to encryption, public-key cryptography can be used to implement digital signature schemes. A digital signature is reminiscent of an ordinary signature; they both have the characteristic that they are easy for a user to produce, but difficult for anyone else to forge. Digital signatures can also be permanently tied to the content of the message being signed. They cannot then be "moved" from one document to another, for any attempt will be detectable. In digital signature schemes, there are two algorithms: one for signing, in which a secret-key is used to process the message and one for verification, in which the matching public-key is used with the message to check the validity of the signature. RSA and Digital Signature Algorithm (DSA) are two of the most popular digital signature schemes. Digital signatures are central to the operation of public-key infrastructures and many network security schemes (SSL/TLS and many VPNs, for example) [65].

5.3. Internet Protocol Security

Internet Protocol Security (IPsec) is a suite of protocols for securing IP communications by authenticating and encrypting each IP packet of a data stream. IPsec also includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used to protect data flows between a pair of hosts (computer users or servers, for example),

between a pair of security gateways (routers or firewalls, for example) or between a security gateway and a host [66].

Summarizing, the objective of IPSec is to provide security services for IP packets at the network layer. These services include access control, data integrity, authentication, protection against replay and data confidentiality.

A common misconception about IPSec is that it is a single protocol for providing these security services for IP traffic. In fact, IPSec is really a suite, or collection, of protocols for security defined by the IPSec working group in the IETF. The baseline IPSec architecture and fundamental components of IPSec are defined in RFC2401 [67] as the following:

- Security protocols Authentication Header and Encapsulation Security Payload;
- Key management ISAKMP, IKE and SKEME;
- Algorithms for encryption and authentication.

5.3.1. IPsec Security Protocols

As mentioned above, IPsec is compounded by several protocols. Among all the IPsec suite of protocols there are two specific protocols to provide traffic security:

- Authentication Header (AH);
- Encapsulating Security Payload (ESP).

Authentication Header

AH provides connectionless integrity, data authentication and optional replay protection but, unlike ESP, it does not provide confidentiality, so AH is used to authenticate, but not encrypt, IP traffic. Consequently, it has a much simpler header than ESP. *Figure 24* shows an AH-protected IP packet.

Authentication is performed by computing a cryptographic hash-based message authentication code over nearly all the fields of the IP packet, and stores this in a newly-added AH header and sent to the other end.

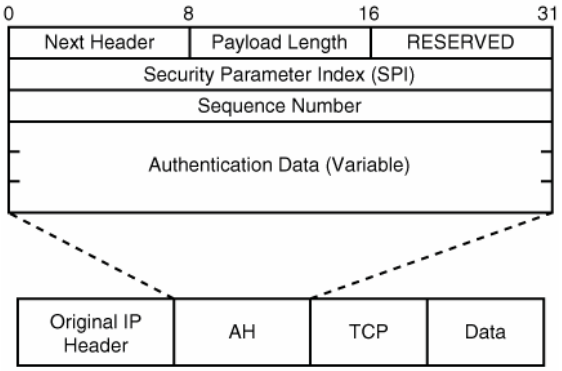

Figure 24: IP packet protected by AH [9].

AH is an IP protocol, identified by a value of 51 in the IP header. The Next header field indicates what follows the AH header.

Encapsulation Security Payload

ESP provides confidentiality, data integrity, and optional data origin authentication and anti-replay services. It provides these services by encrypting the original payload and encapsulating the packet between a header and a trailer, as shown in *Figure 25*.

The IPsec RFCs don't insist upon any particular encryption algorithms, but DES, 3DES, AES and Blowfish are the typically used ones to shield the payload from prying eyes.

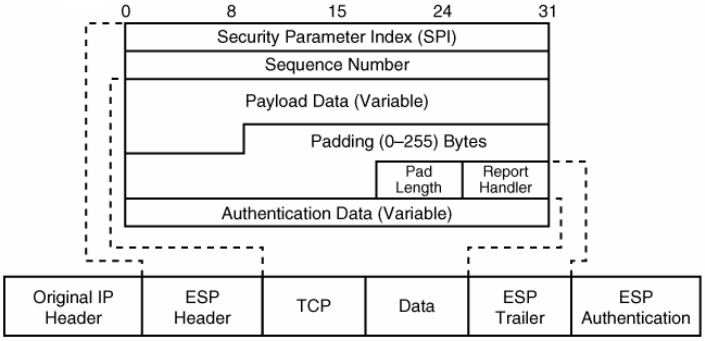

Figure 25: IP packet protected by ESP [9].

ESP is identified by a value of 50 in the IP header. The ESP header is inserted after the IP header and before the upper layer protocol header.

5.3.2. IPsec Modes

IPsec may operate in two distinct ways, the transport and tunnel mode, depending upon whether the secure communication is between two endpoints directly connected (in which case it operates in transport mode) or between two intermediate gateways to which the two endpoints are connected via a clear, i.e., unencrypted, channel (in which case IPsec operates in tunnel mode) [53].

IPsec Transport Mode

In transport mode, an IPSec header (AH or ESP) is inserted between the IP header and the upper layer protocol header.

In this mode, the IP header is the same as that of the original IP packet except for the IP protocol field, which is changed to ESP (50) or AH (51), and the IP header checksum, which is recalculated. IPsec assumes the IP endpoints are reachable. In this mode, the destination IP address in the IP header is not changed by the source IPsec endpoint; therefore, this mode can only be used to protect packets in scenarios in which the IP endpoints and the IPsec endpoints are the same.

IPsec Tunnel Mode

In tunnel mode, the original IP packet is encapsulated in another IP datagram and an IPSec header (AH or ESP) is inserted between the outer and inner headers. Because of this encapsulation with an "outer" IP packet, tunnel mode can be used to provide security services between sites on behalf of IP nodes behind the gateway router at each site. Also, this mode can be used for the telecommuter scenario of connecting from an end host to an IPSec gateway at a site.

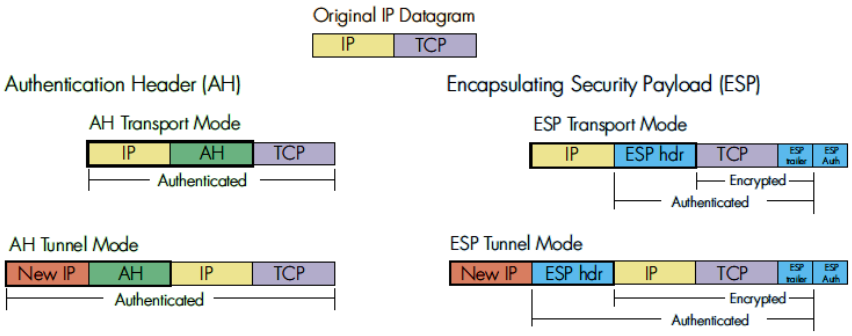

Figure 26: IP packet protected by AH or ESP, in Transport and Tunnel modes.

5.3.3. The Role of IPsec in VoIP

The prevalence and easiness of packet sniffing and other techniques for capturing packets on an IP-based network makes encryption a necessity for VoIP. Security in VoIP is concerned both with protecting what a person says, as well as to whom the person is speaking. IPsec can be used to achieve both of these goals as long as it is applied with ESP using the tunnel method. This secures the identities of both the endpoints and protects the voice data from prohibited users once packets leave the private Intranet. VoIPsec (VoIP using IPsec) helps reduce the threat of man-in-the-middle attacks, packet sniffers and many other types of voice traffic analysis. IPsec makes VoIP more secure than a standard phone line, where people generally assume the need for physical access to tap a phone line is deterrent enough. It is important to note, however, that IPsec is not always a good fit for some applications, so some protocols will continue to rely on their own security features.

5.3.4. IPsec and NAT Incompatibility

IPsec and NAT compatibility is far from ideal. NAT traversal completely invalidates the purpose of AH because the source address of the machine behind the NAT is masked from the outside world. Thus, there is no way to authenticate the true sender of the data. The same reasoning demonstrates the inoperability of source authentication in ESP. There are several other issues that arise when ESP traffic attempts to cross a NAT. If only one of the endpoints is behind a NAT, the situation is easier [68]. If both are behind NATs, IKE negotiation can be used for NAT traversal, with UDP encapsulation of the IPsec packets.

5.4. Transport Layer Security

Transport Layer Security [69] (TLS) is a cryptographic protocol that provide secure communications on the Internet, for such applications as web browsing, e-mail, Internet faxing, instant messaging, IP Telephony and other data transfers, i. e., provides communications privacy over the Internet. TLS is a security protocol from the IETF that is based on the Secure Sockets Layer (SSL) 3.0 protocol developed by Netscape [70]. TLS encrypt the segments of network connections at the Transport Layer end-to-end (*Figure 27*) and is composed by two protocols: the TLS Handshake Protocol and the TLS Record Protocol.

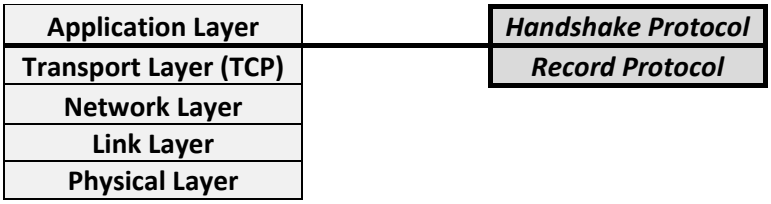

Figure 27: The place of TLS in the Internet Protocol stack.

The TLS protocol allows client/server applications to communicate across a network in a way designed to prevent eavesdropping, tampering and message forgery. TLS provides endpoint authentication and communications confidentiality over the Internet using cryptography.

5.4.1. Establishing an Encrypted Communication

Setting up a communication with TLS always includes a client and a server, and it starts with a Handshake. In this initial state the terms for the data-exchange shall be specified into these steps:

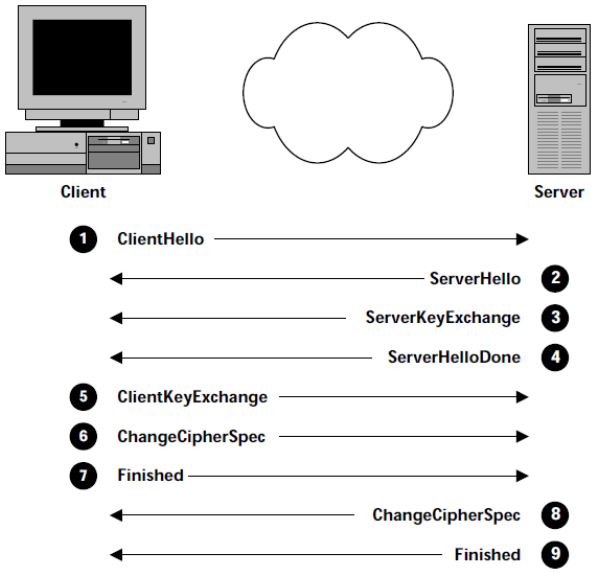

Figure 28: TLS uses 9 messages to establish encrypted communications [71].

Table 8: Negotiation of TLS encrypted communications.

Step	Action
❶	Client sends *ClientHello* message proposing TLS options;
❷	Server responds with *ServerHello* message selecting the TLS options;
❸	Server sends its public key information in *ServerKeyExchange* message;
❹	Server concludes its part of the negotiation with *ServerHelloDone*;
❺	Client sends session key information (encrypted with server's public key) in *ClientCipherSpec* message;
❻	Client sends *ChangeCipherSpec* message to activate the negotiated options for all future messages it will send.

Hello Message

First the client and the server exchange information to reach an agreement upon what cryptographic and compression algorithms to use and share random numbers for generation of keys (*ClientHello* and *ServerHello*). In this step of the communication is also given a unique session ID that allows the reuse of keys for a certain period of time.

Certificate and Key Exchange

In the next step, the server and sometimes also the client, identify themselves using a certificate. The keys are derived both at the server and the client side.

Change Cipher Spec

When the certificate steps are concluded a cipher change message (*ChangeCipherSpec*) will be exchanged, which will inform both parts that it is time to change to the symmetric key and start the real transmission. A check will also be conducted to reassure that no tampering has been made.

If a client reconnects to a server running TLS before the session has expired, the client sends the session ID to indicate it wants to resume. The server can then reopen the communication with the use of the keys derived earlier.

One advantage of TLS is that it is an application protocol independent. Higher level protocols can layer on top of the TLS Protocol transparently. The TLS standard, however, does not specify how protocols add security with TLS; the decisions on how to initiate TLS handshaking and how to interpret the authentication certificates exchanged are left up to the judgment of the designers and implementers of protocols which run on top of TLS [72].

However, the typical algorithms, used in TLS communications, are:

- For key exchange: Diffie-Hellman, ECDH, PSK, RSA or SRP;
- For authentication: DSA, ECDSA or RSA;
- Symmetric ciphers: AES, Camellia, DES, IDEA, RC4 or Triple-DES.

For cryptographic hash function is used HMAC-MD5 or HMAC-SHA.

5.4.2. TLS Handshake Protocol

The TLS Handshake Protocol consists of a suite of three sub-protocols which are used to allow peers to agree upon security parameters for the record layer, authenticate themselves, instantiate negotiated security parameters and report error conditions to each other. The three sub-protocols that compose the TLS Handshake Protocol are the Change cipher spec, the Alert and the Handshake Protocols.

Most of the TLS specifications describe the Handshake Protocol, as it is the one primarily responsible for negotiating TLS sessions. The Handshake Protocol relies on the Record Layer to encapsulate its messages (*Figure 29*).

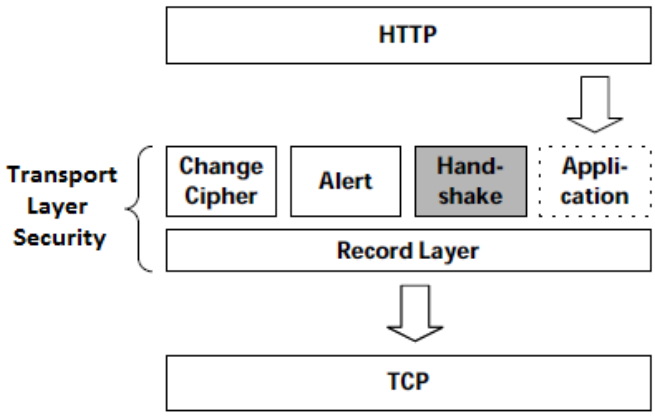

Figure 29: The Handshake protocol deals with dession negotiation.

Change cipher spec Protocol

The Change cipher spec Protocol exists to signal transitions in ciphering strategies, as described into the section **Change Cipher Spec**. The protocol consists of a single message, which is encrypted and compressed under the current (not the pending) connection state.

Alert Protocol

The alert messages convey the severity of the message and a description of the alert. The alert can be of two types: closure alert or error alert; where the former is an assurance that both server and client know that the connection is ending, and the latter is used for error reporting.

Handshake Protocol

This protocol consists of a set of messages that signals transitions in ciphering strategies, i. e., is used to negotiate the secure attributes of a session, as it's described into the section **Establishing an Encrypted Communication**. Handshake messages are supplied to the TLS Record Layer, where they are encapsulated within one or more TLS Plain Text structures, which are processed and transmitted as specified by the current active session state.

The TLS Handshake Protocol provides connection security that has three basic properties [72]:

- The peer's identity can be authenticated using asymmetric or public key, cryptography (RSA or DSS, for example). This authentication can be optional made, but is generally required for at least one of the peers;
- The negotiation of a shared secret is secure, i. e., the negotiated secret is unavailable to eavesdroppers, and for any authenticated connection the secret cannot be obtained, even by an attacker who can place himself in the middle of the connection;
- The negotiation is reliable, which means, no attacker can modify the negotiation communication without being detected by the parties of the communication.

5.4.3. TLS Record Protocol

The TLS Record Protocol is a layered protocol. At each layer, messages may include fields for length, description and content. The Record Protocol takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC,

encrypts and transmits the result. Received data is decrypted, verified, decompressed and reassembled, then delivered to higher level clients.

The TLS Record Protocol is used for encapsulation of various higher level protocols.

The TLS Record Protocol provides connection security that has two basic properties [72]:

- The connection is private. Symmetric cryptography is used for data encryption (for example DES or RC4) and, the keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol). The Record Protocol can also be used without encryption;
- The connection is reliable. Message transport includes a message integrity check using a keyed MAC. Secure hash functions (SHA or MD5, for example) are used for MAC computations. The Record Protocol can operate without a MAC, but is generally only used in this mode while another protocol is using the Record Protocol as a transport for negotiating security parameters.

A TLS connection state is the operating environment of the TLS Record Protocol. It specifies a compression algorithm, encryption algorithm and MAC algorithm. In addition, the parameters for these algorithms are known: the MAC secret, the bulk encryption keys and initialization vectors (IV) for the connection in both the read and the write directions. Logically, there are always four connection states outstanding: the current read and write states and the pending read and write states. All records are processed under the current read and write states. The security parameters for the pending states can be set by the TLS Handshake Protocol, and the Handshake Protocol can selectively make either of the pending states current, in which case the appropriate current state is disposed of and replaced with the pending state; the pending state is then reinitialized to an empty state. It is illegal to make a state which has not been initialized with security parameters a current state. The initial current state always specifies that no encryption, compression or MAC will be used.

6 USED TECHNIQUES OF MEASUREMENT

6.1. Introduction

This chapter presents the different paths that were taken to reach the final goals, with special care to the used techniques of measurement. It is also presented a detailed description of the implementation of the distinct testbeds.

In first section, **Bandwidth Emulation**, it is exposed the initial bandwidth conditions and the used mechanism to limit the bandwidth of Internet connection. **Connection Characteristics** section presents the way how to behold Ostrava – Prague connection specifications and attributes. The **Testbed Environments** section describes the distinct created scenarios. Finally, the section **IxChariot Test Tool** describes step-by-step all the treads taken in order to perform the VoIP simulation tests; this last section is focused on the IxChariot VoIP test parameters and measurement process.

The obtained results, presented in the chapter **8**, are based on several series of measurements, which have been performed in a real network between Ostrava (158.196.81.208) and Prague (147.32.201.116), Czech Republic. For that purpose were assigned two machines running Ubuntu Linux distributions plus another one with IxChariot Console installed; every with a public assigned IP address.

IxChariot, a tool presented in chapter **2**, consists in the IxChariot Console and IxChariot Performance Endpoints. The IxChariot Console, installed on the machine with 158.196.81.204 set IP, allows a selection of several test configurations, such as used codec, timing, number of concurrent calls, test duration and so on.

The public IP addresses 158.196.81.204 and 158.196.81.208 were assigned to Vysoká Škola Báňská - Technická Univerzita Ostrava Katedry Telekomunikační [73], as well as 147.32.201.116 was assigned to České Vysoké Učení Technické v Praze Katedry Telekomunikační [74], by IANA [75].

The used versions of IxChariot Console and IxChariot Performance Endpoints were the Version 6.50 (Retail) Build 458 and Version 6.50 (Retail) Build 395, respectively. The IxChariot Performance Endpoints were installed in both running Ubuntu machines.

6.2. Bandwidth Emulation

The Ostrava – Prague Universities Internet connection is provided by CESNET [76]. CESNET, association of legal entities, was created in 1996 by all universities of Czech Republic and Czech Academy of Sciences, and is a Czech academic network operator. The contemporary CESNET network is built around a DWDM core, which means an extremely good Internet connection between Ostrava and Prague machines. As the scope of this dissertation is focused on comprehend and relate how secure network environments and mechanisms can affect Speech Quality, a limited bandwidth is required in a way to reach accurate and meaningful results.

Therefore, resorted to the use of Simena Network Emulator (NE1000) [77]. A Network Emulator enables to determine how a product or service would perform under various network conditions such as bandwidth, latency, congestion and so on. It emulates these network conditions by transparently capturing and processing the data packets. Network Emulator can be used for any network protocol. Since it operates at Ethernet layer it does not require any configuration change on networks or servers.

Simena Network Emulator emulates bandwidth by throttling the transmission speed at each interface. The specific set bandwidth was 2 Mbps, as can be observed in *Figure 30*.

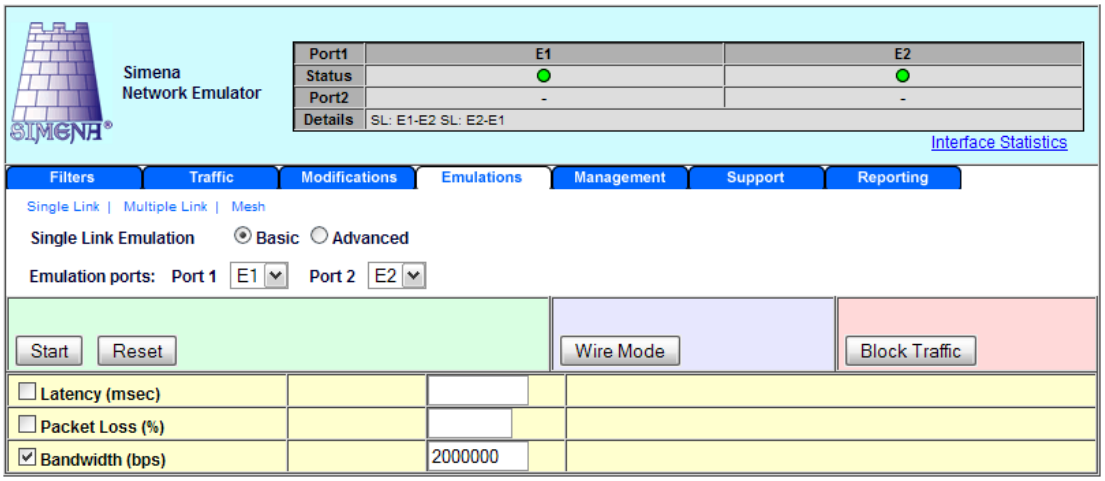

Figure 30: Simena NE1000 bandwidth emulation.

To compare the previous bandwidth with bandwidth after configured the network emulator was used the conjugation of Iperf and JPerf tools. Taking advantage of the fact that both applications can interact between them, on Ostrava machine was used JPerf tool and on the other side, Prague, was used Iperf, in order to achieve the bandwidth of the connection before and after turn on Simena Network Emulator.

On the Prague machine, Iperf server side, was executed on the Linux Terminal the command showed below:

```
~# iperf -s
```

This instruction means that Iperf server will be listening on TCP port 5001. By default, Iperf client connects to Iperf server on TCP port 5001 and the bandwidth displayed by Iperf is the bandwidth from client to server.

On the other hand, at Ostrava machine, Iperf client side, in JPerf application was performed the following instruction:

```
iperf -c 147.32.201.116 -P 1 -i 1 -p 5001 -f m -t 10
```

The above presented instruction means Iperf client will perform a connection to the machine with IP address 147.32.201.116 on TCP port 5001 (default). The "-P" argument means the number of parallel streams, "-i" indicates the interval in seconds between periodic bandwidth reports, "-p" the listen port, "-f" the output format ("m" represents MBits) and, finally, "-t" specifies the test duration time in seconds.

The results of this procedure can be found into the subsection **8.2**.

6.3. Connection Characteristics

Round-trip time (RTT), also called round-trip delay (RTD), is the time required for a packet to travel from a specific source to a specific destination and back again. In this context, the source is the computer initiating the signal and the destination is a remote computer or system that receives the signal and retransmits it. On other words, RTD can be

interpreted by a measure of the current delay on a network, found by timing a packet bounced off some remote host.

Knowing RTD, it is possible to calculate the delay between the two IxChariot Performance Endpoints, Ostrava and Prague machines. The one way symmetric VoIP segment delay (**1**) may be calculated from round-trip and end system delays; if the round-trip delay is denoted, and the end system delays associated with the two endpoints ESD(A) and ESD(B), there is [78]:

$$\text{One Way Symmetric Voice Path Delay} = \frac{\text{RTD}}{2} + \text{ESD(A)} + \text{ESD(B)} \qquad (\mathbf{1})$$

where ESD(A) $_{[ms]}$ is the estimated system delay in milliseconds, related to the coder delay (look-ahead delay) and packetization delay in transmitter for the used codec, and ESD(B) $_{[ms]}$ the dejitter delay in milliseconds plus $0.1 \cdot \text{ESD(A)}$, the de-packetization delay and decompression delay in the receiver. It should be noted that equation (**1**) is an improvement of the original one [78], that only takes in account half of required time for packetizing and de-packetizing, compression and decompression, and dejitter buffer delay.

The specified dejitter buffer was 60 milliseconds. The default ESD(A) values for the used codecs are the following:

- Codec G.711 A-law, ESD(A)=20 ms;
- Codec G.729, ESD(A)=25 ms.

On the Internet, an end user can determine the RTT to and from an IP address by using the Ping and Traceroute network tools.

Ping is a computer network tool used to test whether a particular host is reachable across an IP network, to self test the network interface card of the computer and can also be used as a speed test. Ping works by sending ICMP "echo request" packets to the target host and listening for ICMP "echo response" replies. Ping measures RTT and records the packet loss, printing when finished a statistical summary of the echo response packets received, the minimum RTT, RTT average, maximum RTT and, in case of the Linux and Mac OS X versions, the standard deviation of RTT.

Traceroute, also a computer network tool, is used to determine the route taken by packets between two systems across an IP network, listing all the intermediate routers where a packet must pass through to reach its final destination. Traceroute works by causing each router along a network path to return an ICMP error message. An IP packet contains a time-to-live (TTL) value which specifies how long it can go on its search for a destination before being discarded. Each time a packet passes through a router, its TTL value is decremented by one; when it reaches zero, the packet is dropped and an ICMP Time-To-Live Exceeded error message is returned to the sender. On traceroute packets the maximum TTL value is 30 (default).

In order to determine the RTT, as well as the routes taken by the traffic between Ostrava and Prague machines were used Ping and Traceroute network tools.

Ping

The Ping network tool was used in both machines. For that were performed at each machine's Linux Terminal the following commands:

Ostrava Machine

```
~# ping -c 5 -I 158.196.81.208 147.32.201.116
```

The above presented instruction means that will be performed a "ping" to destination IP address 147.32.201.116. The "-c" argument means the number of sending ICMP "echo request" packets and "-I" indicates the source interface address, argument that can be the numeric IP address or the name of the interface device.

Prague Machine

```
~# ping -c 5 -I 147.32.201.116 158.196.81.208
```

The same instruction was executed on the Prague machine, but with Ostrava machine IP address as destination.

Traceroute

The network tool Traceroute was also used in both machines in order to achieve the RTT and the routes taken by traffic between Ostrava and Prague machines.

On the Ostrava side was performed on the Linux Terminal the "traceroute" instruction from the source address of Ostrava machine to Prague machine IP address. The same command was used on the Prague machine, but with reverse IP addresses.

Ostrava Machine

```
~# traceroute -s 158.196.81.208 147.32.201.116
```

Prague Machine

```
~# traceroute -s 147.32.201.116 158.196.81.208
```

The results of "ping" and "traceroute" instructions can be found into the subsection **8.3**.

6.4. Testbed Environments

Concerning narrowly to the testbed environments, were intended four distinct scenarios:

- Unsecure connection;
- Openswan IPsec secure connection with 128 bits AES default key cipher;
- Openswan IPsec secure connection with 192 bits Triple-DES default key cipher;
- OpenVPN TLS secure connection with 128 bits AES default key cipher;
- OpenVPN TLS secure connection with 192 bits Triple-DES default key cipher;
- Openswan IPsec plus OpenVPN TLS secure connection with 128 bits AES default key ciphers;
- Openswan IPsec plus OpenVPN TLS secure connection with 192 bits Triple-DES default key ciphers.

For each distinct environment were performed various VoIP simulation tests, and consequently the respective measurements, through the use of IxChariot tool. Parameters as the used codec, its characteristics and the number of concurrent calls were especially taken into account for each different test environment.

The used codecs during the performance of VoIP tests were the following:

- G.711 A-law (64 Kbps);
- G.729 (8 Kbps).

G.711 is an ITU-T standard for audio companding. It is primarily used in telephony. G.711 represents logarithmic PCM samples for signals of voice frequencies. There are two main compression algorithms defined in the standard, the μ-law algorithm (used in North America & Japan) and A-law algorithm (used in Europe and the rest of the world). Both are logarithmic, but A-law was specifically designed to be simpler for a computer to process [79]. G.711 A-law has a 64 Kbps bit rate (8 KHz sampling frequency x 8 bits per sample). G.711 Appendix I defines a Packet Loss Concealment (PLC) algorithm to help hide transmission losses in a packetized network.

G.729, also known as CS-ACELP, is as well specified by ITU-T [80]. G.729 is an audio data compression algorithm for voice that compresses voice audio in frames of 10 milliseconds, or 80 samples, in duration. It compresses speech from 16 bit, 8 KHz samples (128 Kbps) to 8 Kbps and was designed for cellular and networking applications. G.729 provides "toll quality" speech, works well with background noise, has been designed to perform well under error conditions and is mostly used in VoIP applications for its low bandwidth requirement. It is one of the main contenders for the baseline codec for Internet Telephony.

The following subsections will briefly present the different projected network environments.

6.4.1. Unsecure Connection Testbed

A basic scheme of the test environment is presented in *Figure 31*. The base of all distinct testbed is compound of a laptop (158.196.81.204) with IxChariot Console Version 6.50 (Retail) Build 458, two IxChariot Performance Endpoints correspondents to Ostrava (Endpoint 1) and Prague (Endpoint 2) machines, a Simena Network Emulator NE1000 with bandwidth emulation set to 2 Mbps and, finally, another laptop connected to a 10Base-T (10Mbps) D-Link DE-809TC Ethernet Hub running Wireshark in order to capture the whole traffic carried through the network between the two Endpoints.

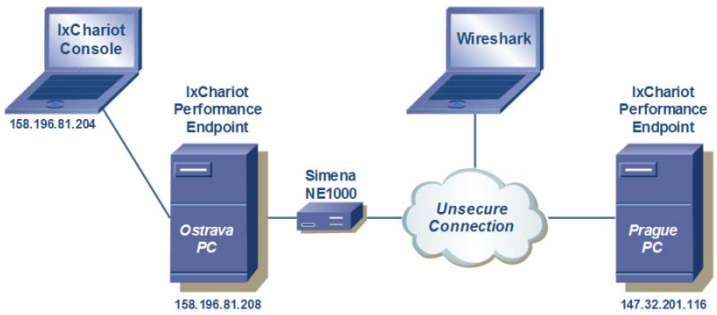

Figure 31: Logical scheme of testbed – Unsecure connection.

On the above presented testbed scheme none security mechanism were used, with the whole traffic between IxChariot Performance Endpoints going thru the traditional unsecure Internet.

Whole traffic between the two distinct Endpoints was captured by Wireshark and individual packets were analyzed.

6.4.2. Openswan IPsec Secure Connection Testbed

Figure 32 presents a basic scheme of the test environment for an IPsec secure connection. The whole testbed is in everything similar to the used one for the case of an unsecure connection, except on the security implemented mechanism.

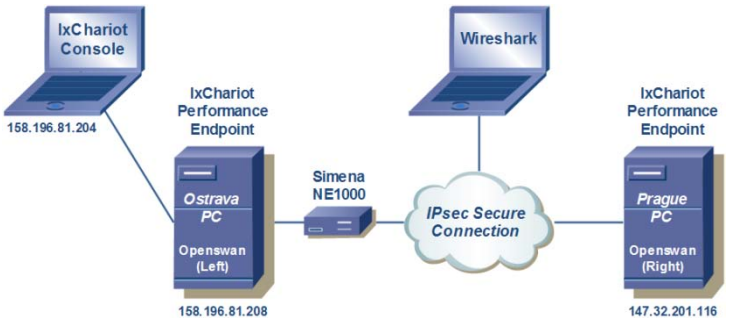

Figure 32: Logical scheme of testbed – IPsec secure connection.

Openswan was the chosen mechanism to implement an IPsec secure connection between IxChariot Performance Endpoints. The whole Openswan installation and configuration steps are presented in Appendix **B.2**.

Openswan configuration file

The basic used template "/etc/ipsec.conf" comprising all gathered information and Openswan configurations is presented below:

```
# /etc/ipsec.conf - Openswan IPsec configuration file
# RCSID $Id: ipsec.conf.in,v 1.15.2.6 2006/10/19 03:49:46 paul Exp $

# This file:  /usr/share/doc/openswan/ipsec.conf-sample
#
# Manual:     ipsec.conf.5


version 2.0     # conforms to second version of ipsec.conf specification

# basic configuration
config setup
        # plutodebug / klipsdebug = "all", "none" or a combation from below:
        # "raw crypt parsing emitting control klips pfkey natt x509 private"
        # eg: plutodebug="control parsing"
        #
        # ONLY enable plutodebug=all or klipsdebug=all if you are a developer !!
        #
        # NAT-TRAVERSAL support, see README.NAT-Traversal
        nat_traversal=yes
        # virtual_private=%v4:10.0.0.0/8,%v4:192.168.0.0/16,%v4:172.16.0.0/12
        #
        # enable this if you see "failed to find any available worker"
        nhelpers=0
```

```
# Add connections here

# sample VPN connections, see /etc/ipsec.d/examples/

#Disable Opportunistic Encryption
include /etc/ipsec.d/examples/no_oe.conf

conn host-to-host
  left=158.196.81.208                          # Local IP address
  leftsubnet=158.196.81.0/24                   # Local netmask
  leftid=@server.com                           #

leftrsasigkey=0sAQPGl/wJ+AMpDehHdRjs0vZRWRig5Gu4nPrfh9OLkojJwznGRlO/qCWVK32O
y5CnyMG5tz1knruOW8FSff4utWoYGc+tmnQM15h593M2/BU3ubNlV/kIOHhs4eGNzDNODxx
CYCIv4w4HBHrfQLodqG9EOrGmb02AfuTrmxjgSQM/ci2G91k5QVdSWDvFyl+UB/U2LRohTLtZi
9nbDCYf7eud9BXJHxi2BTqjLM61as+871tT04vMbDr4NaFX83NuxOtUYgFsxw7jy5aHRiYD+3z4
UCdZeG6p2pPibHx4TqfluiT582TIEfh69JCPc8JUtzkkjt0TtHrsuTGbFsn/n1wl
  leftnexthop=%defaultroute                    # Correct in many situations
  right=147.32.201.116                         # Remote IP address
  rightsubnet=147.32.201.0/24                  # Local netmask
  rightid=@clientPrague.com                    #

rightrsasigkey=0sAQPpWAI3RsIGnpnjshI2HGq46iN0htpEl5YQ3BsM/rfUUnQuCi1LruE2wmDz
DGpGxzwsWq7gkhgWsT7G7I75uxxk4MfBAHPIhrUsxYTmTOU+YKnPiWXPFYnllysYqyQsoqav/6
s07kkCBzgRmgkSjqv1eVCwmJaLD8vj7jQkxocFa8dRT8lRItnwEIjBBwp7j8KiBgRU6ivNdDrYxAuU
0Mq5LW+hkCEbaUP0CMOj0TNcVuqRDCKHqO+HkNVCOPPg+TaDWE8Eb26j8FvNCu/ero4vTY
K57aHq/8g/3LMcqqzsZH2bneyjVe6+gsOJLyzV7ktAXPgFh+ObkdacxtNdlcIL
  rightnexthop=%defaultroute                   # Correct in many situations
  ike=aes128                                   # IKE algorithms (AES cipher)
  esp=aes128                                   # ESP algorithns (AES cipher)
  #ike=3des                                    # IKE algorithms (3DES cipher)
  #esp=3des                                    # ESP algorithns (3DES cipher)
  auto=start                                   # enables the connection at startup
  auth=ah                                      #

# Lines starting with # will not be read by Openswan (ipsec-tools)
```

NOTE: "Left" and "right" should represent the machines that have Openswan installed on them, i. e., the machines with IxChariot Performance Endpoints.

The used ciphers by Openswan IPsec were AES (128 bits block size) with 128 bits key size and Triple-Des (64 bits block size) with 192 bits key size.

6.4.3. OpenVPN TLS Secure Connection Testbed

In *Figure 33* is presented a basic scheme of the test environment for a TLS secure connection. The whole testbed is also in everything similar to the used one for the case of an unsecure connection, except on the security implemented mechanism.

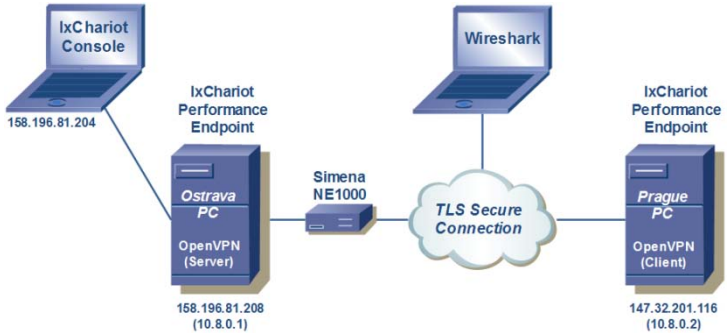

Figure 33: Logical scheme of testbed – TLS secure connection.

OpenVPN was the chosen mechanism to implement a TLS secure connection between IxChariot Performance Endpoints.

The whole OpenVPN installation and configuration steps are presented in Appendix **B.3**. TLS involves three main phases such as negotiation of supported algorithms, keys exchange and authentication, and in the end the symmetric encryption of transmitted data. The endpoints establishing the VPN tunnel are declared one as server (Ostrava machine) and other as client (Prague machine). Before establishing the VPN, the client first reaches the server on a specific port (1194), whereas the server does not need to reach the client. Configuration files are located in directory "/etc/openvpn/" as "server.conf" or "client.conf". The tunnel can be established on UDP or TCP, unfortunately TCP protocol is more widespread, although UDP should be more effective because of real-time applications. The most important information in configuration files is the type of cipher alghoritm, because it affects the number of blocks and overhead.

The used ciphers by OpenVPN TLS were AES (128 bits block size) with 128 bits key size and Triple-Des (64 bits block size) with 192 bits key size.

Several VoIP tests were performed for the OpenVPN TLS solution. In addition to varying the used cipher, it was also tested an OpenVPN TLS connection with and without LZO compression/decompression mode, what is represented by the instruction "comp-lzo" in the configuration files. "comp-lzo" option when added means compression on the VPN link, i. e., compression of the data stream.

Lempel-Ziv-Oberhumer [81], standing for LZO, is a lossless data compression algorithm that is focused on decompression in real-time. This means it favours speed over compression ratio. LZO is a block compression algorithm – it compresses and decompresses a block of data. Block size must be the same for compression and decompression. LZO compresses a block of data into matches and runs of non-matching literals. LZO takes care about long matches and long literal runs so that it produces good results on highly redundant data and deals acceptably with non-compressible data. When dealing with uncompressible data, LZO expands the input block by a maximum of 16 bytes per 1024 bytes input. LZO algorithms and implementations are distributed under the terms of the GNU GPL [7].

OpenVPN configuration files

Server configuration file

The archive "/etc/openvpn/server.conf" is presented below:

```
local 158.196.81.208                    # Local IP address that OpenVPN should listen on
dev tun0                                # Routed IP tunnel
ifconfig 10.8.0.1 10.8.0.2              # Tunnel IP addresses (Server – Client)
tls-server                              # TLS Server
proto tcp-server                        # TCP Server
port 1194                               # TCP/UDP port that OpenVPN should listen on

ca /etc/openvpn/keys/ca.crt             # SSL/TLS root certificate
cert /etc/openvpn/keys/server.crt       # Certificate
key /etc/openvpn/keys/server.key        # Private key (this file should be kept secret)
dh /etc/openvpn/keys/dh1024.pem         # Diffie-Hellman parameters

keepalive 10 120                        # Ping every 10 seconds, assume that remote peer
                                        # is down if no ping received during a 120
                                        # second time period.

;cipher AES-128-CBC                     # AES cipher - 128 bit default key (fixed)
;cipher DES-EDE3-CBC                    # 3DES cipher - 192 bit default key (fixed)

auth none                               #
comp-lzo                                # Enable compression on the VPN link
max-clients 100                         # Assign the maximum number of clients

# The persist options will try to avoid accessing certain resources on restart that may no
# longer be accessible because of the privilege downgrade
persist-key
persist-tun

# Output a short status file showing current connections, truncated and rewritten every
# minute.
status openvpn-status.log

# Set the appropriate level of log file verbosity (3 - reasonable for general usage)
verb 3

# Lines starting with # or ; will not be read by OpenVPN
```

Client configuration file

The archive "/etc/openvpn/client.conf", on the client side (Prague machine), has the following content:

```
dev tun0                                # Routed IP tunnel
remote 158.196.81.208                   # IP address of the Server
ifconfig 10.8.0.2 10.8.0.1              # Tunnel IP addresses (Client – Server)
tls-client                              # TLS Client
proto tcp-client                        # TCP Client
port 1194                               # TCP/UDP port that OpenVPN should listen on


ca /etc/openvpn/keys/ca.crt             # SSL/TLS root certificate
cert /etc/openvpn/keys/client.crt       # Certificate
key /etc/openvpn/keys/client.key        # Private key (this file should be kept secret)
dh /etc/openvpn/keys/dh1024.pem         # Diffie-Hellman parameters

keepalive 10 120                        # Ping every 10 seconds, assume that remote peer
                                        # is down if no ping received during a 120
                                        # second time period.

;cipher AES-128-CBC                     # AES cipher - 128 bit default key (fixed)
;cipher DES-EDE3-CBC                    # 3DES cipher - 192 bit default key (fixed)

auth none                               #
comp-lzo                                # Enable compression on the VPN link

# The persist options will try to avoid accessing certain resources on restart that may no
# longer be accessible because of the privilege downgrade
persist-key
persist-tun

# Output a short status file showing current connections, truncated and rewritten every
# minute.
status openvpn-status.log

# Set the appropriate level of log file verbosity (3 - reasonable for general usage)
verb 3

# Lines starting with # or ; will not be read by OpenVPN
```

Packet Forwarding and Routing

After set the OpenVPN configurations a tunnel interface, tun0, is created and in order to have packet forwarding and the correct routes between IxChariot Console and Performance Endpoints, some instructions were performed each time that OpenVPN was started or even restarted.

Ostrava and Prague machines

The archive "/proc/sys/net/ipv4/ip_forward" is responsible for packet forwarding. The packet forwarding can be enabled at system boot by adding the line "net.ipv4.ip_forward = 1" on the file "/etc/sysctl.conf". To enable packet forwarding without rebooting, as root should be performed the following command:

```
~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Prague machine

After enabled packet forwarding, a route must be added in order to forward all data from IxChariot Console machine (158.196.81.204) through the created tunnel, tun0.

Route manipulates the kernel's IP routing tables. Its primary use is to set up static routes to specific hosts or networks via an interface after it has been configured with the ifconfig program. When "add" or "del" options are used, route modifies the routing tables. Without these options, route displays the current contents of the routing tables. The following expression adds the network address 158.196.81.204 to be gatewayed, which means route packets via a gateway, through the IP 10.0.8.1 (tunnel IP address of Ostrava machine).

```
~# route add –net 158.196.81.204 netmask 255.255.255. 255 gw 10.0.8.1
```

IxChariot Console machine

On the IxChariot Console machine at Command Prompt, cmd.exe, was performed the following instruction in order to route packets with network IP 10.8.0.0/24 destiny through the IP address 158.196.81.208.

```
C:\> route add 10.8.0.0 MASK 255.255.255.0 158.196.81.208
```

6.4.4. Openswan IPsec plus OpenVPN TLS Secure Connection Testbed

In order to get an IPsec plus TLS secure network environment, were used simultaneously Openswan and OpenVPN security mechanisms with the described above configurations.

The used ciphers by Openswan IPsec and OpenVPN TLS were the AES (128 bits block size) with 128 bits key sizes and Triple-Des (64 bits block size) with 192 bits key sizes.

6.5. IxChariot Test Tool

IxChariot, an IXIA product, was the used VoIP test tool. The test is initialized at the console, the conditions are uploaded into the Endpoints and consequently the test is performed. The results are sent back to the console at the end.

To create a Voice over IP test, was initialized the IxChariot Console application and clicked "Add VoIP Pair" on the "Edit" menu, as can be seen in *Figure 34*. It was chosen from the following configuration parameters. Clicking on "Advanced" can be edited the application script to be used in the test.

For a VoIP testing, by default, Endpoint 1 (158.196.81.208) executes a test script, receives results from Endpoint 2 (147.32.201.116) and reports them to the Console over the same network segment.

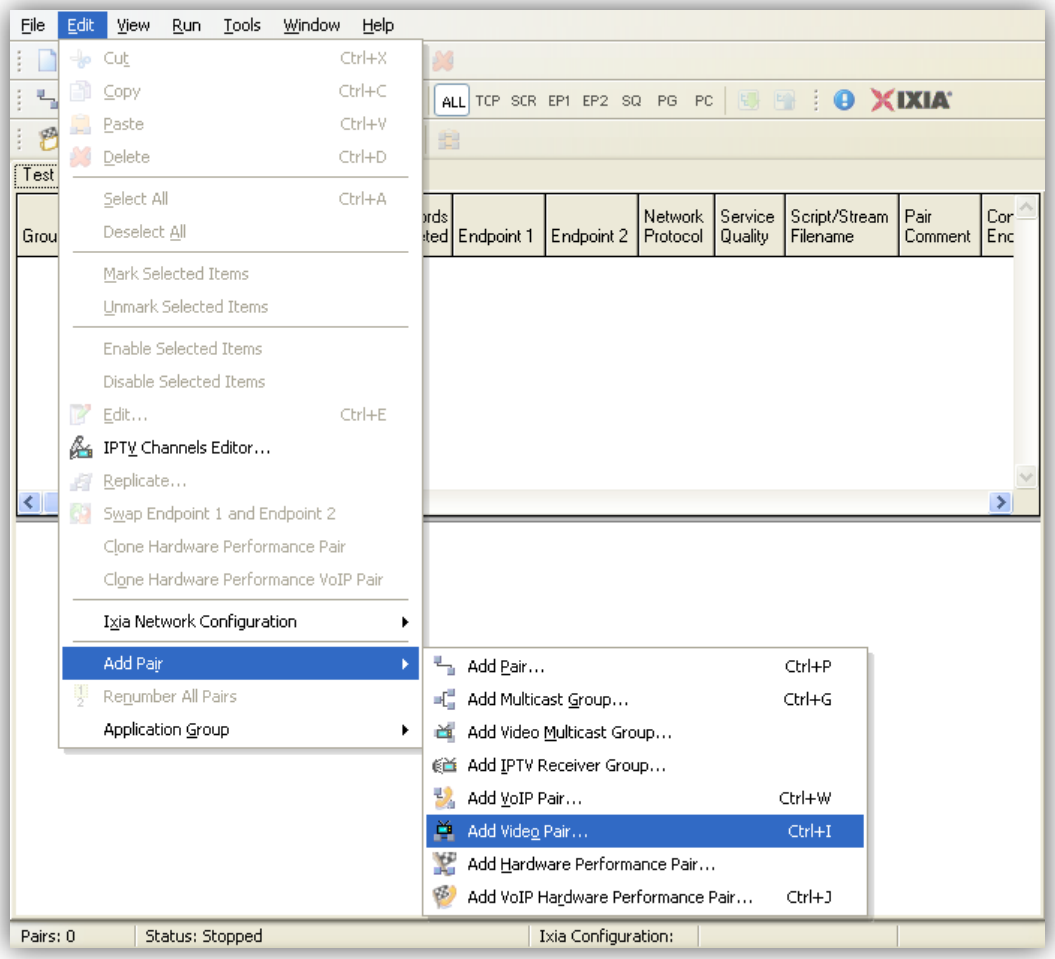

Figure 34: IxChariot Console – Add VoIP Pair example.

Endpoint 1 to Endpoint 2 Traffic (IxChariot VoIP Pair Configuration)

The following parameters are the configurable options on an IxChariot VoIP pair between two Endpoints.

A brief explanation about the different options is presented below.

Pair Comment (optional)

A descriptive word or phrase that let the user easily identify each pair in the Test window.

Endpoint 1 address

The network address of a computer playing the role of Endpoint 1 was 158.196.81.208. The IxChariot Console contacts Endpoint 1 computers directly, sending them the test setup information that is entered. Endpoint 1 acts as the sender of voice data. Should be entered a network address that matches the used protocol.

Endpoint 2 address

The computer playing the role of Endpoint 2 had the IP address 147.32.201.116. Endpoint 1 computer contacts the Endpoint 2 computer with test setup information. Endpoint 2 acts something like the receiver of voice data in a VoIP call.

Codec

The type of codec used on the network. Can be selected from the list one of the supported codecs. For the performed tests were chosen the G.711 A-law (64 Kbps) and G.729 (8 Kbps) codecs.

Packet Loss Concealment

For G.711 μ-law and G.711 A-law codecs only. PLC should improve the MOS estimate, but it is only factored into the calculation if the results include some data loss. Most G.711 codecs today do not support PLC. Were performed VoIP tests for G.711 A-law codec with and without PLC in order to analyze the R-factor estimations, when occurred some data loss.

Use silence suppression

Emulates the effects of silence suppression (also called voice activity detection) on the line during the VoIP test. This option was not chosen.

Voice activity rate

An indicator of the percentage of time during the call that talking is occurring. Determines how much actual voice data the simulated call contains for the tests. Can be set any rate from 1% to 100% to indicate the amount of voice data transmitted during a call. Default value is 50%. Only can be enabled this option if "Use silence suppression" is checked. As "Use silence suppression" was not used, consequently this option was not chosen.

Override delay between voice datagrams

Determines the datagram size to be used in the VoIP test. VoIP applications break voice data into datagrams based on delay or the amount of time between successive datagrams. Default value is 20 ms. Values must be between 10 and 200 ms. IxChariot may adjust values slightly after entered them so that no partial buffers are sent. This option was chosen and the fixed value for override delay between voice datagrams was 20 milliseconds.

Timing record duration

The length of each timing record. Timing records are used to take measurements during an IxChariot test. The length of a timing record determines the number of datagrams whose measurements are included in each timing record the test generates. The default value is 3 seconds or about 150 VoIP datagrams if the datagram size is 20 ms. The time that is set is used to calculate the number of datagrams that will be received in that time period if there is no lost data or delay. The endpoint uses the number of datagrams to be received when determining when a timing record ends. It is therefore possible to have timing records longer than the duration specified.

Number of timing records

It represent how many times a record is performed. For this option was used the value 10, which means that every test had a duration of 30 seconds, since a test duration time is given by "Timing record duration" x "Number of timing records".

Service quality

Emulates the effects of a Quality of Service scheme on call quality. IxChariot's pre-configured VoIP template and any QoS templates already created are available in the list. The template named VoIPQoS is recommended for VoIP testing. However, it is not supported by all endpoint operating systems. This option was not used.

Specify a jitter buffer

Emulates the effects of jitter buffering on the VoIP network. Jitter buffers may be configured based on time (called an "absolute" jitter buffer) or based on a number of datagrams (a "frame-based" jitter buffer). An "absolute" jitter buffer was used with set time of 60 milliseconds.

Figure 35 presents the basic IxChariot Pair used configuration between Ostrava – Prague Endpoints for a G.711 A-law with PLC codec.

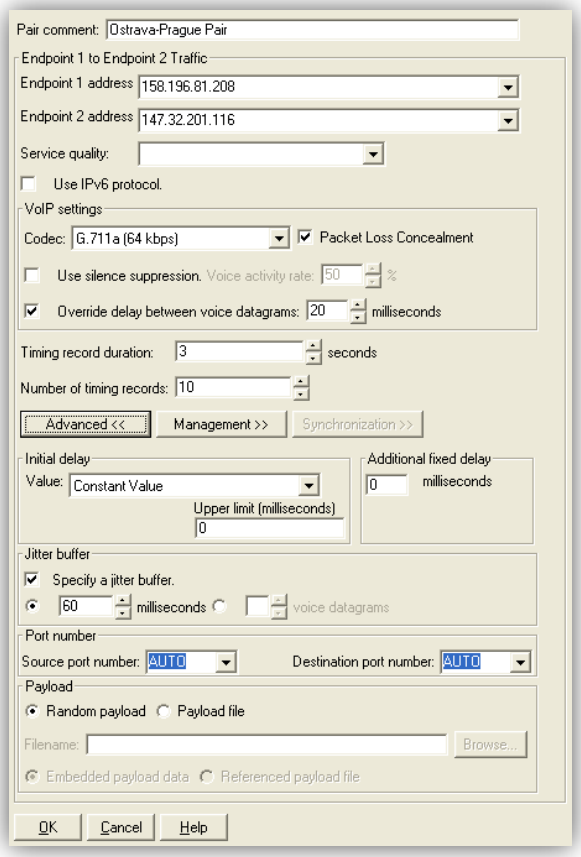

Figure 35: G.711 A-law with PLC Pair configuration example for Unsecure network environment.

The Voice over IP IxChariot test for an OpenVPN TLS secure connection is a bit distinct from the unsecure and Openswan IPsec secure ones. Rather than use for Endpoints addresses the 158.196.81.208 and 147.32.201.116 ones, must be used the IP addresses of the created tunnel, 10.8.0.0/24. Due that fact, the used Endpoints 1 and 2 addresses were 10.8.0.1 and 10.8.0.2, respectively.

A basic IxChariot Pair used configuration between Ostrava – Prague Endpoints for a G.711 A-law with PLC codec, in an OpenVPN TLS secure environment, is shown in *Figure 36*.

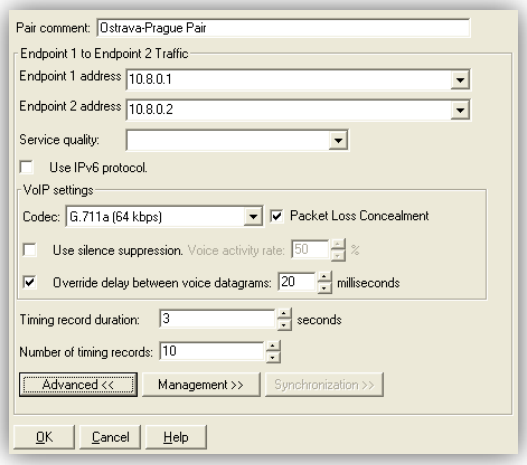

Figure 36: G.711 A-law with PLC Pair configuration example for TLS secure connection.

The Voice over IP tests, between Ostrava – Prague Performance Endpoints, were performed for several numbers of VoIP Pairs. The used version allowed VoIP tests until a maximum of 50 pairs. In order to replicate a configured pair, by choosing the "Replicate..." option when clicked an existent configured pair, it is asked for the number of replications the user desire. This process is shown in *Figure 37*.

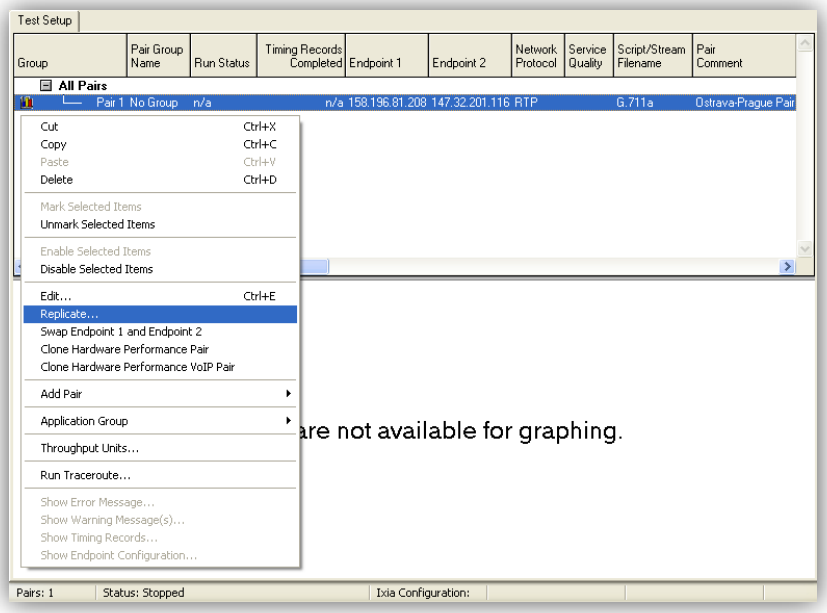

Figure 37: Basic example of VoIP Pairs replication.

Some specific numbers of VoIP pairs were chosen to perform the tests taking into account the bandwidth requirements of the used codecs for the different test environments. The bandwidth requirements calculation process for the distinct used environments can be found into chapter **7**.

For each codec were performed IxChariot VoIP tests for one pair, i. e, one VoIP call.

For G.711 A-law codec the maximum used VoIP pairs was defined taking into account the maximum possible "toll quality" VoIP pairs for an unsecure connection. The maximum "toll quality" VoIP pairs for an unsecure connection achieve results were the references taken to compare with the achieved results for secure environments. For the G.729 codec was used the maximum possible of VoIP pairs for the used IxChariot version, due the fact that was not possible to reach the theoretical maximum possible "toll quality" VoIP pairs limit. Should be mentioned that for each codec several VoIP tests were also performed for closer but lower numbers of VoIP pairs than the defined maximum.

IxChariot Performance Endpoints

The installation process of IxChariot Performance Endpoints, on Ubuntu operating systems, can be found in Appendix **B.1**.

In order to enable IxChariot Performance Endpoints to perform the VoIP tests was executed on the Linux Terminal, at both Ostrava and Prague machines, the following instructions:

```
~# cd /usr/local/Ixia/
/usr/local/Ixia# ./endpoint
Processing INI file (/usr/local/Ixia/endpoint.ini).

ENDPOINT
Performance Endpoint, Version 6.50 (Retail)
Copyright (c) Ixia, 2003-2008.
Copyright (c) NetIQ Corporation, 1995-2003.
U.S. Patents 5,838,919, 5,881,237, 5,937,165, 6,061,725, 6,397,359,
and 6,408,335.
Other Patents Pending.
Build 395

Endpoint INI information in use:
  All available protocols are enabled.
  All consoles may run tests on this endpoint.
  Security Auditing: NONE
  Audit filename: /usr/local/Ixia/endpoint.aud

Support for IP Multicast is enabled.
Support for IPv6 Multicast is enabled.
Support for disabling and enabling TCP's Nagle Algorithm is available.
Support for disabling and enabling generation of the UDP checksum is available.
Support for setting the Maximum Segment Size is available.
Support for TCP protocol has been started.
Support for IP Multicast is enabled.
Support for IPv6 Multicast is enabled.
Support for disabling and enabling TCP's Nagle Algorithm is available.
Support for disabling and enabling generation of the UDP checksum is available.
Support for setting the Maximum Segment Size is available.
Support for TCP-IPv6 protocol has been started.
```

VoIP Test Execution (Batch File)

After set all IxChariot Console configurations and enabled IxChariot Performance Endpoints, the tests are ready to be performed. In order to execute five times each IxChariot VoIP test to have more rigorous results, a batch file was created to facilitate this task.

A batch file is a text file containing a series of commands intended to be executed by the command interpreter.

The created batch file, "Test.bat", comprise the running of each test five times, as well as an instruction to output the test results on html format.

The content of the created "Test.bat" file is presented below.

```
"C:\Program Files\Ixia\Ixchariot\runtst.exe" Test.tst
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_1.tst.txt -c
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_1.tst.html -h
sleep 40

"C:\Program Files\Ixia\Ixchariot\runtst.exe" Test.tst
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_2.tst.txt -c
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_2.tst.html -h
sleep 40

"C:\Program Files\Ixia\Ixchariot\runtst.exe" Test.tst
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_3.tst.txt -c
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_3.tst.html -h
sleep 40

"C:\Program Files\Ixia\Ixchariot\runtst.exe" Test.tst
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_4.tst.txt -c
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_4.tst.html -h
sleep 40

"C:\Program Files\Ixia\Ixchariot\runtst.exe" Test.tst
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_5.tst.txt -c
"C:\Program Files\Ixia\Ixchariot\fmttst.exe" Test.tst Test_5.tst.html -h
sleep 40
```

7 BANDWIDTH REQUIREMENTS

7.1. Introduction

Throughout this chapter is presented and explained the equations of required bandwidth on several unsecure and secure network environments, take into account factors as the used codec, timing, payload size, number of concurrent calls and so on.

A concise comparison about bandwidth requirements between unsecure and distinct secure connections is presented.

7.2. Unsecure Connection Bandwidth Requirements

The basic steps of speech processing on the transmission side are encoding and packetizing [82; 83]. RTP packets are sent in dedicated time and the difference between two consecutive packets depends of the timing variable.

The time related to the process of packetizing is given by the following basic equation:

$$\Delta t = \frac{P_S}{C_R} \quad (1)$$

where $\Delta t_{[s]}$ is the timing, $P_{S\,[b]}$ the payload size and $C_{R\,[bps]}$ represents the codec rate. The values for the $C_{R\,[bps]}$ of the used codecs are the following:

- G.711 A-law, $C_R = 64\ \text{Kbps}$;
- G.729, $C_R = 8\ \text{Kbps}$.

The timing can be derived from the content of RTP packets as the difference of two consecutive timestamps, as it is presented on equation (**2**). The sampling frequency typical value is 8 KHz.

$$\Delta t = \frac{\text{timestamp}_{(N+1)} - \text{timestamp}_{(N)}}{\text{sampling\_frequency}} \quad (2)$$

The set timing for an IxChariot VoIP pair configuration was 20 ms, value that later was confirmed by analyzing individual packets captured by Wireshark.

There is the need to express the size of a packet at Application layer, which should be defined by the following formula:

$$S_{AL} = H_{RTP} + P_S \quad (3)$$

where $S_{AL\,[b]}$ is the expected size of the packet at Application layer, which consists in the sum of RTP header (12 Bytes), $H_{RTP\,[b]}$, and payload size, $P_{S\,[b]}$. It should be noted that for the case of using SRTP, the value of SRTP header, $H_{SRTP\,[b]}$, is 22 Bytes.

The presented equation (**4**) determines the size of the frame, $S_{F[b]}$, at Link layer.

$$S_F = S_{AL} + \sum_{j=1}^{3} H_j \quad (4)$$

$S_{F\,[b]}$ includes the packet at Application layer and the sum of lower located headers of OSI model, where $H_{1\,[b]}$ is the Data Link layer header (26 Bytes), $H_{2\,[b]}$ the Network layer header (20 Bytes) and $H_{3\,[b]}$ the Transport layer header (8 Bytes).

The total bandwidth, $\mathbf{BW_{M\ [kbps]}}$, as a function of payload size, $\mathbf{P_{S\ [b]}}$, and number of concurrent calls, **M**, is determined by the following equation:

$$BW_M = \sum_{i=1}^{M} \frac{S_{F_i}}{\Delta t_i} \quad (5)$$

The total bandwidth, $\mathbf{BW_{M\ [kbps]}}$, presented in the equation (**5**), which is required in case of **M** concurrent calls, when related with the presented formulas (**1**), (**2**) and (**4**) can be obtained the following result:

$$BW_M = M \cdot C_R \cdot \left(1 + \frac{H_{RTP} + \sum_{j=1}^{3} H_j}{P_S}\right) \quad (6)$$

By linking the achieve equation (**6**), the number of possible concurrent calls, **M**, as function of bandwidth, $\mathbf{BW_{M\ [kbps]}}$, and payload size, $\mathbf{P_{S\ [b]}}$, is given by the formula:

$$M = \frac{BW_M}{C_R \cdot \left(1 + \frac{H_{RTP} + \sum_{j=1}^{3} H_j}{P_S}\right)} \quad (7)$$

Figure 38 presents bandwidth as function of timing and number of concurrent calls for the used codecs, G.711 A-law and G.729, in an unsecure connection.

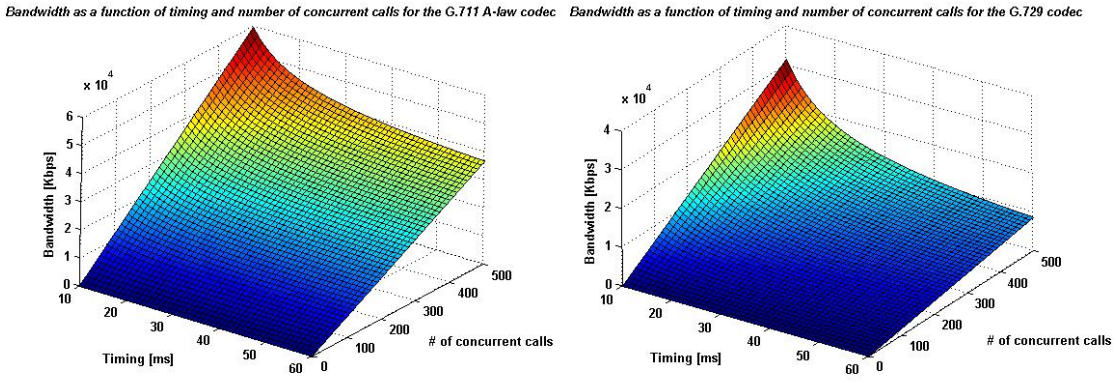

Figure 38: Bandwidth as function of timing and number of concurrent calls.

7.3. IPsec Secure Connection Bandwidth Requirements

For the calculation of bandwidth to an IPsec scenario, on equation $\mathbf{BW_{M\ [kbps]}}$ it is necessary to resize the payload size, $\mathbf{P_S}$, and add the total contribution of the various headers of the distinct layers on $\mathbf{S_{AL}}$ expression. Empirically it was found that the calculation can be considerably simplified and still provide relevant results. The sample call over IPsec was tested three hundred times at Vysoká Škola Báňská - Technická Univerzita Ostrava Katedry Telekomunikační and by analyzing individual captured packets can be claimed that IPsec tunnel mode introduces constants for the typical size of payload the values presented in Table **9**.

Table 9: IPsec secure connection Network Layer, H_2, bandwidth requirements.

	H_2 [b]	
Codec	**w/AES Cipher**	**w/Triple-Des Cipher**
G.711 A-law	704	640
G.729	672	608

In the achieved $\mathbf{BW_{M\,[kbps]}}$ equation (**6**), for an IPsec secure connection, the variable $\mathbf{H_{2\,[b]}}$ should be replaced with the respective value that is presented in the table above, taking into account the used cipher.

7.4. TLS Secure Connection Bandwidth Requirements

Firstly is necessary to remind that TLS is located between two layers of the OSI model, which are Application and Transport layers, how was described into subchapter **5.4**. Therefore is applied $\mathbf{S_{TLS\,[b]}}$ instead of $\mathbf{S_{AL\,[b]}}$. This replacement should be done in respect of the explained location of TLS, so is defined a new parameter, $\mathbf{S_{TLS\,[b]}}$, the expected size of the packet at TLS layer. $\mathbf{S_{TLS\,[b]}}$ is expressed by the equation (**8**).

$$S_{TLS} = C_0 + \left\lceil \frac{S_{AL}}{B_S} \right\rceil \cdot B_S \qquad (8)$$

Is used the symbol $\lceil x \rceil$ to denote the ceiling function where $\lceil x \rceil = \min\{n \in Z \mid x \leq n\}$, what means that ceiling function of 'x' gives the smallest integer greater than or equal to 'x'. The ceiling function was defined by M. Schroeder in 1991 [84] and the symbol was coined by K. Iverson in 1994. The parameter $\mathbf{B_{S\,[b]}}$ represents the cipher block size, that in case of this dissertation can adopt the values 64 or 128 bits, fact that depends on applied cipher algorithm (AES or Triple-DES). $\mathbf{C_{0\,[b]}}$ is a constant that for case of clear TLS takes the zero value. Unfortunately OpenVPN adds supplementary overhead, fact that should be included in $\mathbf{C_{0\,[b]}}$.

The OpenVPN supplementary overhead value was achieved by numerous performed experiments by Miroslav Vozňák. M. Vozňák claims the constant $\mathbf{C_0}$=83 Bytes in case of block size equals to 128 bits and $\mathbf{C_0}$=75 Bytes in case of 64 bits block size [3].

7.5. Theoretical Calculations for Bandwidth Requirements

Table **10** presents the values of required bandwidth of one call for unsecure and several secure environments, as Openswan IPsec, OpenVPN TLS and using SRTP, by applying the equations presented along this chapter. The TLS case calculations took into account the OpenVPN TLS without LZO compression/decompression mode.

Table 10: Values of required bandwidth of one call for distinct secure environments.

Codec	**Timing [ms]**	**No Encryption [Kbps]**	**w/IPsec AES [Kbps]**	**w/IPsec Triple-DES [Kbps]**	**w/TLS AES [Kbps]**	**w/TLS Triple-DES [Kbps]**	**w/SRTP [Kbps]**
G.711 A-law	20	90.4	117.6	114.4	125.2	122.0	94.4
G.729	20	34.4	60.0	56.8	67.6	64.4	38.4

In Table **11** is exhibited the maximum theoretical number of successful calls for a connection of 2 Mbps bandwidth.

Table 11: Number of maximum successful calls for a 2 Mbps connection bandwidth.

Codec	Timing [ms]	Connection Bandwidth [Kbps]	No Encryption	w/IPsec AES	w/IPsec Triple-DES	w/TLS AES	w/TLS Triple-DES	w/SRTP
G.711 A-law	20	2000	22	17	17	15	16	21
G.729	20	2000	58	33	35	29	31	52

The presented equations and values throughout this chapter to an OpenVPN secure link are valid only to independent calls between multiple clients to a server, which means that each call is arriving to the same server from a different client, because OpenVPN multiplexes the data streams between client and server over a single IP port. This process is briefly explained below.

OpenVPN multiplexes all communications over a single IP port [85]. The problem is that OpenVPN uses its own protocol that is not truly TLS, although it makes use of TLS to secure the control channel. The data that OpenVPN sends out is not RFC-2246 [72] compliant because OpenVPN needs to multiplex both the control and data channels over a single network stream [86].

This model has the benefit that SSL/TLS sees a reliable transport layer while the IP packet forwarder sees an unreliable transport layer. The reliability and authentication layers are completely independent of each other.

The server uses the client's certificate to cipher the stream that is sent to the client and it uses OpenSSL functionality to multiplex the tunnel into an UDP or TCP IPv4 stream.

OpenVPN multiplexes two channels onto the VPN: a data channel that carries the users' IP datagrams and a control channel that handles such protocol chores as key negotiation and configuration [87]. Terminating, OpenVPN is designed to multiplex SSL/TLS plus actual IP datagrams over an UDP/TCP port.

Figure 39 presents a basic scheme of how OpenVPN multiplexing is processed.

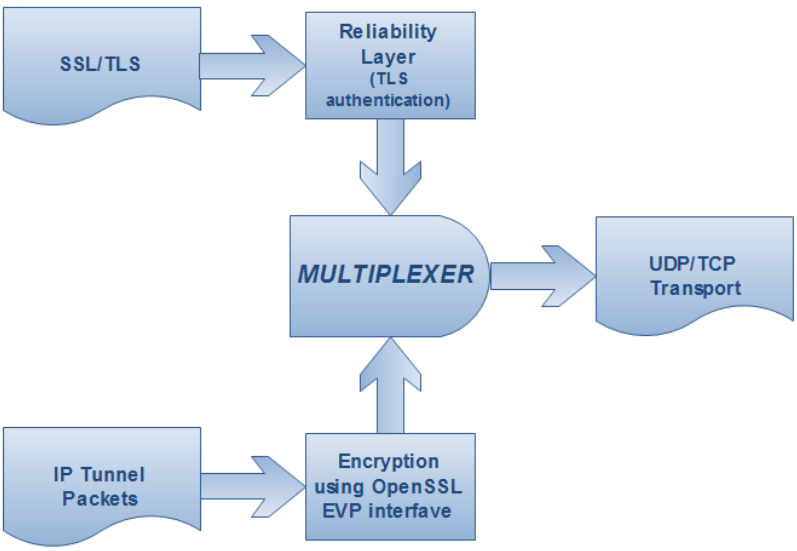

Figure 39: OpenVPN multiplexing technique.

OpenVPN multiplexes the SSL/TLS session used for authentication and key exchange with the actual encrypted tunnel data stream.

In *Figures 40* and *41* are presented the theoretical bandwidth graphics as function of security mechanism and number of concurrent calls for G.711 A-law and G.729 codecs, respectively.

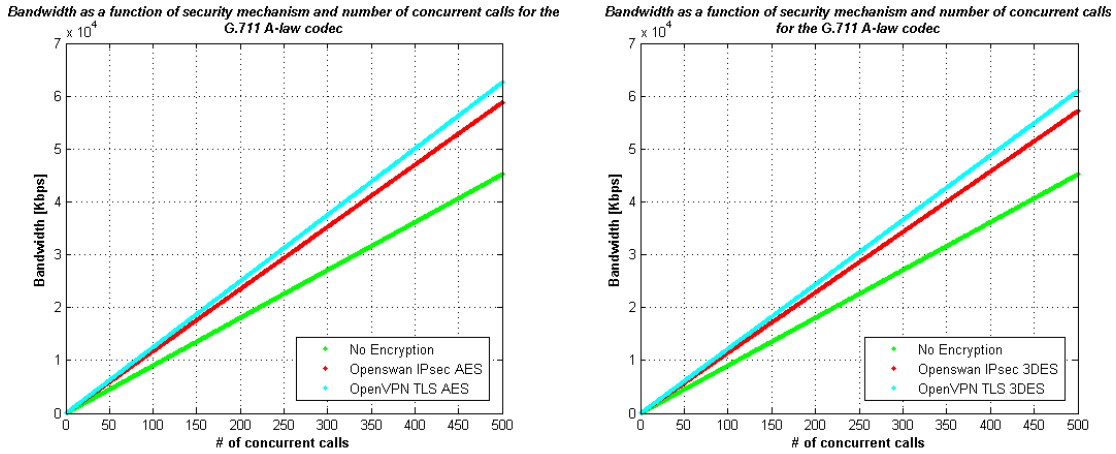

Figure 40: Bandwidth as function of security mechanism and number of concurrent calls for the G.711 A-law codec.

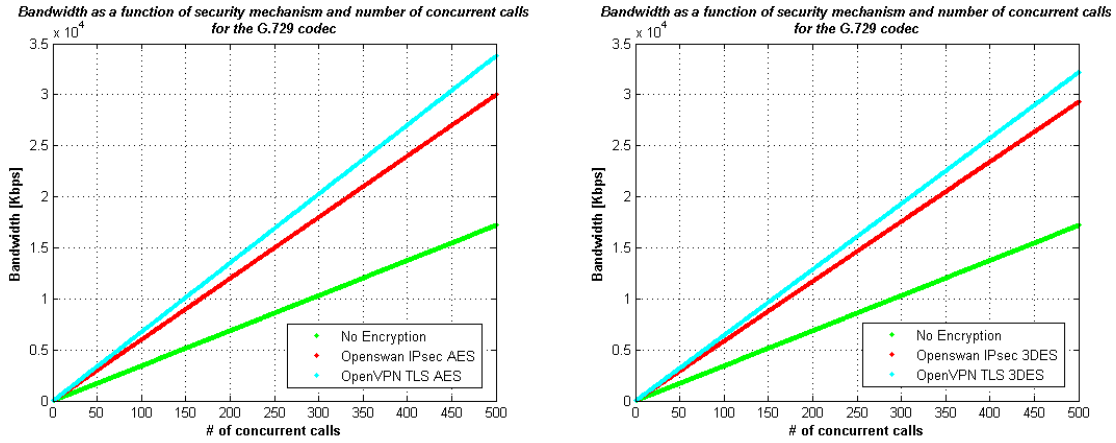

Figure 41: Bandwidth as function of security mechanism and number of concurrent calls for the G.729 codec.

8 ACHIEVED RESULTS

8.1. Introduction

Throughout this chapter is presented the achieved results of the explained procedures throughout chapter **6**.

The obtained results, presented along this chapter, are based on several series of measurements, which have been performed in a real network between Ostrava (158.196.81.208) and Prague (147.32.201.116), Czech Republic.

8.2. Connection Bandwidth

Was performed the measurement of Ostrava – Prague connection bandwidth by using the interoperability of Iperf and JPerf tools. The achieved results for the connection bandwidth between Ostrava and Prague machines, before and after set the Simena Network Emulator (NE1000), can be found below.

Ostrava – Prague Connection Bandwidth

Iperf Server Side Output (Prague Machine)

```
~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 147.32.201.116 port 5001 connected with 158.196.81.208 port 41627
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec    112 MBytes  93.7 Mbits/sec
```

Iperf/JPerf Client Side Output (Ostrava Machine)

```
iperf -c 147.32.201.116 -P 1 -i 1 -p 5001 -f m -t 10
------------------------------------------------------------
Client connecting to 147.32.201.116, TCP port 5001
TCP window size: 0.02 MByte (default)
------------------------------------------------------------
[  3] local 158.196.81.208 port 41627 connected with 147.32.201.116 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  11.1 MBytes  92.7 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  1.0- 2.0 sec  11.2 MBytes  93.7 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  2.0- 3.0 sec  11.3 MBytes  95.2 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  3.0- 4.0 sec  11.2 MBytes  94.0 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  4.0- 5.0 sec  11.3 MBytes  94.8 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  5.0- 6.0 sec  11.2 MBytes  94.4 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  6.0- 7.0 sec  11.4 MBytes  95.6 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  7.0- 8.0 sec  11.1 MBytes  93.1 Mbits/sec
```

```
[ ID] Interval       Transfer     Bandwidth
[  3]  8.0- 9.0 sec  11.2 MBytes  93.8 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  9.0-10.0 sec  11.2 MBytes  93.7 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   112 MBytes  94.1 Mbits/sec
Done.
```

JPerf Connection Bandwidth Graphic Output (Ostrava Machine)

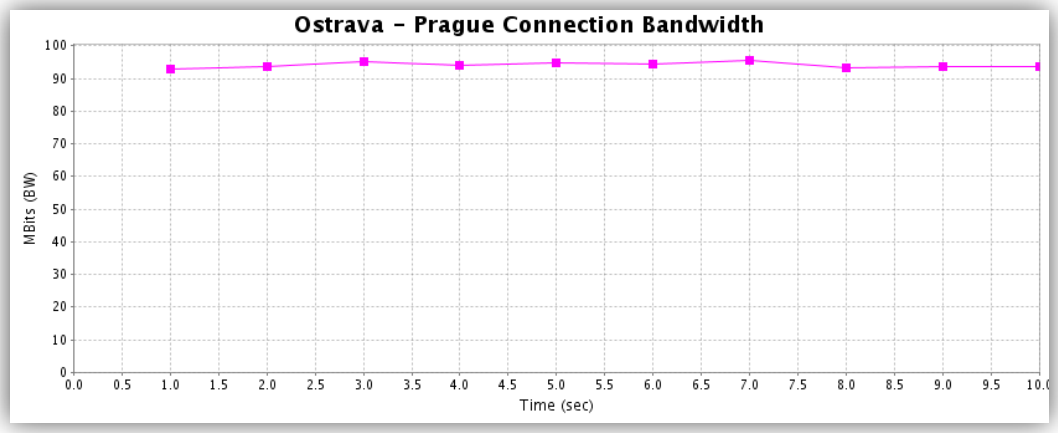

Figure 42: JPerf graphic output for Ostrava – Prague connection bandwidth.

Ostrava – Prague Connection Bandwidth with Simena NE1000

Iperf Server Side Output (Prague Machine)

```
~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 147.32.201.116 port 5001 connected with 158.196.81.208 port 1653
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.1 sec  2.30 MBytes  1.91 Mbits/sec
```

Iperf/JPerf Client Side Output (Ostrava Machine)

```
iperf -c 147.32.201.116 -P 1 -i 1 -p 5001 -f m -t 10
------------------------------------------------------------
Client connecting to 147.32.201.116, TCP port 5001
TCP window size: 0.01 MByte (default)
------------------------------------------------------------
[  3] local 158.196.81.208 port 1653 connected with 147.32.201.116 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  0.24 MBytes  2.03 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  1.0- 2.0 sec  0.23 MBytes  1.90 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  2.0- 3.0 sec  0.23 MBytes  1.90 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  3.0- 4.0 sec  0.23 MBytes  1.90 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  4.0- 5.0 sec  0.23 MBytes  1.90 Mbits/sec
```

```
[ ID] Interval       Transfer     Bandwidth
[  3]  5.0- 6.0 sec  0.23 MBytes  1.97 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  6.0- 7.0 sec  0.23 MBytes  1.90 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  7.0- 8.0 sec  0.23 MBytes  1.90 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  8.0- 9.0 sec  0.23 MBytes  1.90 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  9.0-10.0 sec  0.23 MBytes  1.90 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  2.30 MBytes  1.91 Mbits/sec
Done.
```

JPerf Connection Bandwidth Graphic Output (Ostrava Machine)

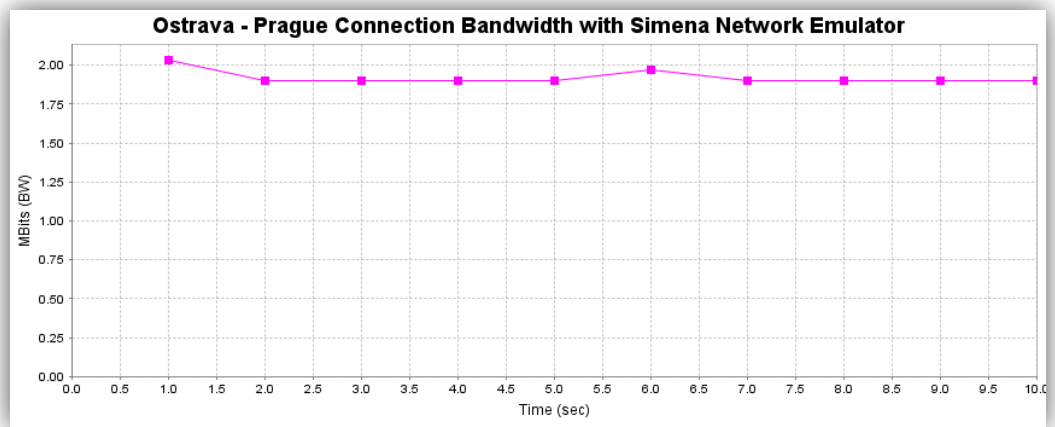

Figure 43: JPerf graphic output for Ostrava – Prague connection bandwidth with Simena NE1000.

8.3. Connection Specifications

It were performed five times, in order to get more concise results, the "ping" and "traceroute" instructions between Ostrava and Prague machines. The achieved "ping" and "traceroute" commands results provide the round-trip-time and the routes taken by packets for the direction Ostrava – Prague and vice-versa.

Ping Output

Ostrava Machine

```
~# ping -c 5 -I 158.196.81.208 147.32.201.116
PING 147.32.201.116 (147.32.201.116) from 158.196.81.208 : 56(84) bytes of data.
64 bytes from 147.32.201.116: icmp_seq=1 ttl=58 time=8.34 ms
64 bytes from 147.32.201.116: icmp_seq=2 ttl=58 time=8.60 ms
64 bytes from 147.32.201.116: icmp_seq=3 ttl=58 time=7.98 ms
64 bytes from 147.32.201.116: icmp_seq=4 ttl=58 time=8.25 ms
64 bytes from 147.32.201.116: icmp_seq=5 ttl=58 time=8.51 ms

--- 147.32.201.116 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 7.983/8.340/8.605/0.246 ms
```

Prague Machine

```
~# ping -c 5 -I 147.32.201.116 158.196.81.208
PING 158.196.81.208 (158.196.81.208) from 147.32.201.116 : 56(84) bytes of data.
64 bytes from 158.196.81.208: icmp_seq=1 ttl=59 time=8.14 ms
64 bytes from 158.196.81.208: icmp_seq=2 ttl=59 time=7.92 ms
64 bytes from 158.196.81.208: icmp_seq=3 ttl=59 time=8.08 ms
64 bytes from 158.196.81.208: icmp_seq=4 ttl=59 time=8.23 ms
64 bytes from 158.196.81.208: icmp_seq=5 ttl=59 time=8.36 ms

--- 158.196.81.208 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4013ms
rtt min/avg/max/mdev = 7.920/8.149/8.362/0.157 ms
```

Traceroute Output

Ostrava Machine

```
~# traceroute -s 158.196.81.208 147.32.201.116
traceroute to 147.32.201.116 (147.32.201.116), 30 hops max, 40 byte packets
 1  vsbgw-v203.vsb.cz (158.196.81.1)  1.272 ms  1.430 ms  1.636 ms
 2  158.196.99.25 (158.196.99.25)  1.858 ms  2.075 ms  2.289 ms
 3  195.113.113.113 (195.113.113.113)  2.825 ms  3.038 ms  3.260 ms
 4  r92-cvut.cesnet.cz (195.113.144.174)  10.653 ms  11.103 ms  11.328 ms
 5  147.32.252.106 (147.32.252.106)  11.551 ms  11.777 ms  12.002 ms
 6  dexter.feld.cvut.cz (147.32.201.116)  12.390 ms  8.671 ms  8.759 ms
```

Prague Machine

```
~# traceroute -s 147.32.201.116 158.196.81.208
traceroute to 158.196.81.208 (158.196.81.208), 30 hops max, 40 byte packets
 1  147.32.192.1 (147.32.192.1)  0.309 ms  0.331 ms  0.368 ms
 2  147.32.252.105 (147.32.252.105)  0.395 ms  0.428 ms  0.467 ms
 3  cvut-r92.cesnet.cz (195.113.144.173)  0.607 ms  0.685 ms  0.774 ms
 4  r96-r50.cesnet.cz (195.113.179.94)  7.618 ms  7.709 ms  7.785 ms
 5  195.113.113.114 (195.113.113.114)  7.391 ms  7.455 ms  7.498 ms
 6  158.196.99.26 (158.196.99.26)  7.633 ms  7.377 ms  7.405 ms
 7  158.196.81.208 (158.196.81.208)  7.802 ms  8.092 ms  9.087 ms
```

8.4. IxChariot Test Output

Every relevant IxChariot Console output results are presented with their averages in Appendix **C**.

As an example of an output result of an IxChariot VoIP test, the following images display the most important and relevant result fields for this dissertation. For the presented example below was chosen one realized test of an OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection for 22 G.711 A-law with PLC codec Pairs.

Table **12** presents the achieved results of a typical IxChariot Voice over IP test for the codec G.711 A-law with PLC in an OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection for 22 pairs.

Table 12: IxChariot achieved results for the codec G.711a PLC in an OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection for 22 pairs.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
22	20	3.70	73.66	205	0.352

Figure 44 exhibits the throughput for each individual pair, be it VoIP call, along the time of the execution of the test. The presented codec rate throughput only takes into account the amount of data generated and transferred by IxChariot Endpoints.

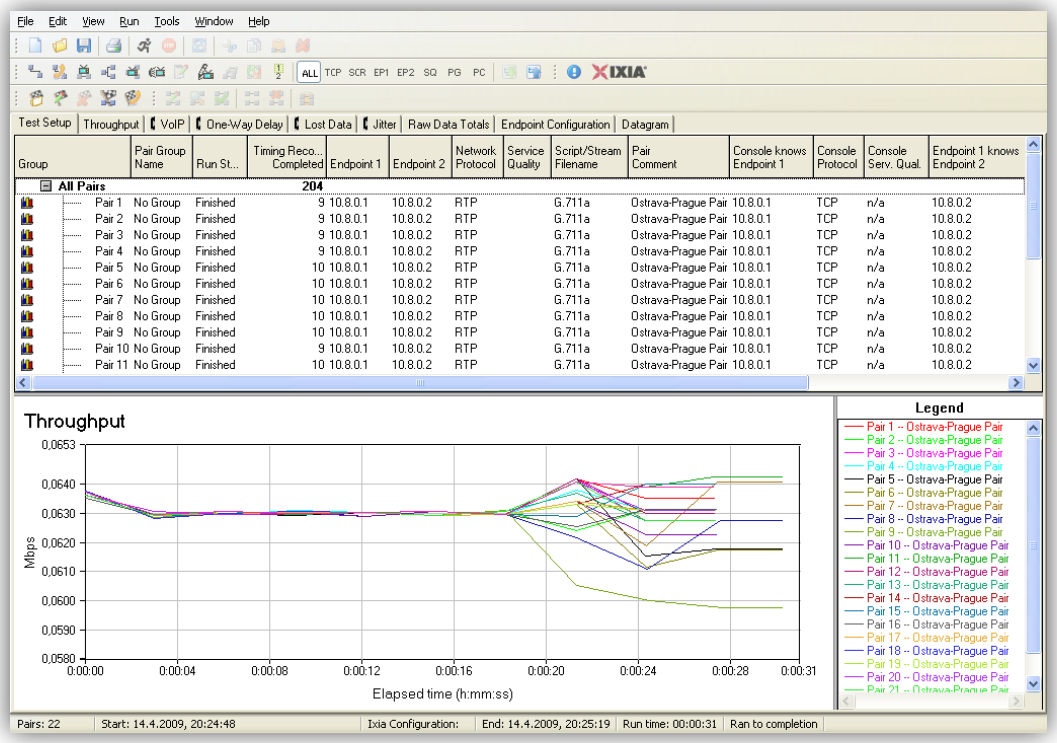

Figure 44: IxChariot Console Throughput example output.

Figure 45 presents a MOS estimation output example for each individual pair along the time of the execution of a test.

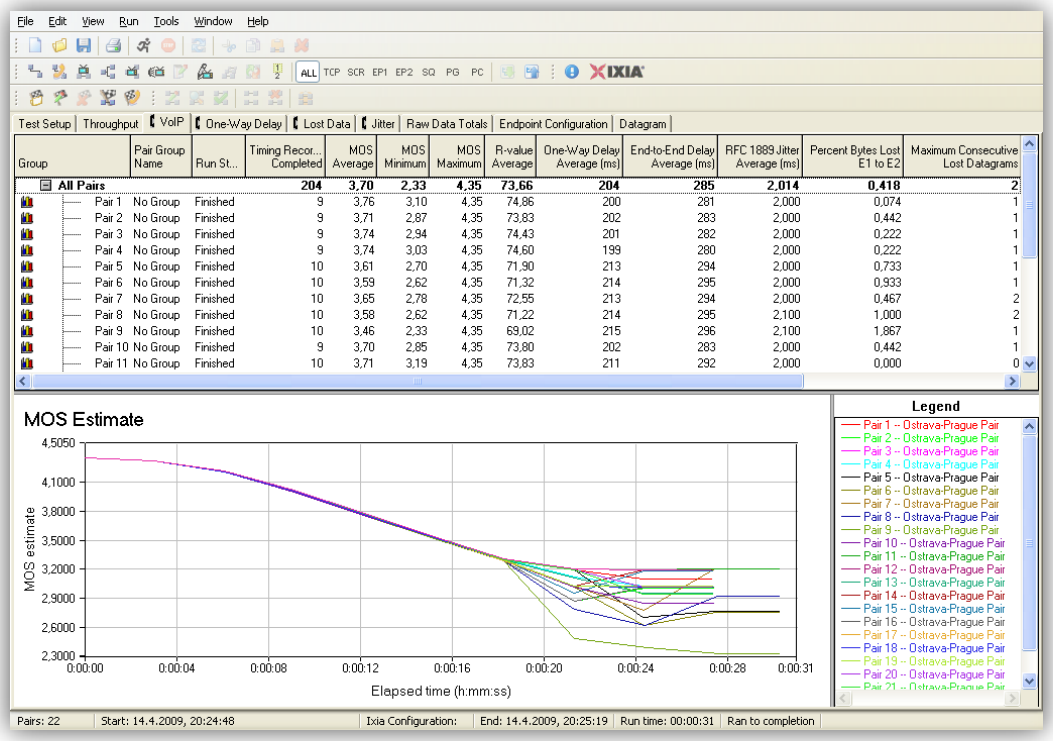

Figure 45: IxChariot Console MOS Estimate example output.

In *Figure 46* is showed the one-way delay for each individual pair along the time of the execution of a test.

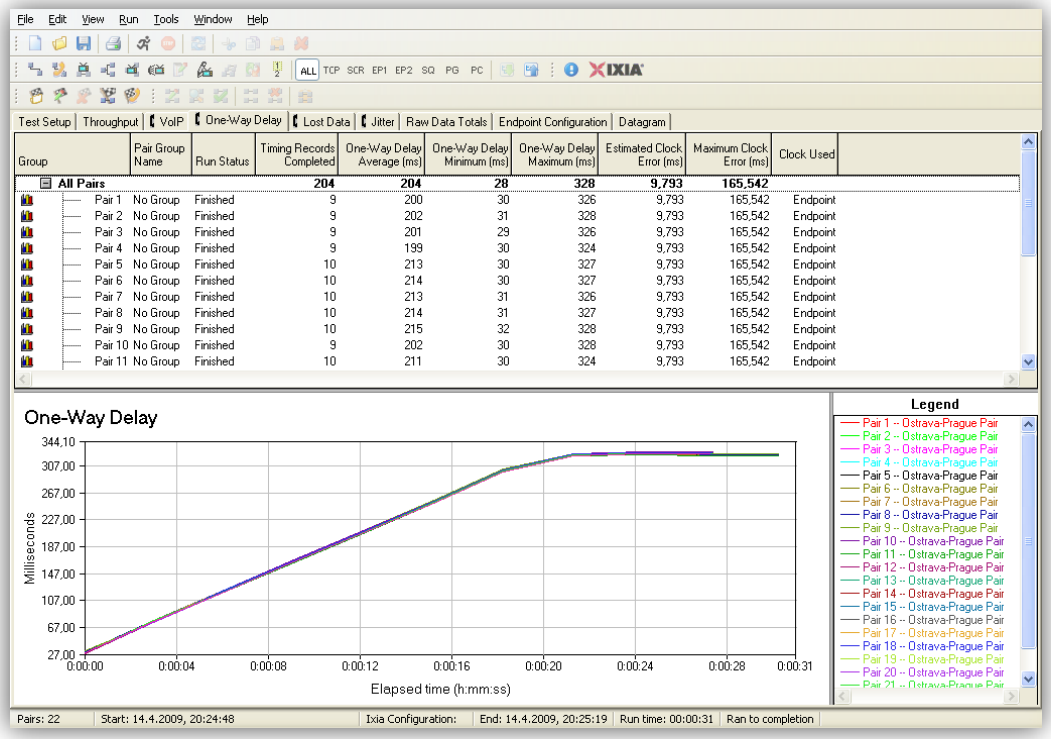

Figure 46: IxChariot Console One-Way Delay example output.

A lost data output example for each individual pair along the time of the execution of a IxChariot VoIP test is presented in *Figure 47*.

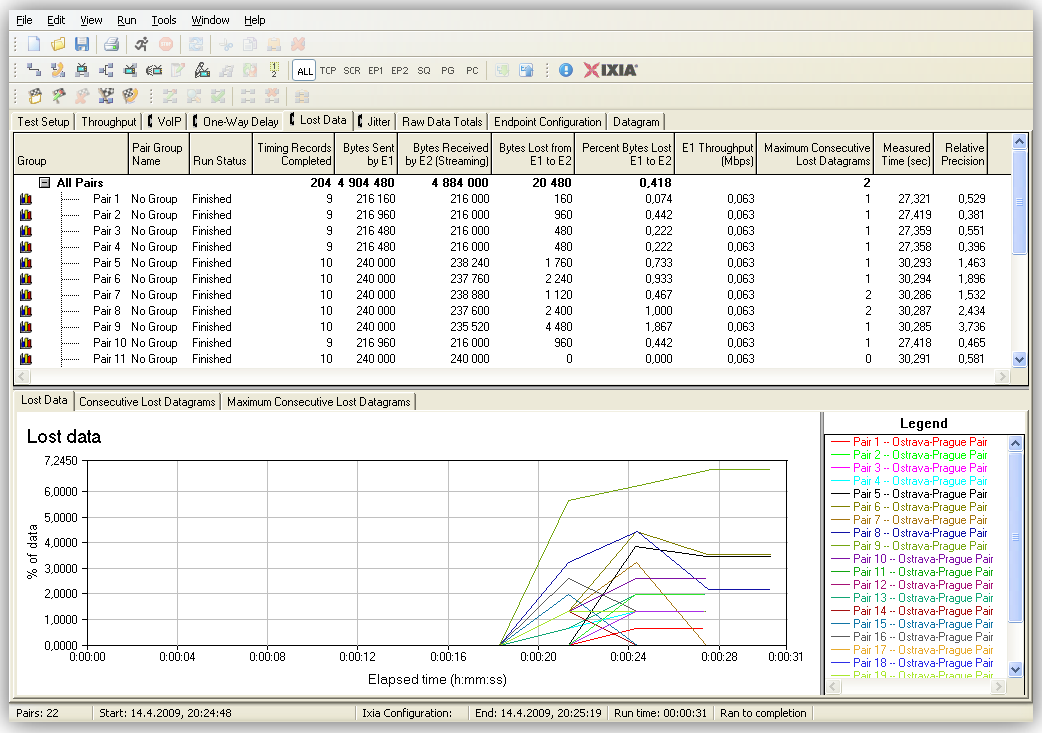

Figure 47: IxChariot Console Loss Data example output.

8.5. ITU-T E-Model R-factor Calculation

From the subsection **ITU-T E-Model** of chapter **4**, it is known the following presuppositions:

$$R = R_0 - I_s - I_d - I_{e,eff} = 93{,}3553 - I_d - I_{e,eff} \quad (1)$$

$$I_{e,eff} = I_e + (95{,}0 - I_e) \cdot \frac{P_{pl}}{\frac{Ppl}{BrustR} + B_{pl}} \quad (2)$$

$$I_d = \begin{cases} 0{,}0267 \cdot d\,, & d < 125\ ms \\ 0{,}1194 \cdot d - 15{,}8760\,, & 175\ ms \leq d \leq 400\ ms \end{cases} \quad (3)$$

By the same way, from the subsection **6.3**, it is known the equation:

$$\text{One Way Symmetric Voice Path Delay} = \frac{RTD}{2} + ESD(A) + ESD(B) \quad (4)$$

By analyzing the achieved results of section **Connection Specifications** of this chapter, it is known that for Ostrava – Prague connection, the repective RTD is 8.340 milliseconds. Therefore, by equation (**3**) the consequent delayed impairment factor, $\mathbf{I_{d\ [ms]}}$, for the used codecs, to a dejjiter buffer of 60 milliseconds, are the following:

- Codec G.711 A-law, ESD(A) = 20 ms $\Rightarrow$ I_d = 86.1700 ms;
- Codec G.729, ESD(A) = 25 ms $\Rightarrow$ I_d = 91.6700 ms.

Resorting to ITU-T Recommendation G.113 [50], it is known the equipment impairment factor, $\mathbf{I_e}$, for the used codecs:

- Codec G.711 A-law, $I_e = 0$;
- Codec G.729, $I_e = 11$.

Assuming for the best possible conditions there is not any loss, by conjugating the equations (**1**), (**2**) and (**3**), are obtained the following R-factor results:

G.711 A-law R-factor Calculation

$$\mathrm{R}=93.3553-\mathrm{I_d}\text{-}\mathrm{I_{e,eff}}=93.3553-86.1700\times 0.0267-0=91.0546 \quad (\mathbf{5})$$

G.729 R-factor Calculation

$$\mathrm{R}=93.3553-\mathrm{I_d}\text{-}\mathrm{I_{e,eff}}=93.3553-91.6700\times 0.0267-11=79.9077 \quad (\mathbf{6})$$

The Mean Option Score, MOS, can be obtained from R-factor as described in ITU-T Recommendation G.107 [48] by the equation (**7**).

$$\mathrm{MOS}=\begin{cases}\mathrm{R}\leq 0:1\\ \mathrm{R}\leq 0\leq 100:1-\frac{7}{1000}\mathrm{R}+\frac{7}{6250}\mathrm{R}^2-\frac{7}{1000000}\mathrm{R}^3\\ \mathrm{R}\geq 100:4.5\end{cases} \quad (\mathbf{7})$$

Thus, the formula (**7**) cannot be inverted to calculate one R-factor from a given MOS value.

Through the use of equation (**7**) with the obtained R-factor values in (**5**) and (**6**) was converted the achieved R-factor scores into MOS ones.

G.711 A-law R-factor into MOS Conversion

$$\mathrm{MOS}=1-\frac{7}{1000}\cdot 91.0546+\frac{7}{6250}\cdot 91.0546^2-\frac{7}{1000000}\cdot 91.0546^3=4.3640 \quad (\mathbf{8})$$

G.729 R-factor into MOS Conversion

$$\mathrm{MOS}=1-\frac{7}{1000}\cdot 79.9077+\frac{7}{6250}\cdot 79.9077^2-\frac{7}{1000000}\cdot 79.9077^3=4.0205 \quad (\mathbf{9})$$

8.6. Results Comparison

In order to compare the achieved results for an unsecure and several secure communication links was chosen as reference the maximum theoretical number for successful VoIP calls to the set bandwidth, 2 Mbps (see section **6.2**).

As explained in chapter **7** and exhibited in Table **11**, the maximum theoretical possible "toll quality" VoIP calls, be it pairs, to a G.711 A-law codec are 22 and to a G.729 are 58. However, the maximum allowed VoIP pairs for the IxChariot Console available version were 50 pairs. Therefore, for G.729 codec was used as reference the maximum possible VoIP pairs, 50.

The following subsections present the graphics of the achieved VoIP tests results to R-factor, percentage lost data and one-way delay variables, for the considered comparison references.

Despite of all the displayed below average results, the whole performed VoIP simulation results are presented into Appendix **C**.

8.6.1. R-factor

The following graphics, presented in *Figures 48*, *49* and *50*, show a brief comparison of the achieved R-factor average scores of different unsecure and secure tested connection links.

G.711 A-law with Packet Loss Concealment (PLC)

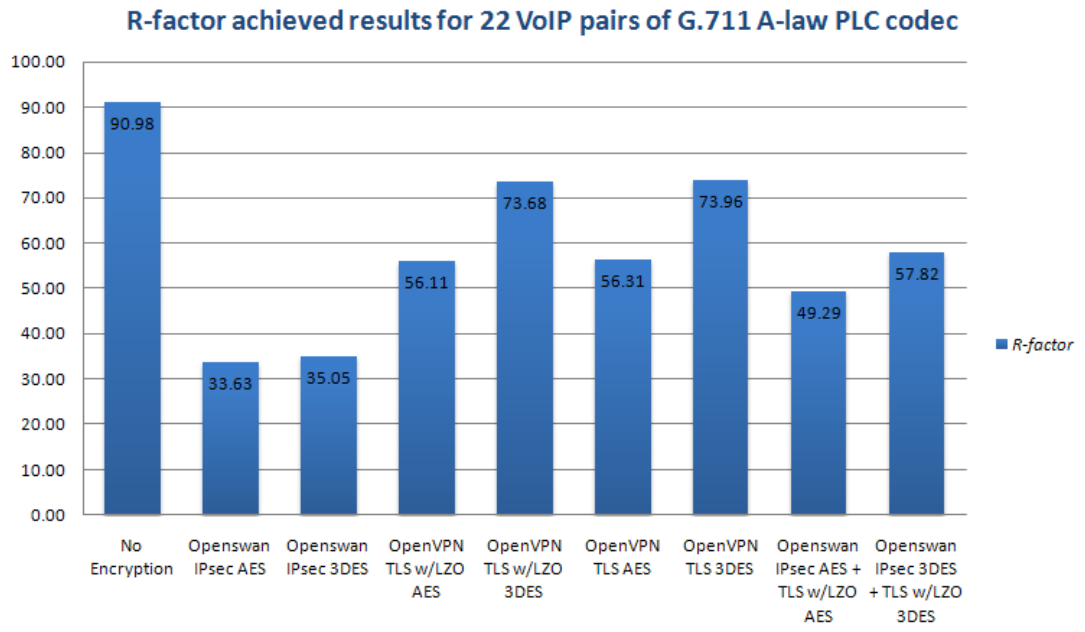

Figure 48: R-factor achieved results for 22 VoIP pairs of G.711 A-law with PLC codec.

G.711 A-law

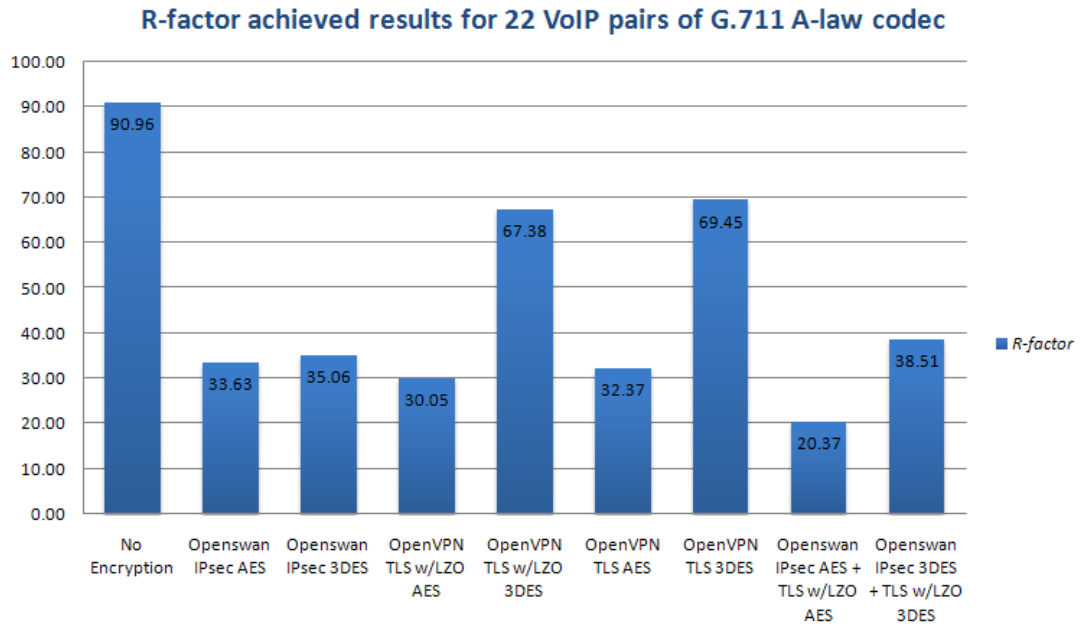

Figure 49: R-factor achieved results for 22 VoIP pairs of G.711 A-law codec.

G.729

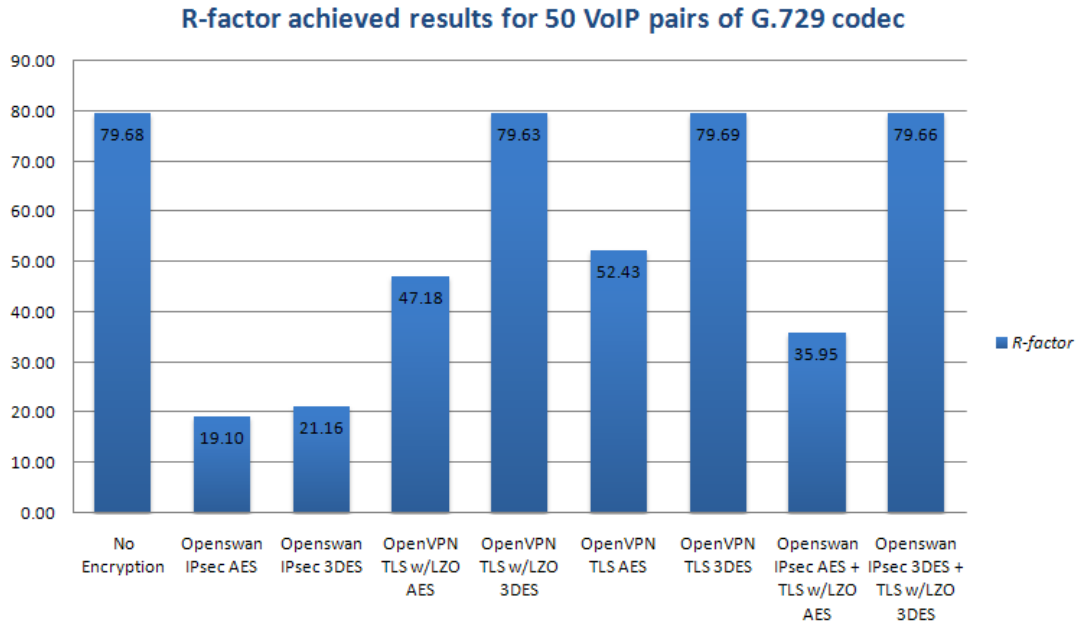

Figure 50: R-factor achieved results for 50 VoIP pairs of G.729 codec.

8.6.2. Percentage Lost Data

The graphics in *Figures 51*, *52* and *53* show a brief comparison of the achieved lost data average scores of different unsecure and secure tested connection links.

G.711 A-law with Packet Loss Concealment (PLC)

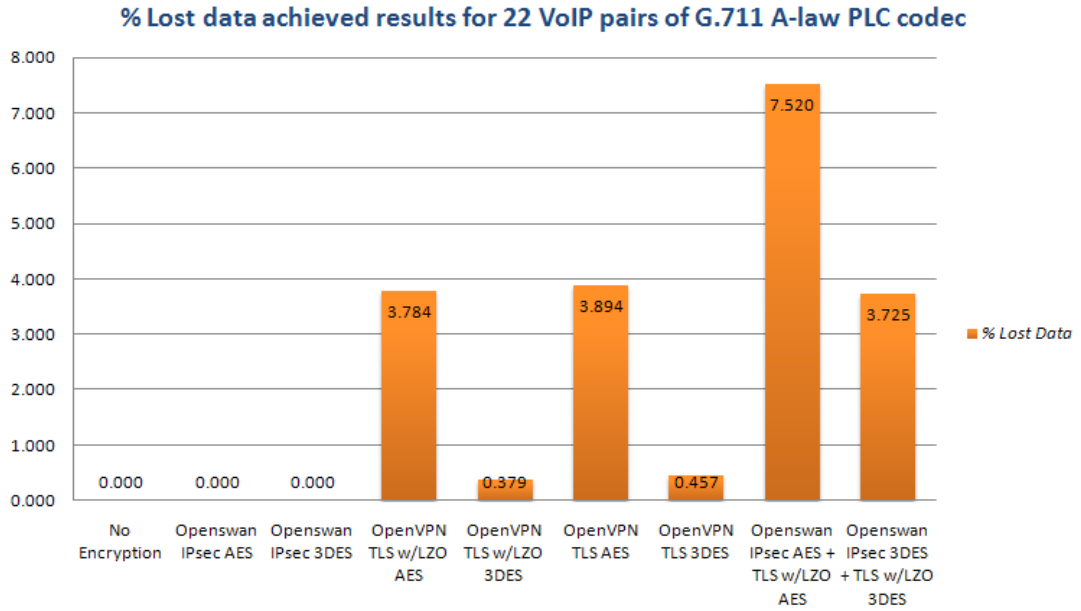

Figure 51: Percentage Lost data achieved results for 22 VoIP pairs of G.711 A-law PLC codec.

G.711 A-law

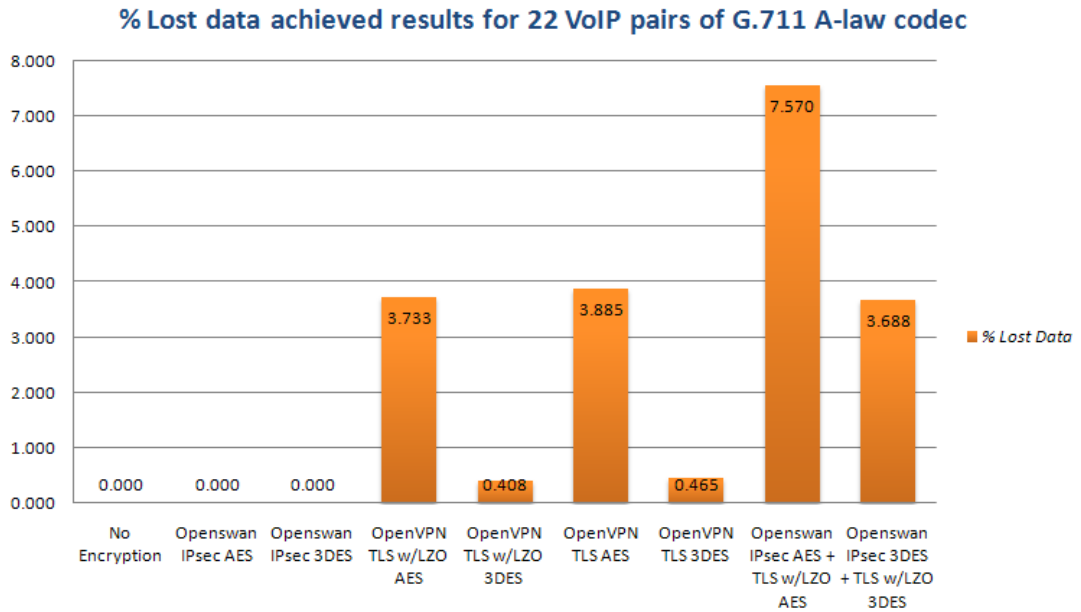

Figure 52: Percentage Lost data achieved results for 22 VoIP pairs of G.711 A-law codec.

G.729

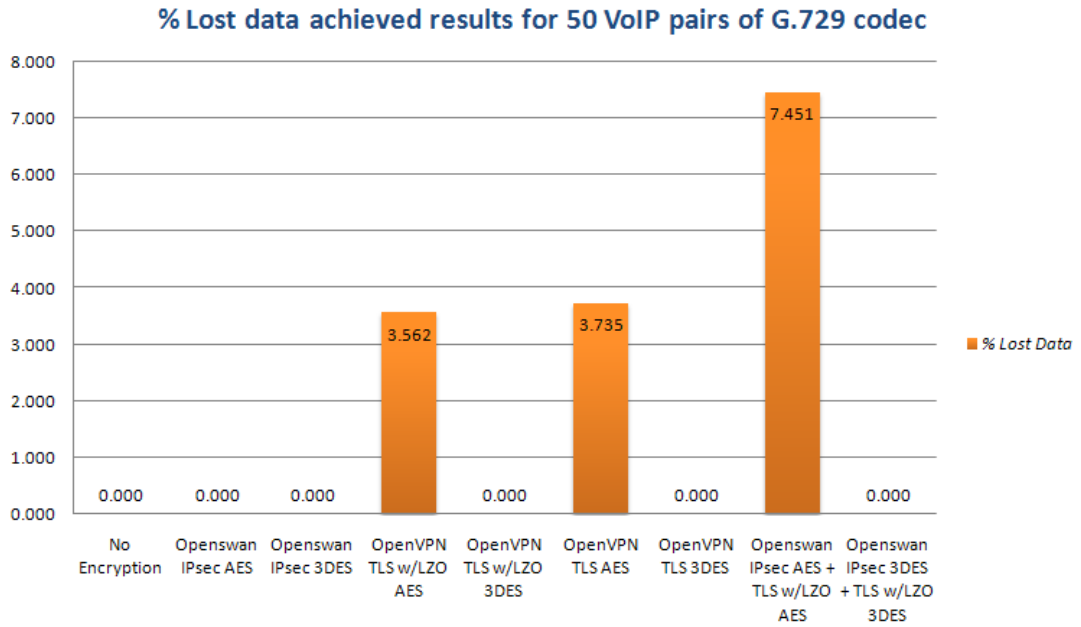

Figure 53: Percentage Lost data achieved results for 50 VoIP pairs of G.729 codec.

8.6.3. One-Way Delay

The following *Figures 54*, *55* and *56* present a brief comparison of the achieved one-way average scores of different tested unsecure and secure connection links.

G.711 A-law with Packet Loss Concealment (PLC)

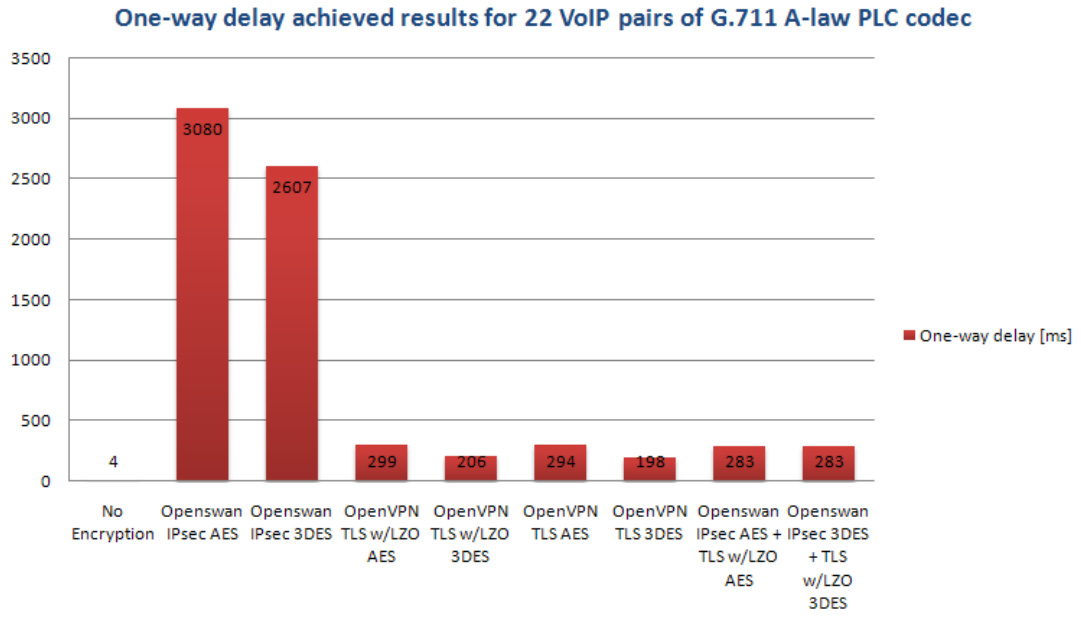

Figure 54: One-way delay achieved results for 22 VoIP pairs of G.711 A-law PLC codec.

G.711 A-law

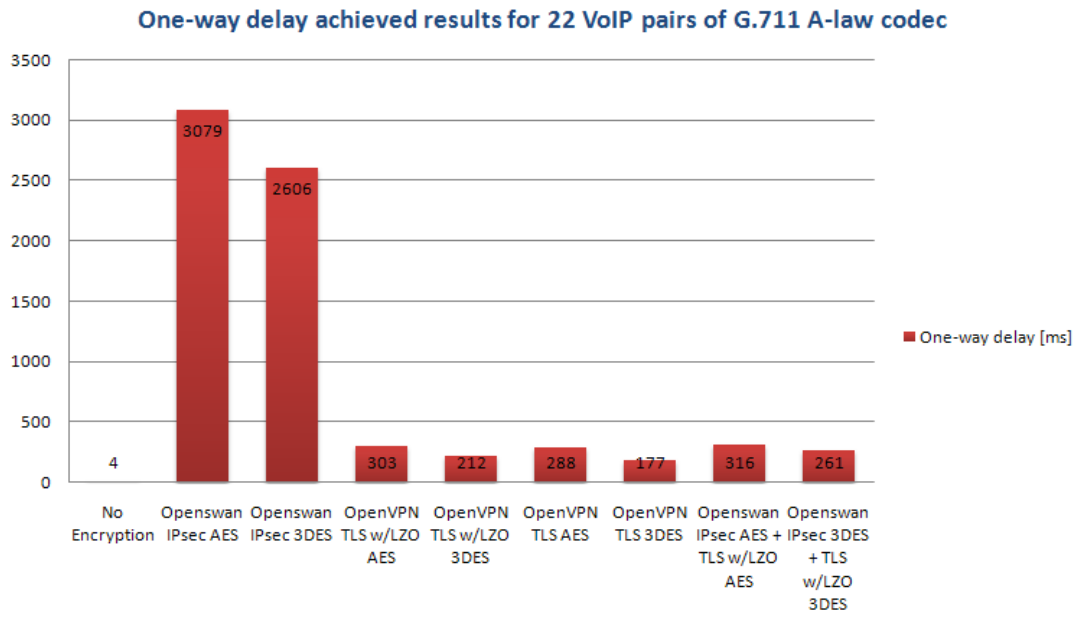

Figure 55: One-way delay achieved results for 22 VoIP pairs of G.711 A-law codec.

G.729

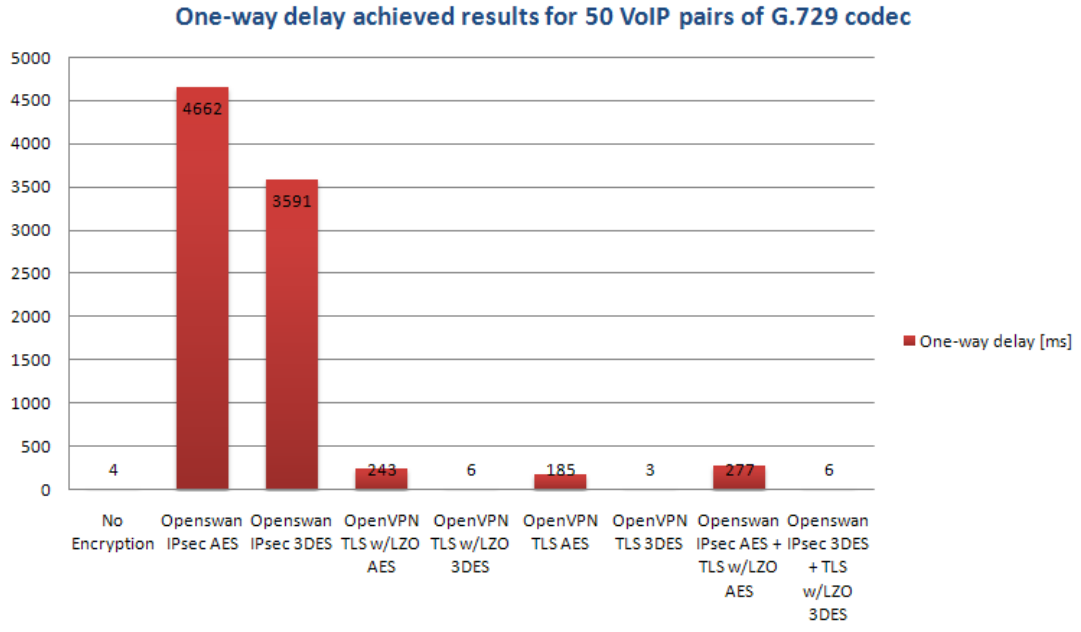

Figure 56: One-way delay achieved results for 50 VoIP pairs of G.729 codec.

9 CONCLUSIONS AND FUTURE WORK

Real-time applications are very sensitive to packet loss and delay, and each variation occurring on the network can modify and influence the final result of a real-time data transmission, such as a VoIP call.

As Voice over IP is an IP-based technology that utilizes the Internet, it also inherits all associated IP vulnerabilities. VoIP is a technology where can be found some security risks and vulnerabilities (see section **3.5**). However, defense techniques using cryptography such as Openswan IPsec and OpenVPN TLS may reduce the danger of security threats. On the other hand, using cryptography reduces the VoIP risks and vulnerabilities but decreases the performance of VoIP services. Applying security mechanisms as Openswan or OpenVPN affect the required bandwidth of IP Telephony. The presented relations throughout chapter **7** should help to understand how the tested security implementations can affect the bandwidth of calls. Through them, it is possible to calculate the bandwidth requirements with high accuracy for the diverse tested communication links.

The impact on performance, with particular attention to speech quality, required bandwidth, transmission delay and lost data has been evaluated through a series of experiments on a real testbed. An expected quality were computed in a simplified E-model, where were taken into account the packet loss and transmission delay. The executed measurements proved the obvious impact of some secure solutions on speech quality.

Increasing bandwidth can cause packet loss and delay, and this way affects the overall speech quality.

Was proved also on chapter **7** that the final bandwidth in Cipher-Block Chaining mode does not depend on key length but on block size. Block size is affected by the cipher method. It has been shown theoretically that an encrypted call with Triple-DES cipher (64 bits block size) requires less network bandwidth when compared with AES cipher (128 bits block size), which in specific circumstances may be reflected on call speech quality, fact that was proved by the realized experiments and can be seen on the presented graphics of subsection **8.6**. Can be observed a negative impact in CBC mode, due the fact that the datagram is divided into same blocks and there is used a padding to keep the block size.

For OpenVPN distinct scenarios with and without LZO compression/decompression mode any relevant difference was achieved. However, by analyzing the graphics average output results, presented in **8.6.1**, can be observed a slight better performance for the case without LZO compression/decompression option enabled, fact that can be a consequence of the compression and decompression time required.

If there is used a mode with many users communicating thru individual encrypted connections, Openswan IPsec solution revels itself a better choice when faced with the OpenVPN TLS one. For multiple individual encrypted connections is better to use Openswan IPsec, because of the increased overhead by OpenVPN. From other point of view, if it is used a mode where many users communicate together using one tunnel, then is better to use the OpenVPN TLS solution rather than Openswan IPsec, due the multiplexing of many RTP streams to one encrypted TLS tunnel (see section **7.5**).

Into the subsection **ITU-T E-Model R-factor Calculation** of chapter **8**, were calculated the R-factor and MOS scores to the used network communication link for the tested codecs, G.711 A-law and G.729. Assuming the best network possible conditions, G.711 A-law has an "excellent" ("toll quality") MOS/R-factor score, while G.729 has only a "good" one. Despite the G.711 A-law better speech quality results for a communication link in good conditions, to large simultaneous amounts of data, be it VoIP calls, or under adverse network conditions, G.729 codec reveals itself a better solution rather than G.711 A-law, because of its lower bandwidth requirements.

The contribution of this dissertation is present how network security mechanisms may impact speech quality. The achieved theoretical results were confirmed in testbed; an experimental study of VoIP under distinct unsecure and secure communication links was performed. The experiments were performed in a real network between Ostrava [73] and Prague [74] universities, and the observed behaviour has been explained. The presented achieved results throughout Appendix **C** will be used for further investigations at Vysoká Škola Báňská - Technická Univerzita Ostrava and be available for consultation.

Secure VoIP communications and the impact of network security on speech quality is far from being a closed chapter. In the case of OpenVPN security solution, comprehend and relate how the multiplexing of the data stream works and behaves in distinct circumstances, and when LZO compression/decompression mode starts have relevant meaning improving the bandwidth usage and consequently the speech quality, are a challenge for a future work. Protocols as SRTP and ZRTP may have a word to declare in terms of VoIP security mechanisms and the study of their behaviour compared with the IPsec and TLS security solutions under security threats is a vast field to be explored.

10 BIBLIOGRAPHY

[1] Wallace, K. (2008). *Cisco Voice over IP (CVOICE)* (Third ed.): Cisco Press

[2] Wallingford, T. (2005). *Switching to VoIP* (First ed.): O'Reilly Media

[3] Vozňák, M. (2008). *Impact of network security on speech quality*. Available http://www.cesnet.cz/doc/techzpravy/2008/impact-of-network-security-on-speech-quality/impact-of-network-security-on-speech-quality.pdf

[4] IXIA. (2008). IxChariot® (Version 6.70 EA). Available http://www.ixiacom.com/

[5] Telchemy. (2009). VQmon/EP. Available http://www.telchemy.com

[6] Richard Gayraud, O. J. (2008). SIPp (Version 3.1). Available http://sipp.sourceforge.net/

[7] Foundation, F. S. (2009). GNU General Public License. Available http://www.gnu.org/

[8] Cisco Systems, I. (2009). Cisco®. Available http://www.cisco.com

[9] Vijay Bollapragada, M. K., S. W. (2005). *IPSec VPN Design* (First ed.): Cisco Press

[10] Project, T. K. (2008). IPsec-Tools (Version 0.7.1). Available http://ipsec-tools.sourceforge.net

[11] Michael Richardson, P. W. (2008). Opensawn (Version 2.6.19). Available http://www.openswan.org

[12] Gilmore, J. (2004). FreeS/WAN Project. Available http://www.freeswan.org

[13] Steffen, A. (2009). strongSwan (Version 4.2.11). Available http://www.strongswan.org

[14] VPN, T. O. S. (2009). OpenVPN™ (Version 2.0.9). Available http://openvpn.net

[15] Instruments®, N. (2009). Observer (Version 13). Available http://www.networkinstruments.com

[16] NLANR/DAST. (2008). Iperf (Version 2.0.4). Available http://iperf.sourceforge.net/

[17] Richasse, N. (2008). JPerf (Version 2.0.0). Available http://code.google.com/p/xjperf/

[18] Combs, G. (2009). Wireshark® (Version 1.0.7). Available http://www.wireshark.org

[19] Jonathan Davidson, J. P. (2000). *Voice over IP Fundamentals* (First ed.): Cisco Press

[20] Session Initiation Protocol. (2002).

[21] Hypertext Transfer Protocol. (1999).

[22] Simple Mail Transfer Protocol. (2008).

[23] Johnston, A. B. (2004). *Sip: Understanding the Session Initiation Protocol* (Second ed.): Artech House

[24] Megaco. (2003).

[25] Media Gateway Control Protocol. (2003).

[26] Internet Protocol version 4. (1981).

[27] Internet Protocol version 6. (1998).

[28] Hain, T. (2008). *A Pragmatic Report on IPv4 Address Space Consumption*: Cisco Systems. Available http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_8-3/ipv4.html

[29] User Datagram Protocol. (1980).

[30] Transmission Control Protocol. (1981).

[31] Real-time Transport Protocol. (2003).

[32] Anton Kos, B. K., S. T. (2002). *Techniques for Performance Improvement of VoIP Applications*. Paper presented at the Electrotechnical Conference.

[33] IANA. (2009). Real-Time Transport Protocol (RTP) Parameters. Available http://www.iana.org/assignments/rtp-parameters

[34] M. Baugher, D. M., Cisco Systems. (2004).

[35] M. Baugher, D. M., Cisco Systems. (2004). *The Secure Real-time Transport Protocol (SRTP)*. Available http://tools.ietf.org/html/rfc3711

[36] Ltd., V. T. (2006). *SRTP (Secure Real-time Transport Protocol)*

[37] Cisco Systems, I. (2006). Voice Over IP - Per Call Bandwidth Consumption. Available http://www.ciscosystems.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml

[38] Olivier Hersent, J. P., D. G. (2005). *IP Telephony: Deploying Voice-over-IP Protocols* (First ed.): John Wiley & Sons, Ltd.

[39] Hurley, M. (2007). *VoIP VULNERABILITIES*. Available http://www.ccip.govt.nz/newsroom/information-notes/ccip-information-note-current.pdf

[40] 2, I.-T. S. G. (2006). *Teletraffic Engineering Handbook* (First ed.): ITU. Available http://www.com.dtu.dk/teletraffic/handbook/telenook.pdf

[41] Franken, L. (1996). *Quality of Service Management: A Model-Based Approach.* University of Twente

[42] Raake, A. (2006). *Speech Quality of VoIP: Assessment and Prediction* (First ed.): John Wiley & Sons, Ltd.

[43] Anderson, J. (2002). *Addressing VoIP Speech Quality with Non-intrusive Measurements*: Agilent Technologies

[44] ITU-T. (1996). *ITU-T Recommendation P.800 - Methods for subjective determination of transmission quality*. Available http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-P.800-199608-I!!PDF-E&type=items

[45] Networks, D. (2007). *Voice Quality beyond IP QoS*

[46] Zue, C. L. (2005). *Modeling and Assessing Secure Voice over IP Performance.* Massacusetts Institute of Technology

[47] Fotios Liotopoulos, T. D., V. K. (2005). *VoRTTeX: Evaluation of perceived VoIP QoS by third-party, using the E-Model*

[48] ITU-T. (2005). *ITU-T Recommendation G.107 (Amendment 1) - The E-model, a computational model for use in transmission planning*. Available http://eu.sabotage.org/www/ITU/G/G107.doc

[49] Vozňák, M. (2009). *Spojovací soustavy: Hodnocení kvality řeči*. Available http://homel.vsb.cz/~voz29/ss/ss_10pr.pdf

[50] ITU-T. (2007). *ITU-T Recommendation G.113 - Transmission impairments due to speech processing*. Available http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.113-200711-I!!PDF-E&type=items

[51] IXIA. (2004). *Assessing VoIP Call Quality Using the E-model*. Available http://www.ixiacom.com/library/white_papers/display?skey=voip_quality

[52] Chuah, C.-N. (2009). *Providing End-to-End QoS for IP based Latency sensitive Applications*: Department of Electrical Engineering and Computer Science, University of California at Berkeley

[53] Roberto Barbieri, D. B., E. R. (2002). *Voice over IPsec: Analysis and Solutions*. Paper presented at the 18th Annual Computer Security Applications Conference.

[54] Medicine, U. o. M. S. o. (2006). confidentiality, integrity, availability (CIA). Available http://privacy.med.miami.edu/glossary/xd_confidentiality_integrity_availability.htm

[55] Gao, L. L. (2006). *Security in VoIP-Current Situation and Necessary Development.* Linköping Institute of Technology

[56] Paul Wouters, K. B. (2006). *Building and Integrating Virtual Private Networks with Openswan* (First ed.): Packt Publishing

[57] Kohel, D. R. (2008). *Cryptography* (First ed.)

[58] Wikipedia. (2009). Cryptography. Available http://en.wikipedia.org/wiki/Cryptography

[59] Corporation, M. (2009). Encryption. Available http://technet.microsoft.com/en-us/library/cc962028(loband).aspx

[60] Wikipedia. (2009). Advanced Encryption Standard. Available http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[61] Shafi Goldwasser, M. B. (2008). *Lecture Notes on Cryptography* (First ed.)

[62] Wikipedia. (2009). Data Encryption Standard. Available http://en.wikipedia.org/wiki/Data_Encryption_Standard

[63] William F. Ehrsam, C. H. W. M., John L. Smith, Walter L. Tuchman. (1976). US Patent 4074066.

[64] Wikipedia. (2009). Block cipher modes of operation. Available http://en.wikipedia.org/wiki/Cipher_block_chaining#Cipher-block_chaining_.28CBC.29

[65] Schneier, B. (1996). *Applied Cryptography* (Second ed.): John Wiley & Sons, Ltd.

[66] S. Kent, B. C., R. A. (1998). *IP Encapsulating Security Payload (ESP)*. Available http://www.ietf.org/rfc/rfc2406.txt

[67] S. Kent, B. C., R. A. (1998). *Security Architecture for the Internet Protocol (IPsec overview) Obsolete by RFC 4301*. Available http://tools.ietf.org/html/rfc2401

[68] Straka, R. (2002). *IPsec VPN Traffic and NAT / NAPT Traversal: Searching for a Universal Solution*: Center for Information Security, University of Tulsa

[69] Transport Layer Security. (2008).

[70] Alan O. Freier, P. K., Paul C. Kocher. (1996). *The SSL Protocol Version 3.0*. Available http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt

[71] Thomas, S. A. (2000). *SSL and TLS Essentials: Securing the Web* (First ed.): John Wiley & Sons, Ltd.

[72] T. Dierks, C., C. A., C. (1999). *The TLS Protocol Version 1.0*. Available http://www.ietf.org/rfc/rfc2246.txt

[73] Vysoká Škola Báňská - Technická Univerzita Ostrava Katedry Telekomunikační. (2009). Available http://kat454.vsb.cz/

[74] České Vysoké Učení Technické v Praze Katedry Telekomunikační. (2009). Available http://www.comtel.cz

[75] IANA - Internet Assigned Numbers Authority. (2009). Available http://www.iana.org

[76] CESNET: Czech NREN Operator. (2009). Available http://www.ces.net

[77] Simena®. (2009). Network Emulator (NE1000). Available http://www.simena.net/NE1000.htm

[78] T. Friedman, R. C., A. C. (2003). *RTP Control Protocol Extended Reports (RTCP XR)*. Available http://www.rfc-editor.org/rfc/rfc3611.txt

[79] ITU-T. (1993). *ITU-T Recommendation G.711 - Puse Code Modulation (PCM) of Voice Frequencies*. Available http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.711-198811-I!!PDF-E&type=items

[80] ITU-T. (1996). *ITU-T Recommendation G.729 - Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP)*. Available http://www.ece.cmu.edu/~ece796/documents/g729.pdf

[81] Oberhumer, M. (2009). LZO real-time data compression library. Available http://www.oberhumer.com/opensource/lzo/

[82] Mirsolav Vozňák, M. H., V. Kyrbashov. (2007). *Factors influencing voice quality in VoIP technology*. Paper presented at the 9th International Conference on Informatics.

[83] I. Baroňák, M. H., M. O. (2007). *Mathematical model of VoIP connection delay*. Paper presented at the Telecommunications, Networks and Systems.

[84] Schroeder, M. (1991). *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. New York: W. H. Freeman & Co.

[85] Wikipedia. (2009). OpenVPN. Available http://en.wikipedia.org/wiki/Openvpn

[86] Cepek, J. (2007). OpenVPN RFC-2246 Compliance Question. Available http://openvpn.net/archive/openvpn-users/2007-07/msg00187.html

[87] Snader, J. (2005). *VPNs Illustrated: Tunnels, VPNs, and IPsec* (First ed.): Addison Wesley Professional. Available http://fengnet.com/book/VPNs%20Illustrated%20Tunnels%20%20VPNsand%20IPsec/

[88] Vincent, L. (2008). lshw (Hardware Lister) (Version B.02.13). Available http://lshw.org

A APPENDIX – MACHINES CHARACTERISTICS

A.1. Hardware Configuration Specifications

lshw (Hardware Lister) [88] is a small tool to provide detailed information on the hardware configuration of a machine. It can report exact memory configuration, firmware version, main board configuration, CPU version and speed, cache configuration, bus speed and so on.

Resorting to this application, was performed the command "lshw -class CLASS" on the terminal of both Ostrava and Prague machines, in order to gather some detailed information of the hardware configuration.

Ostrava Machine

```
~# lshw -class CPU -class memory -class network
  *-firmware
       description: BIOS
       vendor: FUJITSU SIEMENS // Phoenix Technologies Ltd.
       physical id: 0
       version: 5.00 R1.03.1761 (05/14/2004)
       size: 117KiB
       capacity: 448KiB
       capabilities: pci pnp apm upgrade shadowing escd cdboot bootselect int13floppynec
int13floppytoshiba int13floppy360 int13floppy1200 int13floppy720 int13floppy2880
int5printscreen int9keyboard int14serial int17printer int10video acpi usb agp ls120boot
zipboot biosbootspecification
  *-cpu
       description: CPU
       product: Intel(R) Celeron(R) CPU 2.60GHz
       vendor: Intel Corp.
       physical id: 4
       bus info: cpu@0
       version: 15.2.9
       slot: CPU
       size: 2600MHz
       capacity: 2600MHz
       width: 32 bits
       clock: 400MHz
       capabilities: fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe up pebs bts cid xtpr
       configuration: id=0
    *-cache:0
         description: L1 cache
         physical id: 5
         slot: L1 Cache
         size: 8KiB
         capacity: 20KiB
         capabilities: burst synchronous internal write-back data
    *-cache:1
         description: L2 cache
```

```
       physical id: 6
       slot: L2 Cache
       size: 128KiB
       capacity: 512KiB
       capabilities: burst internal write-back unified
 *-memory
     description: System Memory
     physical id: 1f
     slot: System board or motherboard
     size: 1GiB
     capacity: 2GiB
   *-bank:0
       description: DIMM DDR Synchronous 332 MHz (3.0 ns)
       physical id: 0
       slot: DIMM-1
       size: 512MiB
       width: 64 bits
       clock: 332MHz (3.0ns)
   *-bank:1
       description: DIMM DDR Synchronous 332 MHz (3.0 ns)
       physical id: 1
       slot: DIMM-2
       size: 512MiB
       width: 64 bits
       clock: 332MHz (3.0ns)
 *-network
     description: Ethernet interface
     product: RTL-8169 Gigabit Ethernet
     vendor: Realtek Semiconductor Co., Ltd.
     physical id: c
     bus info: pci@0000:00:0c.0
     logical name: eth2
     version: 10
     serial: 00:0e:2e:c6:f3:12
     size: 1GB/s
     capacity: 1GB/s
     width: 32 bits
     clock: 66MHz
     capabilities: pm bus_master cap_list ethernet physical tp 10bt 10bt-fd 100bt 100bt-fd
1000bt-fd autonegotiation
     configuration: autonegotiation=on broadcast=yes driver=r8169 driverversion=2.2LK
duplex=full ip=158.196.81.208 latency=64 link=yes maxlatency=64 mingnt=32 module=r8169
multicast=yes port=twisted pair speed=1GB/s
```

Prague Machine

```
~# lshw -class CPU -class memory -class network
  *-firmware
       description: BIOS
       vendor: American Megatrends Inc.
       physical id: 0
       version: 1018.002 (08/16/2007)
       size: 64KiB
       capacity: 448KiB
       capabilities: isa pci pnp apm upgrade shadowing escd cdboot bootselect socketedrom
edd int13floppy1200 int13floppy720 int13floppy2880 int5printscreen int9keyboard
int14serial int17printer int10video acpi usb agp ls120boot zipboot biosbootspecification
  *-cpu
       description: CPU
       product: AMD Athlon(tm) 64 Processor 3000+
       vendor: Advanced Micro Devices [AMD]
       physical id: 4
       bus info: cpu@0
       version: 15.15.0
       serial: To Be Filled By O.E.M.
       slot: Socket 939
       size: 1800MHz
       capacity: 3600MHz
       width: 64 bits
       clock: 200MHz
       capabilities: boot fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr
pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx mmxext fxsr_opt x86-64
3dnowext 3dnow up lahf_lm cpufreq
     *-cache:0
          description: L1 cache
          physical id: 5
          slot: L1-Cache
          size: 128KiB
          capacity: 128KiB
          capabilities: pipeline-burst internal varies data
     *-cache:1
          description: L2 cache
          physical id: 6
          slot: L2-Cache
          size: 512KiB
          capacity: 512KiB
          capabilities: pipeline-burst internal varies unified
  *-memory
       description: System Memory
       physical id: 3c
       slot: System board or motherboard
       size: 1GiB
     *-bank:0
          description: DIMM DDR Synchronous 400 MHz (2.5 ns)
          product: PartNum0
```

```
        vendor: Manufacturer0
        physical id: 0
        serial: SerNum0
        slot: DIMM0
        size: 1GiB
        width: 64 bits
        clock: 400MHz (2.5ns)
     *-bank:1
        description: DIMM [empty]
        product: PartNum1
        vendor: Manufacturer1
        physical id: 1
        serial: SerNum1
        slot: DIMM1
  *-network
      description: Ethernet interface
      product: 88E8001 Gigabit Ethernet Controller
      vendor: Marvell Technology Group Ltd.
      physical id: a
      bus info: pci@0000:00:0a.0
      logical name: eth0
      version: 13
      serial: 00:11:2f:e0:38:25
      size: 100MB/s
      capacity: 1GB/s
      width: 32 bits
      clock: 66MHz
      capabilities: pm vpd bus_master cap_list ethernet physical tp 10bt 10bt-fd 100bt 100bt-
fd 1000bt 1000bt-fd autonegotiation
      configuration: autonegotiation=on broadcast=yes driver=skge driverversion=1.13
duplex=full firmware=N/A ip=147.32.201.116 latency=64 link=yes maxlatency=31 mingnt=23
module=skge multicast=yes port=twisted pair speed=100MB/s
```

A.2. Software Information Specifications

"lsb_release" is a terminal instruction that prints distribution specific information, as distributer ID, description, release or codename.

uname is a software program in Unix-based operating systems that prints the name, version and other details about the current machine and the operating system running on it. The "uname -r" instruction prints the kernel version of the machine.

The terminal instructions "lsb_release -a" and "uname -r" were performed in both machines to gather the used Operating Systems details.

dpkg is also a software at the base of the Unix-based package management system. dpkg is used to install, remove and provide information about packages. dpkg was used to report the status of specified packages, such as openswan and openvpn, by performing the instruction "dpkg -s" at the Linux Console.

Ostrava Machine

```
~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 8.04.2
Release:        8.04
Codename:       hardy
```

```
~# uname -r
2.6.24-23-generic
```

```
~# dpkg -s openswan
Package: openswan
Status: install ok installed
Priority: optional
Section: net
Installed-Size: 5784
Maintainer: Ubuntu MOTU Developers <ubuntu-motu@lists.ubuntu.com>
Architecture: i386
Version: 1:2.4.9+dfsg-1build1
Provides: ike-server
Depends: bsdmainutils, debianutils (>= 1.7), host, iproute, ipsec-tools, libc6 (>= 2.7-1),
libcurl3 (>= 7.16.2-1), libgmp3c2, libldap-2.4-2 (>= 2.4.7), libpam0g (>= 0.99.7.1), libssl0.9.8
(>= 0.9.8f-1), openssl
Pre-Depends: debconf | debconf-2.0
Suggests: curl, openswan-modules-source | linux-patch-openswan
Conflicts: freeswan (<< 2.04-12)
Conffiles:
 /etc/ipsec.d/policies/clear d6b642e02e9f404c382165cef713d114
 /etc/ipsec.d/policies/clear-or-private 60c09411c9d5a267beb7e914579324d0
 /etc/ipsec.d/policies/private-or-clear d5262151f3fd2bed298670e89f5df26b
 /etc/ipsec.d/policies/private 4446d2670602535162dd5828bcf9fbdc
 /etc/ipsec.d/policies/block 621572de957deb19412e28b920240014
 /etc/ipsec.d/examples/no_oe.conf e2f487a813602cb8ecbf41c88a43e038
 /etc/ipsec.d/examples/l2tp-cert.conf 5c920c1b4105a859841c1382790569c5
 /etc/ipsec.d/examples/l2tp-cert-orgWIN2KXP.conf d13153609e82b8aa7564e363ab843be6
 /etc/ipsec.d/examples/l2tp-psk.conf 7595bb1210ee8a7f2093a70340a363db
 /etc/ipsec.d/examples/l2tp-psk-orgWIN2KXP.conf cdf7130ee02f833f2baeeabbf759e705
 /etc/ipsec.d/examples/linux-linux.conf 61c0e1b217b2ab94e11450295a463e61
 /etc/ipsec.d/examples/sysctl.conf 78e5504a4bef23f32f803b666b824343
 /etc/init.d/ipsec 3d442a900a030cc4df5ed3987d649a24
 /etc/logcheck/ignore.d.paranoid/openswan 5b11ff38eece30e93e45477b880dbe51
 /etc/logcheck/ignore.d.server/openswan cf2c607088873b349abccddca4271cb6
 /etc/logcheck/ignore.d.workstation/openswan cf2c607088873b349abccddca4271cb6
 /etc/logcheck/violations.ignore.d/openswan fca4432e2667e5f34bcba90007b85b1a
 /etc/ipsec.conf ec9a18af6343a5d3348b40720cec623e
 /etc/ipsec.secrets 929e32a3af669e42363ce153904c5833
Description: IPSEC utilities for Openswan
 IPSEC is Internet Protocol SECurity. It uses strong cryptography to provide
```

both authentication and encryption services. Authentication ensures that
packets are from the right sender and have not been altered in transit.
Encryption prevents unauthorised reading of packet contents.
.
This version of Openswan supports Opportunistic Encryption (OE) out of the
box. OE enables you to set up IPsec tunnels to a site without
co-ordinating with the site administrator, and without hand
configuring each tunnel. If enough sites support OE, a "FAX effect"
occurs, and many of us can communicate without eavesdroppers.
.
In addition to OE, you may manually configure secure tunnels through
untrusted networks. Everything passing through the untrusted net is
encrypted by the IPSEC gateway machine and decrypted by the gateway
at the other end. The result is Virtual Private Network or VPN. This
is a network which is effectively private even though it includes
machines at several different sites connected by the insecure Internet.
.
Please note that you will need a recent kernel (>=2.4.24 or 2.6.x)
for using this package. The standard Debian kernel includes both IPSEC
and crypto support, patching the kernel is no longer necessary!
.
If you want to use the KLIPS IPSec code for kernel modules instead of the
native ones, you will need to install either openswan-modules-source or
linux-patch-openswan and build the respective modules for your kernel.
Original-Maintainer: Rene Mayrhofer <rmayr@debian.org>

~# **dpkg -s openvpn**
Package: openvpn
Status: install ok installed
Priority: optional
Section: net
Installed-Size: 1060
Maintainer: Ubuntu Core Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: i386
Version: 2.1~rc7-1ubuntu3.3
Depends: debconf | debconf-2.0, libc6 (>= 2.4), liblzo2-2, libpam0g (>= 0.99.7.1), libssl0.9.8
(>= 0.9.8g-4ubuntu3.1), openssl-blacklist (>> 0.3.2), openvpn-blacklist
Suggests: openssl, resolvconf
Conffiles:
/etc/openvpn/update-resolv-conf 0ad583f96f573add56991ed26e0b463a
/etc/network/if-up.d/openvpn db488b1391f00d828d9a96f127ec117d
/etc/network/if-down.d/openvpn d5e9116a7a9bbf8b3c52c79bc5857d99
/etc/default/openvpn 83d9c4d24b3cbf0c5dd256b6dd6a9175
/etc/init.d/openvpn 0de14ec35143cd97e088c9be95cde615
Description: Virtual Private Network daemon
An application to securely tunnel IP networks over a single UDP or TCP port.
It can be used to access remote sites, make secure point to point connections,
enhance WiFi security, etc.
.
OpenVPN uses all of the encryption, authentication, and certification features
of the OpenSSL library (any cipher, key size, or HMAC digest).

```
.
OpenVPN may use static, pre-shared keys or TLS-based dynamic key exchange.  It
also supports VPNs with dynamic endpoints (DHCP or dial-up clients), tunnels
over NAT or connection-oriented stateful firewalls (like Linux's iptables).
Original-Maintainer: Alberto Gonzalez Iniesta <agi@inittab.org>
```

Prague Machine

```
~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 8.10
Release:        8.10
Codename:       intrepid
```

```
~# uname -r
2.6.27-11-generic
```

```
~# dpkg -s openswan
Package: openswan
Status: install ok installed
Priority: optional
Section: net
Installed-Size: 5756
Maintainer: Ubuntu MOTU Developers <ubuntu-motu@lists.ubuntu.com>
Architecture: i386
Version: 1:2.4.12+dfsg-1.3
Provides: ike-server
Depends: libc6 (>= 2.4), libcurl3 (>= 7.16.2-1), libgmp3c2, libldap-2.4-2 (>= 2.4.7), libpam0g
(>= 0.99.7.1), libssl0.9.8 (>= 0.9.8f-5), bsdmainutils, debianutils (>= 1.7), ipsec-tools, openssl,
host, iproute
Pre-Depends: debconf | debconf-2.0
Suggests: openswan-modules-source | linux-patch-openswan, curl
Conflicts: freeswan (<< 2.04-12)
Conffiles:
 /etc/ipsec.d/policies/clear 68fa01f663438322e2466a7fef91ee0e
 /etc/ipsec.d/policies/clear-or-private f6f10c5863f209d141706c7aa236a9a1
 /etc/ipsec.d/policies/private-or-clear 1c137d74b7ec03f6e4ee5279aa5a7d6d
 /etc/ipsec.d/policies/private cc81dd5453c6500c7490ebfce7ca15b0
 /etc/ipsec.d/policies/block fd634c2e6d4febf11ad73237fe096abc
 /etc/ipsec.d/examples/no_oe.conf 72bdedd5319aedbe2d1a818c2288824f
 /etc/ipsec.d/examples/l2tp-cert.conf 3143ed022d3c15af412b84379bd582d1
 /etc/ipsec.d/examples/l2tp-psk.conf 8a9ed8f2190672b75416c036c640cffe
 /etc/ipsec.d/examples/linux-linux.conf 61c0e1b217b2ab94e11450295a463e61
 /etc/ipsec.d/examples/sysctl.conf 78e5504a4bef23f32f803b666b824343
 /etc/init.d/ipsec 8ad085a483556b81d5f23f44c525151b
 /etc/logcheck/ignore.d.paranoid/openswan 5b11ff38eece30e93e45477b880dbe51
 /etc/logcheck/ignore.d.server/openswan cf2c607088873b349abccddca4271cb6
 /etc/logcheck/ignore.d.workstation/openswan cf2c607088873b349abccddca4271cb6
```

/etc/logcheck/violations.ignore.d/openswan fca4432e2667e5f34bcba90007b85b1a
/etc/ipsec.conf 013835716010ad7e90096feb3ad53e94
/etc/ipsec.secrets 6d4cef765ce350ca5d6c4b4b6a9fed96
Description: IPSEC utilities for Openswan
IPSEC is Internet Protocol SECurity. It uses strong cryptography to provide
both authentication and encryption services. Authentication ensures that
packets are from the right sender and have not been altered in transit.
Encryption prevents unauthorised reading of packet contents.
.
This version of Openswan supports Opportunistic Encryption (OE) out of the
box. OE enables you to set up IPsec tunnels to a site without
co-ordinating with the site administrator, and without hand
configuring each tunnel. If enough sites support OE, a "FAX effect"
occurs, and many of us can communicate without eavesdroppers.
.
In addition to OE, you may manually configure secure tunnels through
untrusted networks. Everything passing through the untrusted net is
encrypted by the IPSEC gateway machine and decrypted by the gateway
at the other end. The result is Virtual Private Network or VPN. This
is a network which is effectively private even though it includes
machines at several different sites connected by the insecure Internet.
.
Please note that you will need a recent kernel (>=2.4.24 or 2.6.x)
for using this package. The standard Debian kernel includes both IPSEC
and crypto support, patching the kernel is no longer necessary!
.
If you want to use the KLIPS IPSec code for kernel modules instead of the
native ones, you will need to install either openswan-modules-source or
linux-patch-openswan and build the respective modules for your kernel.
Original-Maintainer: Rene Mayrhofer <rmayr@debian.org>

~# **dpkg -s openvpn**
Package: openvpn
Status: install ok installed
Priority: optional
Section: net
Installed-Size: 1168
Maintainer: Ubuntu Core Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: i386
Version: 2.1~rc11-1ubuntu2
Depends: debconf | debconf-2.0, libc6 (>= 2.4), liblzo2-2, libpam0g (>= 0.99.7.1), libpkcs11-helper1, libssl0.9.8 (>= 0.9.8g-9), openssl-blacklist (>= 0.4), openvpn-blacklist, lsb-base (>= 3.2-14)
Recommends: net-tools
Suggests: openssl, resolvconf
Conffiles:
/etc/openvpn/update-resolv-conf 0ad583f96f573add56991ed26e0b463a
/etc/network/if-up.d/openvpn ef0c34d3c92b8849f1e04168caa605d3
/etc/network/if-down.d/openvpn 6f027552ae527133e46efb9201e0b9fc
/etc/bash_completion.d/openvpn 5cab8dd1689cc5b338886557cf7a25a9
/etc/default/openvpn bc0812d4608dca7c4be229c572c146e4

```
/etc/init.d/openvpn a529083912a91ca2bd1db0059bfbd8d3
Description: virtual private network daemon
 OpenVPN is an application to securely tunnel IP networks over a
 single UDP or TCP port. It can be used to access remote sites, make
 secure point-to-point connections, enhance wireless security, etc.
 .
 OpenVPN uses all of the encryption, authentication, and certification
 features provided by the OpenSSL library (any cipher, key size, or
 HMAC digest).
 .
 OpenVPN may use static, pre-shared keys or TLS-based dynamic key exchange. It
 also supports VPNs with dynamic endpoints (DHCP or dial-up clients), tunnels
 over NAT or connection-oriented stateful firewalls (such as Linux's iptables).
Original-Maintainer: Alberto Gonzalez Iniesta <agi@inittab.org>
```

B APPENDIX – INSTALLATION GUIDES

B.1. IxChariot Performance Endpoint Installation Tutorial

Firstly, the initial step in order to install an IxChariot Performance Endpoint on an running Ubuntu machine is download the respective Linux packet on the "Downloads & Uploads" section of IXIA webpage (http://www.ixiacom.com/support/endpoint_library/).

After downloaded the respective IxChariot Performance Endpoint tar.gz file, it is necessary to perform on the Linux Console the following instruction, in order to extract the packet content and process the installation:

```
~# ls
IxChariot_Performance_Endpoint 6.50.tar.gz
~# tar -xf IxChariot_Performance_Endpoint_6.50.tar.gz
~# ls
endpoint.install  IxChariot_Performance_Endpoint_6.50.tar.gz  temp
~# ./endpoint.install accept_license
```

B.2. Openswan Installation and Configuration Tutorial

Requirements

To configure a network-to-network (host-to-host) connection it is necessary to have:

- Two Linux gateways;
- Openswan and ipsec-tools installed on both gateways;
- The two machines should either be behind the "same" NAT or have public routable IP addresses, with no firewalls.

Gather information

For each gateway, it is necessary to compile the following information:

- Gateway IP;
- A name by which gateway can identify itself for IPsec negotiations. Its form is a Fully Qualified Domain Name preceded by an @ sign, i.e. @example.com.

NOTE: It does not need to be within a domain that is owned. It can be a made-up name.

Figure 57: Openswan implementation scheme.

Installation/Configuration Step-by-Step

Install Openswan

To install Openswan it is necessary to run on the terminal of each machine the following command:

```
# apt-get install openswan ipsec-tools
```

Start Openswan

To start the IPsec session it is necessary to perform, on both machines, the following command:

```
# service ipsec start
```

Get the leftrsasigkey

On the local Linux Openswan gateway, print the IPsec public key:

```
# ipsec showhostkey --left
```

The output should look similar to:

```
        # RSA 2048 bits   server.com   Mon Feb 23 14:50:57 2009
leftrsasigkey=0sAQPGl/wJ+AMpDehHdRjs0vZRWRig5Gu4nPrfh9OLkojJwznGRlO/qCWVK32O
y5CnyMG5tz1knruOW8FSff4utWoYGc+tmnQM15h593M2/BU3ubNlV/kIOHhs4eGNzDNODxx
CYCIv4w4HBHrfQLodqG9EOrGmb02AfuTrmxjgSQM/ci2G91k5QVdSWDvFyl+UB/U2LRohTLtZi
9nbDCYf7eud9BXJHxi2BTqjLM61as+871tT04vMbDr4NaFX83NuxOtUYgFsxw7jy5aHRiYD+3z4
UCdZeG6p2pPibHx4TqfluiT582TIEfh69JCPc8JUtzkkjt0TtHrsuTGbFsn/n1wl
```

If there are not any key or it shows nothing, it is necessary to do:

```
# ipsec newhostkey --output /etc/ipsec.secret --bits 2048 --hostname server.com
```

Get the rightrsasigkey

On the console at the remote side it is necessary to type:

```
# ipsec showhostkey --right
```

It should show something like:

```
        # RSA 2048 bits   client.com   Mon Feb 23 15:05:14 2009
rightrsasigkey=0sAQPpWAI3RsIGnpnjshI2HGq46iN0htpEl5YQ3BsM/rfUUnQuCi1LruE2wmDz
DGpGxzwsWq7gkhgWsT7G7I75uxxk4MfBAHPIhrUsxYTmTOU+YKnPiWXPFYnllysYqyQsoqav/6
s07kkCBzgRmgkSjqv1eVCwmJaLD8vj7jQkxocFa8dRT8lRItnwEIjBBwp7j8KiBgRU6ivNdDrYxAuU
0Mq5LW+hkCEbaUP0CMOj0TNcVuqRDCKHqO+HkNVCOPPg+TaDWE8Eb26j8FvNCu/ero4vTY
K57aHq/8g/3LMcqqzsZH2bneyjVe6+gsOJLyzV7ktAXPgFh+ObkdacxtNdlcIL
```

Edit /etc/ipsec.conf

Back on the local gateway, it is necessary to edit the template "/etc/ipsec.conf" and substitute the information for the gathered data.

```
# /etc/ipsec.conf - Openswan IPsec configuration file
# RCSID $Id: ipsec.conf.in,v 1.15.2.6 2006/10/19 03:49:46 paul Exp $

# This file:  /usr/share/doc/openswan/ipsec.conf-sample
#
# Manual:     ipsec.conf.5


version 2.0     # conforms to second version of ipsec.conf specification

# basic configuration
config setup
        # plutodebug / klipsdebug = "all", "none" or a combation from below:
        # "raw crypt parsing emitting control klips pfkey natt x509 private"
        # eg: plutodebug="control parsing"
        #
        # ONLY enable plutodebug=all or klipsdebug=all if you are a developer !!
        #
        # NAT-TRAVERSAL support, see README.NAT-Traversal
        nat_traversal=yes
        # virtual_private=%v4:10.0.0.0/8,%v4:192.168.0.0/16,%v4:172.16.0.0/12
        #
        # enable this if you see "failed to find any available worker"
        nhelpers=0

# Add connections here

# sample VPN connections, see /etc/ipsec.d/examples/

#Disable Opportunistic Encryption
include /etc/ipsec.d/examples/no_oe.conf

conn host-to-host
  left=158.196.81.208                           # Local IP address
  leftsubnet=158.196.81.0/24                    # Local netmask
  leftid=@server.com                            #

leftrsasigkey=0sAQPGl/wJ+AMpDehHdRjs0vZRWRig5Gu4nPrfh9OLkojJwznGRlO/qCWVK32O
y5CnyMG5tz1knruOW8FSff4utWoYGc+tmnQM15h593M2/BU3ubNlV/kIOHhs4eGNzDNODxx
CYCIv4w4HBHrfQLodqG9EOrGmb02AfuTrmxjgSQM/ci2G91k5QVdSWDvFyl+UB/U2LRohTLtZi
9nbDCYf7eud9BXJHxi2BTqjLM61as+871tT04vMbDr4NaFX83NuxOtUYgFsxw7jy5aHRiYD+3z4
UCdZeG6p2pPibHx4TqfluiT582TIEfh69JCPc8JUtzkkjt0TtHrsuTGbFsn/n1wl
  leftnexthop=%defaultroute                     # Correct in many situations
  right=147.32.201.116                          # Remote IP address
  rightsubnet=147.32.201.0/24                   # Local netmask
  rightid=@clientPrague.com                     #
```

```
rightrsasigkey=0sAQPpWAI3RsIGnpnjshI2HGq46iN0htpEl5YQ3BsM/rfUUnQuCi1LruE2wmDz
DGpGxzwsWq7gkhgWsT7G7I75uxxk4MfBAHPIhrUsxYTmTOU+YKnPiWXPFYnllysYqyQsoqav/6
s07kkCBzgRmgkSjqv1eVCwmJaLD8vj7jQkxocFa8dRT8lRItnwEIjBBwp7j8KiBgRU6ivNdDrYxAuU
0Mq5LW+hkCEbaUP0CMOj0TNcVuqRDCKHqO+HkNVCOPPg+TaDWE8Eb26j8FvNCu/ero4vTY
K57aHq/8g/3LMcqqzsZH2bneyjVe6+gsOJLyzV7ktAXPgFh+ObkdacxtNdlcIL
  rightnexthop=%defaultroute                    # Correct in many situations
  ike=aes128                                    # IKE algorithms (AES cipher)
  esp=aes128                                    # ESP algorithns (AES cipher)
  #ike=3des                                     # IKE algorithms (3DES cipher)
  #esp=3des                                     # ESP algorithns (3DES cipher)
  auto=start                                    # enables the connection at startup
  auth=ah                                       #

# Lines starting with # will not be read by Openswan (ipsec-tools)
```

NOTE: "Left" and "right" should represent the machines that have Openswan installed on them.

Must be copied the "conn host-to-host" to the remote-side "/etc/ipsec.conf". It can be done with a flash disk or by other ways.

Define Gateways:

```
# route add default gw IP_ADDRESS
```

NOTE: It is necessary define as default gateway the IP of each machine.

Restart Openswan at both sides:

```
# /etc/init.d/ipsec restart
```

Start the connection

Locally, it is necessary to type:

```
# ipsec auto --up host-to-host
```

It should be seen:

```
104 "host-to-host" #1: STATE_MAIN_I1: initiate
003 "host-to-host" #1: ignoring unknown Vendor ID payload [4f45606c50487c5662707575]
003 "host-to-host" #1: received Vendor ID payload [Dead Peer Detection]
003 "host-to-host" #1: received Vendor ID payload [RFC 3947] method set to=110
106 "host-to-host" #1: STATE_MAIN_I2: sent MI2, expecting MR2
003 "host-to-host" #1: NAT-Traversal: Result using RFC 3947 (NAT-Traversal): no NAT
detected
108 "host-to-host" #1: STATE_MAIN_I3: sent MI3, expecting MR3
004 "host-to-host" #1: STATE_MAIN_I4: ISAKMP SA established {auth=OAKLEY_RSA_SIG
cipher=aes_128 prf=oakley_md5 group=modp1536}
117 "host-to-host" #2: STATE_QUICK_I1: initiate
```

```
004 "host-to-host" #2: STATE_QUICK_I2: sent QI2, IPsec SA established {ESP=>0x8152a629
<0xfe7de2f9 xfrm=AES_128-HMAC_MD5 NATD=none DPD=none}
```

NOTE: The SPI# is logged, vendors IDs are logged and the cryptographic parameters are also logged.

The new tunnel protects only net-net traffic, not gateway-gateway, or gateway-subnet.

Commands

Openswan can be started, stopped or restarted after booting using the following ipsec initialization scripts:

```
# service ipsec start
# service ipsec stop
# service ipsec restart
```

B.3. OpenVPN Installation and Configuration Tutorial

Installation/Configuration Step-by-Step

Install OpenVPN

To install OpenVPN it is necessary to run on the terminal of each machine the following command:

```
# apt-get install openvpn
```

Generating the Certificates

To create the certificates must be used the "easy-rsa", a set of scripts included into OpenVPN. For that it is necessary to perform on the Server Linux Terminal the following instructions:

```
# cd /usr/share/doc/openvpn/examples/easy-rsa/2.0
# mkdir /etc/openvpn/easy-rsa
# cp -a * /etc/openvpn/easy-rsa
```

Was created a new folder "/etc/openvpn/easy-rsa" that must have the following content:

```
2.0 build-key build-req make-crl revoke-full
build-ca build-key-pass build-req-pass openssl.cnf sign-req
build-dh build-key-pkcs12 clean-all README.gz vars
build-inter build-key-server list-crl revoke-crt
```

All the configurations must now be done inside of the folder "/etc/openvpn/easy-rsa".

To start must be changed the archive "vars".

```
# nano vars
```

In the end of "vars" archive there are a set of parameters used to generate keys (country, province, city, etc.), that can be edited like:

```
export KEY_COUNTRY=CZ
export KEY_PROVINCE=MORAVIA
export KEY_CITY="OSTRAVA"
export KEY_ORG="VSB"
export KEY_EMAIL="smsparada@ua.pt"
```

Next, to upload the variables inside of the archive "vars" is necessary to run the following commands:

```
# source vars
# ./clean-all
# ./build-ca
```

```
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'ca.key'
-----
```

Then it will be asked to enter information that will be incorporated into the certificate request. What it should be entered is what is called a Distinguished Name.

There are quite a few fields, but some of them can be left in blank. For some fields there are default values. If it is entered '.', the field will be left at blank.

```
Country Name (2 letter code) [CZ]: CZ
State or Province Name (full name) [MORAVIA]: MORAVIA
Locality Name (eg, city) [OSTRAVA]: OSTRAVA
Organization Name (eg, company) [VSB]: VSB
Organizational Unit Name (eg, section) []: FEI
Common Name (eg, your name or your server's hostname) []: SMSP
Email Address [smsparada@ua.pt]: smsparada@ua.pt
```

This will create a folder "/etc/openvpn/easy-rsa/keys" with the following content:

```
ca.crt ca.key index.txt serial
```

To generate the server certificate is used the "build-key-server" script, specifying as parameter the name of the archive which will be used ("server", for example):

```
# cd /etc/openvpn/easy-rsa/
# ./build-key-server server
```

NOTE: Must be used the same information included in the "build-ca".

```
Common Name (eg, your name or your server's hostname) []: SMSP
A challenge password []: ****
Sign the certificate? [y/n]: y
```

```
1 out of 1 certificate requests certified, commit? [y/n] y
```

Next, will be generated the keys used by the clients by running the script "build-key":

```
# ./build-key client
```

```
Generating a 1024 bit RSA private key
.++++++
.....++++++
writing new private key to 'client.key'
NOTE: Must be confirmed the parameters used in "build-key-server".
Common Name (eg, your name or your server's hostname) []: Client
...
Sign the certificate? [y/n]: y
1 out of 1 certificate requests certified, commit? [y/n] y
```

Now must be performed the following command in order to generate the Diffie-Hellman parameters and increase the security:

```
# ./build-dh
# rm keys/*.csr
```

In the end must be found a set of archives inside of "/etc/openvpn/easy-rsa/keys" similar to:

```
ca.crt client.key index.txt server.crt client.crt
ca.key index.txt.attr server.key serial
```

Now it is necessary to install the keys, on both server and client.

For the server we must copy the files "ca.crt", "server.crt", "server.key" and the Diffie-Hellman key (dh1024.pem) to a new folder "/etc/openvpn/keys".

```
# cd /etc/openvpn/easy-rsa/keys
# mkdir /etc/openvpn/keys
# cp -a ca.crt server.crt server.key /etc/openvpn/keys/
# cp -a dh1024.pem /etc/openvpn/keys/
```

All the clients must have the archives "ca.crt", "dh1024.pem" and all the ".crt" and ".key" correspondent files. On the client side it is necessary, as well to create a new folder "/etc/openvpn/keys" and copy the files into.

Synchronizing the Server-Client clocks

All the machine clocks must be synchronized. For that purpose, the following command must be executed on both, server and client machines:

```
# ntpdate -u pool.ntp.org
```

Figure 58: OpenVPN implementation scheme.

Server configuration file

Must be created, on the server machine, the configuration archive used by the OpenVPN.

```
# nano /etc/openvpn/server.conf
```

The archive "/etc/openvpn/server.conf" must have the following content:

```
local 158.196.81.208                # Local IP address that OpenVPN should listen on
dev tun0                            # Routed IP tunnel
ifconfig 10.8.0.1 10.8.0.2          # Tunnel IP addresses (Server – Client)
tls-server                          # TLS Server
proto tcp-server                    # TCP Server
port 1194                           # TCP/UDP port that OpenVPN should listen on

ca /etc/openvpn/keys/ca.crt         # SSL/TLS root certificate
cert /etc/openvpn/keys/server.crt   # Certificate
key /etc/openvpn/keys/server.key    # Private key (this file should be kept secret)
dh /etc/openvpn/keys/dh1024.pem     # Diffie-Hellman parameters

keepalive 10 120                    # Ping every 10 seconds, assume that remote peer
                                    # is down if no ping received during a 120
                                    # second time period.

;cipher AES-128-CBC                # AES cipher - 128 bit default key (fixed)
;cipher DES-EDE3-CBC               # 3DES cipher - 192 bit default key (fixed)

auth none                           #
comp-lzo                            # Enable compression on the VPN link
max-clients 100                     # Assign the maximum number of clients

# The persist options will try to avoid accessing certain resources on restart that may no
# longer be accessible because of the privilege downgrade
persist-key
persist-tun

# Output a short status file showing current connections, truncated and rewritten every
# minute.
status openvpn-status.log
```

```
# Set the appropriate level of log file verbosity (3 - reasonable for general usage)
verb 3

# Lines starting with # or ; will not be read by OpenVPN
```

Client configuration file

Analogously, must be created the client configuration file used by the OpenVPN. The archive "/etc/openvpn/client.conf", on the client machine, must be like:

```
dev tun0                                # Routed IP tunnel
remote 158.196.81.208                   # IP address of the Server
ifconfig 10.8.0.2 10.8.0.1              # Tunnel IP addresses (Client – Server)
tls-client                              # TLS Client
proto tcp-client                        # TCP Client
port 1194                               # TCP/UDP port that OpenVPN should listen on


ca /etc/openvpn/keys/ca.crt             # SSL/TLS root certificate
cert /etc/openvpn/keys/client.crt       # Certificate
key /etc/openvpn/keys/client.key        # Private key (this file should be kept secret)
dh /etc/openvpn/keys/dh1024.pem         # Diffie-Hellman parameters

keepalive 10 120                        # Ping every 10 seconds, assume that remote peer
                                        # is down if no ping received during a 120
                                        # second time period.


;cipher AES-128-CBC                     # AES cipher - 128 bit default key (fixed)
;cipher DES-EDE3-CBC                    # 3DES cipher - 192 bit default key (fixed)

auth none                               #
comp-lzo                                # Enable compression on the VPN link

# The persist options will try to avoid accessing certain resources on restart that may no
# longer be accessible because of the privilege downgrade
persist-key
persist-tun

# Output a short status file showing current connections, truncated and rewritten every
# minute.
status openvpn-status.log

# Set the appropriate level of log file verbosity (3 - reasonable for general usage)
verb 3

# Lines starting with # or ; will not be read by OpenVPN
```

Restart OpenVPN

After performed all the configurations on both, client and server machines, it is necessary to restart the OpenVPN in order to apply the new set configurations:

```
# /etc/init.d/openvpn restart
```

Now when performed the instruction "ifconfig tun" on the server side, it should show an output similar to:

```
# ifconfig tun
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.1  P-t-P:10.8.0.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

NOTE: The output of the command "ifconfig tun" on the client side must be similar.

Commands

OpenVPN can be started, stopped and restarted using the following commands:

```
# service openvpn start
# service openvpn stop
# service openvpn restart
```

C APPENDIX – IXCHARIOT CONSOLE ACHIEVED RESULTS

Throughout this appendix are exposed the results and their averages, of the performed VoIP tests by the IxChariot software. There are presented the results of three distinct codecs (G.711 A-law PLC, G.711 A-law and G.729) for unsecure and different secure connections.

C.1. Codec G.711 A-law with Packet Loss Concealment (PLC)

Unsecure connection

Table 13: IxChariot achieved results for the codec G.711a PLC in an unsecure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	90.99	3	0.000
1	20	4.36	91.04	0	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1 Pair Average	20	4.36	91.03	1	0.000
Variance	-	0.0000	0.0008	2.0000	0.0000
19	20	4.36	90.95	5	0.000
19	20	4.36	91.03	1	0.000
19	20	4.36	90.97	4	0.000
19	20	4.36	91.03	1	0.000
19	20	4.36	91.03	1	0.000
19 Pairs Average	20	4.36	91.00	2	0.000
Variance	-	0.0000	0.0015	3.8000	0.0000
21	20	4.36	90.97	4	0.000
21	20	4.36	91.01	2	0.000
21	20	4.36	90.95	5	0.000
21	20	4.36	91.01	2	0.000
21	20	4.36	91.00	2	0.000
21 Pairs Average	20	4.36	90.99	3	0.000
Variance	-	0.0000	0.0007	2.0000	0.0000
22	20	4.36	90.94	5	0.000
22	20	4.36	91.00	2	0.000
22	20	4.36	90.96	4	0.000
22	20	4.36	90.99	6	0.000
22	20	4.36	90.99	2	0.000
22 Pairs Average	20	4.36	90.98	4	0.000
Variance	-	0.0000	0.0006	3.2000	0.0000
23	20	4.36	90.90	7	0.000
23	20	4.36	90.85	10	0.000
23	20	4.36	90.88	8	0.000
23	20	4.36	90.94	5	0.000
23	20	4.36	90.88	8	0.000
23 Pairs Average	20	4.36	90.89	8	0.000
Variance	-	0.0000	0.0011	3.3000	0.0000

Openswan IPsec with AES cipher secure connection

Table 14: IxChariot achieved results for the codec G.711a PLC in an Openswan IPsec with AES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.04	1	0.000
1	20	4.36	90.99	3	0.000
1	20	4.36	91.04	0	0.000
1	20	4.36	91.01	2	0.000
1	20	4.36	91.03	1	0.000
1 Pair Average	20	4.36	91.02	1	0.000
Variance	-	0.0000	0.0005	1.3000	0.0000
19	20	2.45	47.78	851	0.000
19	20	2.44	47.52	858	0.000
19	20	2.45	47.90	848	0.000
19	20	2.45	47.74	852	0.000
19	20	2.48	48.37	834	0.000
19 Pairs Average	20	2.45	47.86	849	0.000
Variance	-	0.0002	0.0995	79.8000	0.0000
21	20	1.90	35.98	2338	0.000
21	20	1.89	35.97	2365	0.000
21	20	1.90	36.07	2336	0.000
21	20	1.90	35.94	2346	0.000
21	20	1.90	35.94	2346	0.000
21 Pairs Average	20	1.90	35.98	2346	0.000
Variance	-	0.0000	0.0029	131.2000	0.0000
22	20	1.79	33.59	3081	0.000
22	20	1.79	33.67	3074	0.000
22	20	1.79	33.62	3079	0.000
22	20	1.79	33.62	3092	0.000
22	20	1.79	33.65	3076	0.000
22 Pairs Average	20	1.79	33.63	3080	0.000
Variance	-	0.0000	0.0010	49.3000	0.0000
23	20	1.71	31.83	3837	0.000
23	20	1.71	31.84	3829	0.000
23	20	1.71	31.80	3834	0.000
23	20	1.71	31.85	3828	0.000
23	20	1.71	31.80	3834	0.000
23 Pairs Average	20	1.71	31.82	3832	0.000
Variance	-	0.0000	0.0005	14.3000	0.000

Openswan IPsec with Triple-DES cipher secure connection

Table 15: IxChariot achieved results for the codec G.711a PLC in an Openswan IPsec with Triple-DES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.02	2	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	90.99	3	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.01	2	0.000
1 Pair Average	20	4.36	91.01	2	0.000
Variance	-	0.0000	0.0005	1.2000	0.0000
19	20	2.98	58.67	448	0.000
19	20	2.96	58.41	453	0.000
19	20	2.98	58.75	450	0.000
19	20	2.98	58.66	449	0.000
19	20	2.98	58.41	453	0.000
19 Pairs Average	20	2.98	58.58	451	0.000
Variance	-	0.0001	0.0253	5.3000	0.0000
21	20	2.00	38.10	1889	0.000
21	20	1.99	38.05	1884	0.000
21	20	2.00	38.14	1879	0.000
21	20	1.99	38.05	1884	0.000
21	20	2.00	38.24	1874	0.000
21 Pairs Average	20	2.00	38.12	1882	0.000
Variance	-	0.0000	0.0062	32.5000	0.0000
22	20	1.86	35.03	2606	0.000
22	20	1.86	35.08	2603	0.000
22	20	1.85	34.98	2621	0.000
22	20	1.86	35.08	2603	0.000
22	20	1.86	35.09	2601	0.000
22 Pairs Average	20	1.86	35.05	2607	0.000
Variance	-	0.0000	0.0022	66.2000	0.0000
23	20	1.76	32.95	3330	0.000
23	20	1.76	32.95	3331	0.000
23	20	1.76	32.94	3340	0.000
23	20	1.76	32.95	3331	0.000
23	20	1.76	33.00	3326	0.000
23 Pairs Average	20	1.76	32.96	3332	0.000
Variance	-	0.0000	0.0006	26.3000	0.0000

OpenVPN TLS with AES cipher and LZO compression/decompression secure connection

Table 16: IxChariot achieved results for the codec G.711a PLC in an OpenVPN TLS with AES cipher and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	90.92	6	0.000
1	20	4.36	90.93	6	0.000
1	20	4.36	90.92	6	0.000
1	20	4.36	90.92	6	0.000
1	20	4.36	90.92	6	0.000
1 Pair Average	20	4.36	90.92	6	0.000
Variance	-	0.0000	0.0000	0.0000	0.0000
19	20	4.36	90.93	5	0.000
19	20	4.36	90.98	3	0.000
19	20	4.36	90.91	6	0.000
19	20	4.36	90.81	11	0.000
19	20	4.36	90.92	6	0.000
19 Pairs Average	20	4.36	90.91	6	0.000
Variance	-	0.0000	0.0039	8.7000	0.0000
21	20	4.28	87.76	84	0.000
21	20	4.26	86.71	95	0.000
21	20	4.24	86.63	98	0.000
21	20	4.22	85.96	105	0.000
21	20	4.29	88.24	74	0.000
21 Pairs Average	20	4.26	87.06	91	0.000
Variance	-	0.0008	0.8500	149.7000	0.0000
22	20	2.91	56.80	287	3.894
22	20	2.94	56.87	294	3.664
22	20	2.89	55.77	308	3.648
22	20	2.85	55.44	304	3.842
22	20	2.89	55.69	303	3.870
22 Pairs Average	20	2.90	56.11	299	3.784
Variance	-	0.0011	0.4486	72.7000	0.0139
23	20	2.61	47.65	314	7.712
23	20	2.31	44.12	319	8.061
23	20	2.41	45.67	314	7.736
23	20	2.39	45.17	324	7.620
23	20	2.37	44.49	324	7.968
23 Pairs Average	20	2.42	45.42	319	7.819
Variance	-	0.0129	1.9132	25.0000	0.0347

OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection

Table 17: IxChariot achieved results for the codec G.711a PLC in an OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	90.93	c	0.000
1	20	4.36	90.93	5	0.000
1	20	4.36	90.97	3	0.000
1	20	4.36	90.96	4	0.000
1	20	4.36	90.92	6	0.000
1 Pair Average	20	4.36	90.94	5	0.000
Variance	-	0.0000	0.0005	0.0005	0.0000
19	20	4.36	91.02	1	0.000
19	20	4.36	91.05	0	0.000
19	20	4.36	91.05	0	0.000
19	20	4.36	91.05	0	0.000
19	20	4.36	90.93	5	0.000
19 Pairs Average	20	4.36	91.02	1	0.000
Variance	-	0.0000	0.0027	4.7000	0.0000
21	20	4.36	90.95	5	0.000
21	20	4.36	90.88	8	0.000
21	20	4.36	90.82	11	0.000
21	20	4.36	90.94	5	0.000
21	20	4.36	90.86	9	0.000
21 Pairs Average	20	4.36	90.89	8	0.000
Variance	-	0.0000	0.0030	6.8000	0.0000
22	20	3.70	73.66	205	0.352
22	20	3.71	73.79	204	0.354
22	20	3.70	73.57	207	0.347
22	20	3.66	73.71	212	0.424
22	20	3.70	73.66	204	0.418
22 Pairs Average	20	3.69	73.68	206	0.379
Variance	-	0.0004	0.0065	11.3000	0.0015
23	20	2.81	53.72	313	4.661
23	20	2.78	52.94	313	4.684
23	20	2.78	53.13	313	4.614
23	20	2.79	53.14	313	4.586
23	20	2.80	53.20	313	4.707
23 Pairs Average	20	2.79	53.23	313	4.650
Variance	-	0.0002	0.0858	0.0000	0.0025

OpenVPN TLS with AES cipher secure connection

Table 18: IxChariot achieved results for the codec G.711a PLC in an OpenVPN TLS with AES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	90.87	1	0.000
1	20	4.36	91.04	0	0.000
1	20	4.36	90.88	1	0.000
1	20	4.36	91.03	1	0.000
1	20	4.36	91.02	1	0.000
1 Pair Average	20	4.36	90.97	1	0.000
Variance	-	0.0000	0.0073	0.2000	0.0000
19	20	4.36	90.84	7	0.000
19	20	4.36	90.91	7	0.000
19	20	4.36	90.89	8	0.000
19	20	4.36	90.79	12	0.000
19	20	4.36	90.88	8	0.000
19 Pairs Average	20	4.36	90.86	8	0.000
Variance	-	0.0000	0.0023	4.3000	0.0000
21	20	4.31	88.78	67	0.000
21	20	4.29	88.29	78	0.000
21	20	4.26	87.37	90	0.000
21	20	4.24	86.57	98	0.000
21	20	4.32	89.18	60	0.000
21 Pairs Average	20	4.28	88.04	79	0.000
Variance	-	0.0011	1.1299	246.8000	0.0000
22	20	2.89	56.23	293	3.933
22	20	2.94	57.18	286	3.845
22	20	2.87	55.96	297	3.900
22	20	2.86	55.34	308	3.885
22	20	2.93	56.84	287	3.906
22 Pairs Average	20	2.90	56.31	294	3.894
Variance	-	0.0013	0.5269	79.7000	0.0010
23	20	2.50	46.10	309	7.991
23	20	2.47	45.33	324	8.017
23	20	2.44	45.13	319	8.075
23	20	2.54	46.40	318	8.003
23	20	2.53	46.50	313	7.867
23 Pairs Average	20	2.50	45.89	317	7.991
Variance	-	0.0017	0.3919	33.3000	0.0058

OpenVPN TLS with Triple-DES cipher

Table 19: IxChariot achieved results for the codec G.711a PLC in an OpenVPN TLS with Triple-DES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	90.99	3	0.000
1	20	4.36	91.03	1	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1 Pair Average	20	4.36	91.03	1	0.000
Variance	-	0.0000	0.0007	1.7000	0.0000
19	20	4.36	91.05	0	0.000
19	20	4.36	91.04	0	0.000
19	20	4.36	91.05	1	0.000
19	20	4.36	90.98	3	0.000
19	20	4.36	91.05	0	0.000
19 Pairs Average	20	4.36	91.03	1	0.000
Variance	-	0.0000	0.0009	1.7000	0.0000
21	20	4.36	91.03	1	0.000
21	20	4.36	91.01	2	0.000
21	20	4.36	90.98	3	0.000
21	20	4.36	91.05	0	0.000
21	20	4.36	91.01	2	0.000
21 Pairs Average	20	4.36	91.03	2	0.000
Variance	-	0.0000	0.0007	1.3000	0.0000
22	20	3.76	75.08	185	0.455
22	20	3.73	74.24	194	0.467
22	20	3.62	71.87	218	0.527
22	20	3.65	72.62	216	0.404
22	20	3.80	75.97	177	0.433
22 Pairs Average	20	3.71	73.96	198	0.457
Variance	-	0.0057	2.8841	337.5000	0.0021
23	20	2.73	53.23	303	4.742
23	20	2.72	52.96	303	4.788
23	20	2.79	53.28	310	4.670
23	20	2.71	53.07	302	4.754
23	20	2.79	53.30	310	4.684
23 Pairs Average	20	2.75	53.17	306	4.728
Variance	-	0.0015	0.0217	16.3000	0.0024

Openswan IPsec plus OpenVPN TLS with AES ciphers and LZO compression/decompression secure connection

Table 20: IxChariot achieved results for the codec G.711a PLC in an Openswan IPsec plus OpenVPN TLS with AES ciphers and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	90.98	3	0.000
1	20	4.36	90.97	4	0.000
1	20	4.36	90.99	3	0.000
1	20	4.36	90.98	3	0.000
1	20	4.36	91.02	2	0.000
1 Pair Average	20	4.36	90.99	3	0.000
Variance	-	0.0000	0.0004	0.5000	0.0000
19	20	4.36	90.95	4	0.000
19	20	4.36	90.90	7	0.000
19	20	4.36	90.93	5	0.000
19	20	4.36	90.85	9	0.000
19	20	4.36	90.94	5	0.000
19 Pairs Average	20	4.36	90.91	6	0.000
Variance	-	0.0000	0.0016	4.0000	0.0000
21	20	3.04	59.18	276	3.352
21	20	2.99	57.64	296	3.305
21	20	2.95	57.15	302	3.254
21	20	2.86	56.10	309	3.375
21	20	3.15	61.16	257	3.317
21 Pairs Average	20	3.00	58.25	288	3.321
Variance	-	0.0116	3.8844	451.5000	0.0022
22	20	2.60	49.33	277	7.679
22	20	2.71	50.24	277	7.436
22	20	2.55	48.44	289	7.391
22	20	2.65	49.08	293	7.448
22	20	2.60	49.38	279	7.648
22 Pairs Average	20	2.62	49.29	283	7.520
Variance	-	0.0037	0.4197	56.0000	0.0176
23	20	1.91	34.57	328	11.745
23	20	2.19	36.59	351	11.490
23	20	2.11	35.60	352	11.701
23	20	1.95	34.11	346	11.719
23	20	2.28	38.82	332	11.478
23 Pairs Average	20	2.09	35.94	342	11.627
Variance	-	0.0246	3.5146	123.2000	0.0172

Openswan IPsec plus OpenVPN TLS with Triple-DES ciphers and LZO compression/decompression secure connection

Table 21: IxChariot achieved results for the codec G.711a PLC in an Openswan IPsec plus OpenVPN TLS with Triple-DES ciphers and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.05	0	0.000
1	20	4.36	91.02	1	0.000
1	20	4.36	90.96	4	0.000
1	20	4.36	91.02	1	0.000
1	20	4.36	91.04	0	0.000
1 Pair Average	20	4.36	91.02	1	0.000
Variance	-	0.0000	0.0012	2.7000	0.0000
19	20	4.36	90.84	10	0.000
19	20	4.36	91.00	2	0.000
19	20	4.36	90.92	6	0.000
19	20	4.36	90.90	7	0.000
19	20	4.36	90.95	5	0.000
19 Pairs Average	20	4.36	90.92	6	0.000
Variance	-	0.0000	0.0035	8.5000	0.0000
21	20	4.27	87.68	82	0.000
21	20	4.26	87.46	82	0.000
21	20	4.27	87.55	85	0.000
21	20	4.24	86.81	94	0.000
21	20	4.28	87.95	78	0.000
21 Pairs Average	20	4.26	87.49	84	0.000
Variance	-	0.0002	0.1787	36.2000	0.0000
22	20	2.97	56.98	292	3.845
22	20	3.03	58.25	279	3.703
22	20	3.02	57.95	284	3.764
22	20	3.01	58.37	277	3.667
22	20	2.94	57.55	284	3.645
22 Pairs Average	20	2.99	57.82	283	3.725
Variance	-	0.0014	0.3207	33.7000	0.0065
23	20	2.45	46.92	294	7.835
23	20	2.35	44.67	318	7.988
23	20	2.52	46.09	324	7.738
23	20	2.29	43.87	317	7.113
23	20	2.61	47.82	300	7.736
23 Pairs Average	20	2.44	45.87	311	7.682
Variance	-	0.0165	2.5983	165.8000	0.1117

C.2. Codec G.711 A-law

Unsecure connection

Table 22: IxChariot achieved results for the codec G.711a in an unsecure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1 Pair Average	20	4.36	91.00	2	0.000
Variance	-	0.0000	0.0000	0.0000	0.0000
19	20	4.36	90.98	3	0.000
19	20	4.36	91.02	1	0.000
19	20	4.36	90.98	3	0.000
19	20	4.36	91.04	1	0.000
19	20	4.36	91.04	0	0.000
19 Pairs Average	20	4.36	91.01	2	0.000
Variance	-	0.0000	0.0009	1.8000	0.0000
21	20	4.36	91.00	2	0.000
21	20	4.36	90.99	3	0.000
21	20	4.36	91.00	2	0.000
21	20	4.36	90.96	4	0.000
21	20	4.36	90.97	4	0.000
21 Pairs Average	20	4.36	90.98	3	0.000
Variance	-	0.0000	0.0003	1.0000	0.0000
22	20	4.36	90.93	5	0.000
22	20	4.36	91.00	2	0.000
22	20	4.36	90.98	3	0.000
22	20	4.36	90.94	5	0.000
22	20	4.36	90.96	4	0.000
22 Pairs Average	20	4.36	90.96	4	0.000
Variance	-	0.0000	0.0008	1.7000	0.0000
23	20	4.36	90.81	11	0.000
23	20	4.36	90.80	12	0.000
23	20	4.36	90.81	11	0.000
23	20	4.36	90.78	12	0.000
23	20	4.36	90.79	12	0.000
23 Pairs Average	20	4.36	90.80	12	0.000
Variance	-	0.0000	1.7000	0.3000	0.0000

Openswan IPsec with AES cipher secure connection

Table 23: IxChariot achieved results for the codec G.711a in an Openswan IPsec with AES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1 Pair Average	20	4.36	91.00	2	0.000
Variance	-	0.0000	0.0000	0.0000	0.0000
19	20	2.47	48.22	839	0.000
19	20	2.50	48.91	817	0.000
19	20	2.49	48.68	827	0.000
19	20	2.49	48.62	837	0.000
19	20	2.48	48.37	834	0.000
19 Pairs Average	20	2.49	48.56	831	0.000
Variance	-	0.0001	0.0730	80.2000	0.0000
21	20	1.90	36.03	2348	0.000
21	20	1.90	35.99	2338	0.000
21	20	1.90	36.06	2336	0.000
21	20	1.90	35.99	2337	0.000
21	20	1.90	35.98	2337	0.000
21 Pairs Average	20	1.90	36.01	2339	0.000
Variance	-	0.0000	0.0012	24.7000	0.0000
22	20	1.79	33.59	3081	0.000
22	20	1.79	33.65	3078	0.000
22	20	1.79	33.59	3082	0.000
22	20	1.79	33.64	3076	0.000
22	20	1.79	33.66	3076	0.000
22 Pairs Average	20	1.79	33.63	3079	0.000
Variance	-	0.0000	0.0011	7.8000	0.0000
23	20	1.71	31.82	3834	0.000
23	20	1.71	31.81	3835	0.000
23	20	1.71	31.82	3834	0.000
23	20	1.71	31.81	3835	0.000
23	20	1.71	31.82	3834	0.000
23 Pairs Average	20	1.71	31.82	3834	0.000
Variance	-	0.0000	0.0000	0.3000	0.0000

Openswan IPsec with Triple-DES cipher secure connection

Table 24: IxChariot achieved results for the codec G.711a in an Openswan IPsec with Triple-DES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.00	2	0.000
1	20	4.36	90.98	3	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	90.99	3	0.000
1 Pair Average	20	4.36	90.99	2	0.000
Variance	-	0.0000	0.0001	0.3000	0.0000
19	20	2.99	58.92	444	0.000
19	20	2.97	58.56	451	0.000
19	20	2.97	58.42	453	0.000
19	20	2.99	58.95	444	0.000
19	20	2.96	58.31	455	0.000
19 Pairs Average	20	2.98	58.63	449	0.000
Variance	-	0.0002	0.0845	26.3000	0.0000
21	20	1.99	38.04	1884	0.000
21	20	2.00	38.14	1880	0.000
21	20	1.99	38.04	1884	0.000
21	20	2.01	38.47	1884	0.000
21	20	2.00	38.14	1880	0.000
21 Pairs Average	20	2.00	38.17	1882	0.000
Variance	-	0.0001	0.0314	4.8000	0.0000
22	20	1.86	35.03	2607	0.000
22	20	1.86	35.07	2610	0.000
22	20	1.86	35.03	2607	0.000
22	20	1.86	35.06	2604	0.000
22	20	1.86	35.09	2602	0.000
22 Pairs Average	20	1.86	35.06	2606	0.000
Variance	-	0.0000	0.0007	9.5000	0.0000
23	20	1.76	32.97	3331	0.000
23	20	1.76	32.96	3332	0.000
23	20	1.76	32.96	3332	0.000
23	20	1.76	32.97	3332	0.000
23	20	1.76	32.93	3348	0.000
23 Pairs Average	20	1.76	32.96	3335	0.000
Variance	-	0.0000	0.0003	53.0000	0.0000

OpenVPN TLS with AES cipher and LZO compression/decompression secure connection

Table 25: IxChariot achieved results for the codec G.711a in an OpenVPN TLS with AES cipher and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.04	0	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.05	0	0.000
1 Pair Average	20	4.36	91.04	0	0.000
Variance	-	0.0000	0.0005	0.8000	0.0000
19	20	4.36	90.96	4	0.000
19	20	4.36	90.93	6	0.000
19	20	4.36	90.91	6	0.000
19	20	4.36	90.95	5	0.000
19	20	4.36	90.93	5	0.000
19 Pairs Average	20	4.36	90.94	5	0.000
Variance	-	0.0000	0.0004	0.7000	0.0000
21	20	4.30	88.62	70	0.000
21	20	4.25	86.81	96	0.000
21	20	4.23	86.29	102	0.000
21	20	4.26	87.30	92	0.000
21	20	4.29	88.20	76	0.000
21 Pairs Average	20	4.27	87.44	87	0.000
Variance	-	0.0008	0.9272	185.2000	0.0000
22	20	1.79	27.14	291	3.855
22	20	1.71	25.08	304	3.915
22	20	1.96	31.82	315	3.591
22	20	1.96	32.46	308	3.833
22	20	2.01	33.76	296	3.673
22 Pairs Average	20	1.89	30.05	303	3.773
Variance	-	0.0166	13.9685	90.7000	0.0184
23	20	1.60	21.02	289	7.988
23	20	1.62	22.67	293	7.939
23	20	1.76	25.22	296	7.710
23	20	1.57	19.72	301	8.000
23	20	1.58	21.46	315	7.591
23 Pairs Average	20	1.63	22.02	299	7.846
Variance	-	0.0060	4.3165	101.2000	0.0340

OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection

Table 26: IxChariot achieved results for the codec G.711a in an OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.00	2	0.000
1	20	4.36	90.99	3	0.000
1	20	4.36	90.98	3	0.000
1	20	4.36	91.00	2	0.000
1	20	4.36	91.01	2	0.000
1 Pair Average	20	4.36	91.00	2	0.000
Variance	-	0.0000	0.0001	0.3000	0.0000
19	20	4.36	91.05	0	0.000
19	20	4.36	91.02	1	0.000
19	20	4.36	90.97	4	0.000
19	20	4.36	91.03	1	0.000
19	20	4.36	91.05	0	0.000
19 Pairs Average	20	4.36	91.02	1	0.000
Variance	-	0.0000	0.0011	2.7000	0.0000
21	20	4.36	91.00	2	0.000
21	20	4.36	90.95	4	0.000
21	20	4.36	90.98	3	0.000
21	20	4.36	90.99	3	0.000
21	20	4.36	91.00	2	0.000
21 Pairs Average	20	4.36	90.98	3	0.000
Variance	-	0.0000	0.0004	0.7000	0.0000
22	20	3.35	65.92	226	0.418
22	20	3.50	69.09	206	0.404
22	20	3.50	69.24	204	0.362
22	20	3.43	67.63	213	0.386
22	20	3.29	65.04	212	0.470
22 Pairs Average	20	3.41	67.38	212	0.408
Variance	-	0.0086	3.5133	74.2000	0.0016
23	20	1.95	32.69	255	4.481
23	20	1.82	29.09	277	4.681
23	20	1.96	32.57	294	4.638
23	20	1.89	31.47	300	4.522
23	20	1.82	29.39	300	4.704
23 Pairs Average	20	1.89	31.04	285	4.605
Variance	-	0.0046	2.9433	373.3000	0.0097

OpenVPN TLS with AES cipher secure connection

Table 27: IxChariot achieved results for the codec G.711a in an OpenVPN TLS with AES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.03	1	0.000
1	20	4.36	91.03	1	0.000
1	20	4.36	90.97	3	0.000
1	20	4.36	91.01	2	0.000
1	20	4.36	91.04	0	0.000
1 Pair Average	20	4.36	91.02	1	0.000
Variance	-	0.0000	0.0008	1.3000	0.0000
19	20	4.36	91.03	1	0.000
19	20	4.36	91.05	0	0.000
19	20	4.36	91.05	0	0.000
19	20	4.36	91.05	0	0.000
19	20	4.36	91.05	0	0.000
19 Pairs Average	20	4.36	91.05	0	0.000
Variance	-	0.0000	0.0001	0.2000	0.0000
21	20	4.30	88.47	72	0.000
21	20	4.23	86.25	103	0.000
21	20	4.26	87.25	90	0.000
21	20	4.28	87.72	85	0.000
21	20	4.28	87.94	81	0.000
21 Pairs Average	20	4.27	87.53	86	0.000
Variance	-	0.0007	0.7011	131.7000	0.0000
22	20	2.05	35.38	295	3.858
22	20	1.98	32.77	291	3.858
22	20	1.89	30.00	285	3.891
22	20	2.05	35.44	286	3.903
22	20	1.82	28.27	281	3.915
22 Pairs Average	20	1.96	32.37	288	3.885
Variance	-	0.0103	10.2680	29.8000	0.0007
23	20	1.59	21.09	275	8.049
23	20	1.56	19.52	278	8.035
23	20	1.59	20.79	284	7.968
23	20	1.49	18.26	298	8.032
23	20	1.59	21.19	271	8.014
23 Pairs Average	20	1.56	20.17	281	8.020
Variance	-	0.0019	1.5854	110.7000	0.0010

OpenVPN TLS with Triple-DES cipher

Table 28: IxChariot achieved results for the codec G.711a in an OpenVPN TLS with Triple-DES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1 Pair Average	20	4.36	91.05	0	0.000
Variance	-	0.0000	0.0000	0.0000	0.0000
19	20	4.36	91.05	0	0.000
19	20	4.36	91.04	1	0.000
19	20	4.36	90.90	3	0.000
19	20	4.36	90.95	5	0.000
19	20	4.36	91.02	1	0.000
19 Pairs Average	20	4.36	90.92	2	0.000
Variance	-	0.0000	0.0042	4.0000	0.0000
21	20	4.36	91.03	1	0.000
21	20	4.36	91.01	2	0.000
21	20	4.36	91.01	2	0.000
21	20	4.36	91.02	1	0.000
21	20	4.36	91.03	1	0.000
21 Pairs Average	20	4.36	91.02	1	0.000
Variance	-	0.0000	0.0001	0.3000	0.0000
22	20	3.47	68.96	181	0.473
22	20	3.47	68.86	186	0.503
22	20	3.48	69.32	173	0.491
22	20	3.57	71.30	169	0.398
22	20	3.45	68.83	177	0.461
22 Pairs Average	20	3.49	69.45	177	0.465
Variance	-	0.0022	1.1030	44.2000	0.0017
23	20	1.87	30.61	272	4.696
23	20	1.92	32.14	269	4.716
23	20	1.91	31.99	288	4.722
23	20	1.86	30.59	292	4.612
23	20	1.85	30.55	266	4.716
23 Pairs Average	20	1.88	31.18	277	4.692
Variance	-	0.0010	0.6619	138.8000	0.0021

Openswan IPsec plus OpenVPN TLS with AES ciphers and LZO compression/decompression secure connection

Table 29: IxChariot achieved results for the codec G.711a in an Openswan IPsec plus OpenVPN TLS with AES ciphers and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.02	1	0.000
1	20	4.36	90.97	4	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1 Pair Average	20	4.36	91.03	1	0.000
Variance	-	0.0000	0.0012	3.0000	0.0000
19	20	4.36	90.85	9	0.000
19	20	4.36	90.81	11	0.000
19	20	4.36	90.83	10	0.000
19	20	4.36	90.67	18	0.000
19	20	4.36	90.83	10	0.000
19 Pairs Average	20	4.36	90.80	12	0.000
Variance	-	0.0000	0.0053	13.3000	0.0000
21	20	2.19	38.16	320	3.337
21	20	1.74	25.89	313	3.429
21	20	2.13	37.06	317	3.416
21	20	1.73	26.22	317	3.429
21	20	1.79	27.82	304	3.368
21 Pairs Average	20	1.92	31.03	314	3.396
Variance	-	0.0506	36.7644	38.7000	0.0017
22	20	1.60	21.87	304	7.552
22	20	1.74	26.88	315	7.448
22	20	1.60	22.00	325	7.548
22	20	1.45	17.09	341	7.791
22	20	1.43	14.03	297	7.509
22 Pairs Average	20	1.56	20.37	316	7.570
Variance	-	0.0161	24.5602	302.8000	0.0171
23	20	1.30	13.06	341	11.684
23	20	1.32	12.69	337	11.667
23	20	1.34	14.45	319	11.716
23	20	1.20	8.88	324	11.420
23	20	1.28	11.23	333	11.374
23 Pairs Average	20	1.29	12.02	331	11.572
Variance	-	0.0029	4.4776	83.2000	0.0262

Openswan IPsec plus OpenVPN TLS with Triple-DES ciphers and LZO compression/decompression secure connection

Table 30: IxChariot achieved results for the codec G.711a in an Openswan IPsec plus OpenVPN TLS with Triple-DES ciphers and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.36	91.05	0	0.000
1	20	4.36	91.04	0	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1	20	4.36	91.05	0	0.000
1 Pair Average	20	4.36	91.05	0	0.000
Variance	-	0.0000	0.0000	0.0000	0.000
19	20	4.36	90.93	5	0.000
19	20	4.36	90.87	8	0.000
19	20	4.36	90.92	6	0.000
19	20	4.36	91.01	2	0.000
19	20	4.36	90.94	5	0.000
19 Pairs Average	20	4.36	90.93	5	0.000
Variance	-	0.0000	0.0025	4.7000	0.0000
21	20	4.31	89.06	61	0.000
21	20	4.31	88.85	66	0.000
21	20	4.29	88.14	79	0.000
21	20	4.23	86.39	98	0.000
21	20	4.29	88.18	74	0.000
21 Pairs Average	20	4.29	88.12	76	0.000
Variance	-	0.0011	1.1033	205.3000	0.0000
22	20	2.04	34.93	252	3.821
22	20	2.11	36.46	261	3.736
22	20	2.21	39.28	270	3.648
22	20	2.24	39.88	275	3.597
22	20	2.34	41.98	249	3.688
22 Pairs Average	20	2.19	38.51	261	3.698
Variance	-	0.0136	7.8824	125.3000	0.0073
23	20	1.86	29.08	255	7.835
23	20	1.57	19.35	273	7.997
23	20	1.87	27.90	280	7.846
23	20	1.80	26.98	288	7.716
23	20	1.75	25.99	250	7.855
23 Pairs Average	20	1.77	25.86	269	7.850
Variance	-	0.0149	14.5453	263.7000	0.0100

C.3. Codec G.729

Unsecure connection

Table 31: IxChariot achieved results for the codec G.729 in an unsecure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.01	79.69	3	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.72	2	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.68	4	0.000
1 Pair Average	20	4.01	79.72	2	0.000
Variance	-	0.0000	0.0011	3.2000	0.0000
38	20	4.01	79.70	3	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.71	2	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38 Pairs Average	20	4.01	79.73	1	0.000
Variance	-	0.0000	0.0006	2.0000	0.0000
48	20	4.01	79.75	0	0.000
48	20	4.01	79.68	3	0.000
48	20	4.01	79.70	3	0.000
48	20	4.01	79.60	7	0.000
48	20	4.01	79.68	3	0.000
48 Pairs Average	20	4.01	79.68	3	0.000
Variance	-	0.0000	0.0029	6.2000	0.0000
49	20	4.01	79.74	0	0.000
49	20	4.01	79.67	4	0.000
49	20	4.01	79.74	0	0.000
49	20	4.01	79.66	4	0.000
49	20	4.01	79.68	4	0.000
49 Pairs Average	20	4.01	79.70	2	0.000
Variance	-	0.0000	0.0015	4.8000	0.0000
50	20	4.01	79.66	5	0.000
50	20	4.01	79.70	3	0.000
50	20	4.01	79.71	2	0.000
50	20	4.01	79.67	4	0.000
50	20	4.01	79.68	4	0.000
50 Pairs Average	20	4.01	79.68	4	0.000
Variance	-	0.0000	0.0004	1.3000	0.0000

Openswan IPsec with AES cipher secure connection

Table 32: IxChariot achieved results for the codec G.729 in an Openswan IPsec with AES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.01	79.71	2	0.000
1	20	4.01	79.74	1	0.000
1	20	4.01	79.72	2	0.000
1	20	4.01	79.71	2	0.000
1	20	4.01	79.70	3	0.000
1 Pair Average	20	4.01	79.72	2	0.000
Variance	-	0.0000	0.0003	0.5000	0.0000
38	20	2.49	48.25	413	0.000
38	20	2.50	48.33	414	0.000
38	20	2.50	48.40	411	0.000
38	20	2.51	48.52	409	0.000
38	20	2.50	48.43	413	0.000
38 Pairs Average	20	2.50	48.39	412	0.000
Variance	-	0.0001	0.0104	4.0000	0.0000
48	20	1.30	20.39	3947	0.000
48	20	1.30	20.39	3947	0.000
48	20	1.30	20.39	3947	0.000
48	20	1.30	20.42	3944	0.000
48	20	1.30	20.39	3947	0.000
48 Pairs Average	20	1.30	20.40	3946	0.000
Variance	-	0.0000	0.0002	1.8000	0.0000
49	20	1.27	19.71	4305	0.000
49	20	1.27	19.70	4305	0.000
49	20	1.27	19.71	4307	0.000
49	20	1.27	19.73	4303	0.000
49	20	1.28	19.73	4313	0.000
49 Pairs Average	20	1.27	19.72	4307	0.000
Variance	-	0.0000	0.0002	14.8000	0.0000
50	20	1.25	19.08	4664	0.000
50	20	1.26	19.12	4657	0.000
50	20	1.25	19.08	4665	0.000
50	20	1.26	19.13	4663	0.000
50	20	1.25	19.09	4662	0.000
50 Pairs Average	20	1.25	19.10	4662	0.000
Variance	-	0.0000	0.0006	9.7000	0.0000

Openswan IPsec with Triple-DES cipher secure connection

Table 33: IxChariot achieved results for the codec G.729 in an Openswan IPsec with Triple-DES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.01	79.72	2	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.70	3	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.75	0	0.000
1 Pair Average	20	4.01	79.73	1	0.000
Variance	-	0.0000	0.0005	2.0000	0.0000
38	20	4.01	79.59	8	0.000
38	20	4.01	79.65	5	0.000
38	20	4.01	79.60	8	0.000
38	20	4.01	79.67	4	0.000
38	20	4.01	79.60	8	0.000
38 Pairs Average	20	4.01	79.62	7	0.000
Variance	-	0.0000	0.0013	3.8000	0.0000
48	20	1.39	22.86	2914	0.000
48	20	1.38	22.81	2917	0.000
48	20	1.38	22.81	2916	0.000
48	20	1.40	23.11	2896	0.000
48	20	1.40	23.13	2888	0.000
48 Pairs Average	20	1.39	22.94	2906	0.000
Variance	-	0.0001	0.0263	177.2000	0.0000
49	20	1.36	22.14	3228	0.000
49	20	1.36	22.20	3229	0.000
49	20	1.36	22.14	3235	0.000
49	20	1.36	22.10	3231	0.000
49	20	1.36	22.09	3234	0.000
49 Pairs Average	20	1.36	22.13	3231	0.000
Variance	-	0.0000	0.0019	9.3000	0.0000
50	20	1.32	21.13	3595	0.000
50	20	1.33	21.19	3588	0.000
50	20	1.32	21.13	3596	0.000
50	20	1.32	21.17	3590	0.000
50	20	1.33	21.19	3588	0.000
50 Pairs Average	20	1.32	21.16	3591	0.000
Variance	-	0.0000	0.0009	14.8000	0.0000

OpenVPN TLS with AES cipher and LZO compression/decompression secure connection

Table 34: IxChariot achieved results for the codec G.729 in an OpenVPN TLS with AES cipher and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.00	79.37	5	0.000
1	20	4.01	79.69	3	0.000
1	20	4.01	79.71	2	0.000
1	20	4.01	79.70	3	0.000
1	20	4.01	79.71	2	0.000
1 Pair Average	20	4.01	79.64	3	0.000
Variance	-	0.0000	0.2222	1.5000	0.0000
38	20	4.01	79.67	4	0.000
38	20	4.01	79.73	1	0.000
38	20	4.01	79.66	4	0.000
38	20	4.01	79.71	2	0.000
38	20	4.01	79.66	4	0.000
38 Pairs Average	20	4.01	79.69	3	0.000
Variance	-	0.0000	0.0010	2.0000	0.0000
48	20	3.62	70.78	129	0.000
48	20	3.59	70.13	135	0.000
48	20	3.59	70.09	135	0.000
48	20	3.62	70.90	128	0.000
48	20	3.62	70.83	128	0.000
48 Pairs Average	20	3.61	70.55	131	0.000
Variance	-	0.0003	0.1604	13.5000	0.0000
49	20	2.95	57.42	188	1.781
49	20	2.85	55.57	204	1.679
49	20	2.80	54.48	212	1.785
49	20	2.68	52.08	231	1.835
49	20	2.96	57.62	178	1.837
49 Pairs Average	20	2.85	55.43	203	1.783
Variance	-	0.0134	5.2252	428.8000	0.0041
50	20	2.52	48.02	236	3.587
50	20	2.51	48.04	247	3.147
50	20	2.43	46.41	245	3.744
50	20	2.39	45.38	259	3.716
50	20	2.50	48.04	226	3.615
50 Pairs Average	20	2.47	47.18	243	3.562
Variance	-	0.0032	1.5044	153.3000	0.0581

OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection

Table 35: IxChariot achieved results for the codec G.729 in an OpenVPN TLS with Triple-DES cipher and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.01	79.75	0	0.000
1	20	3.83	76.06	38	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.75	0	0.000
1 Pair Average	20	3.97	79.01	8	0.000
Variance	-	0.0065	2.7232	288.8000	0.0000
38	20	4.01	79.74	1	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38 Pairs Average	20	4.01	79.75	0	0.000
Variance	-	0.0000	0.0000	0.2000	0.0000
48	20	4.00	79.56	1	0.043
48	20	4.01	79.74	1	0.000
48	20	4.01	79.73	1	0.000
48	20	4.01	79.69	3	0.000
48	20	4.01	79.73	1	0.000
48 Pairs Average	20	4.01	79.69	1	0.009
Variance	-	0.0000	0.0056	0.8000	0.0004
49	20	4.01	79.68	4	0.000
49	20	4.01	79.57	9	0.000
49	20	4.01	79.75	0	0.000
49	20	4.01	79.74	0	0.000
49	20	4.01	79.73	1	0.000
49 Pairs Average	20	4.01	79.69	3	0.000
Variance	-	0.0000	0.0055	14.7000	0.0000
50	20	4.01	79.71	2	0.000
50	20	4.01	79.56	10	0.000
50	20	4.01	79.59	8	0.000
50	20	4.01	79.55	10	0.000
50	20	4.01	79.73	1	0.000
50 Pairs Average	20	4.01	79.63	6	0.000
Variance	-	0.0000	0.0073	19.2000	0.0000

OpenVPN TLS with AES cipher secure connection

Table 36: IxChariot achieved results for the codec G.729 in an OpenVPN TLS with AES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.01	79.75	0	0.000
1	20	4.01	79.75	0	0.000
1	20	4.00	79.45	1	0.000
1	20	4.01	79.68	3	0.000
1	20	4.01	79.75	0	0.000
1 Pair Average	20	4.01	79.68	1	0.000
Variance	-	0.0000	0.0169	1.7000	0.0000
38	20	4.01	79.69	3	0.000
38	20	4.01	79.68	3	0.000
38	20	4.01	79.65	5	0.000
38	20	4.01	79.64	6	0.000
38	20	4.01	79.57	9	0.000
38 Pairs Average	20	4.01	79.65	5	0.000
Variance	-	0.0000	0.0022	6.2000	0.0000
48	20	3.59	70.30	132	0.000
48	20	3.57	69.69	139	0.000
48	20	3.60	70.48	131	0.000
48	20	3.58	70.01	135	0.000
48	20	3.60	70.35	132	0.000
48 Pairs Average	20	3.59	70.17	134	0.000
Variance	-	0.0002	0.1003	10.7000	0.0000
49	20	2.82	54.92	207	1.788
49	20	2.77	53.77	215	1.910
49	20	2.78	53.98	221	1.793
49	20	2.74	53.16	232	1.622
49	20	2.81	54.69	212	1.773
49 Pairs Average	20	2.78	54.10	217	1.777
Variance	-	0.0010	0.5068	92.3000	0.0105
50	20	2.77	53.53	172	3.748
50	20	2.79	53.86	173	3.705
50	20	2.68	51.63	192	3.731
50	20	2.56	49.25	214	3.775
50	20	2.79	53.86	174	3.715
50 Pairs Average	20	2.72	52.43	185	3.735
Variance	-	0.0099	4.0130	331.0000	0.0008

OpenVPN TLS with Triple-DES cipher

Table 37: IxChariot achieved results for the codec G.729 in an OpenVPN TLS with Triple-DES cipher secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.01	79.75	0	0.000
1	20	4.00	79.47	0	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.75	0	0.000
1	20	4.00	79.47	0	0.000
1 Pair Average	20	4.01	79.64	0	0.000
Variance	-	0.0000	0.0235	0.0000	0.0000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.70	3	0.000
38	20	4.01	79.61	7	0.000
38	20	4.01	79.74	1	0.000
38 Pairs Average	20	4.01	79.71	2	0.000
Variance	-	0.0000	0.0035	8.7000	0.0000
48	20	4.01	79.74	1	0.000
48	20	4.01	79.73	1	0.000
48	20	4.01	79.73	1	0.000
48	20	4.01	79.70	3	0.000
48	20	4.01	79.68	4	0.000
48 Pairs Average	20	4.01	79.72	2	0.000
Variance	-	0.0000	0.0006	2.0000	0.0000
49	20	4.01	79.73	1	0.000
49	20	4.01	79.69	3	0.000
49	20	4.01	79.73	1	0.000
49	20	4.01	79.69	3	0.000
49	20	4.01	79.73	1	0.000
49 Pairs Average	20	4.01	79.71	2	0.000
Variance	-	0.0000	0.0005	1.2000	0.0000
50	20	4.01	79.72	1	0.000
50	20	4.01	79.72	2	0.000
50	20	4.01	79.72	2	0.000
50	20	4.01	79.57	9	0.000
50	20	4.01	79.72	1	0.000
Variance	-	0.0000	0.0045	11.5000	0.0000
50 Pairs Average	20	4.01	79.69	3	0.000

Openswan IPsec plus OpenVPN TLS with AES ciphers and LZO compression/decompression secure connection

Table 38: IxChariot achieved results for the codec G.729 in an Openswan IPsec plus OpenVPN TLS with AES ciphers and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.01	79.71	2	0.000
1	20	4.01	79.69	3	0.000
1	20	4.01	79.67	4	0.000
1	20	4.01	79.70	2	0.000
1	20	4.01	79.70	3	0.000
1 Pair Average	20	4.01	79.69	3	0.000
Variance	-	0.0000	0.0002	0.7000	0.0000
38	20	4.01	79.68	4	0.000
38	20	4.01	79.66	4	0.000
38	20	4.01	79.70	2	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.67	4	0.000
38 Pairs Average	20	4.01	79.69	3	0.000
Variance	-	0.0000	0.0013	3.2000	0.0000
48	20	2.42	46.01	255	3.690
48	20	2.30	43.90	265	3.719
48	20	2.21	42.49	269	3.851
48	20	2.47	46.88	277	3.066
48	20	2.44	46.60	246	3.650
48 Pairs Average	20	2.37	45.18	262	3.595
Variance	-	0.0120	3.6174	146.8000	0.0932
49	20	2.28	42.72	236	5.816
49	20	2.09	39.92	245	5.668
49	20	2.30	42.49	258	5.475
49	20	2.13	40.46	240	5.831
49	20	2.15	41.07	238	5.627
49 Pairs Average	20	2.19	41.33	243	5.683
Variance	-	0.0088	1.5226	77.8000	0.0216
50	20	2.02	36.66	265	7.475
50	20	2.06	37.22	269	7.408
50	20	1.89	33.91	285	7.609
50	20	1.93	34.25	293	7.591
50	20	2.10	37.71	272	7.171
50 Pairs Average	20	2.00	35.95	277	7.451
Variance	-	0.0078	3.0666	138.2000	0.0313

Openswan IPsec plus OpenVPN TLS with Triple-DES ciphers and LZO compression/decompression secure connection

Table 39: IxChariot achieved results for the codec G.729 in an Openswan IPsec plus OpenVPN TLS with Triple-DES ciphers and LZO compression/decompression secure connection.

# of VoIP Pairs	Timing [ms]	MOS average	R-value average	One-way delay average [ms]	% Bytes Lost E1-E2
1	20	4.01	79.75	0	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.75	0	0.000
1	20	4.01	79.73	1	0.000
1	20	4.01	79.75	0	0.000
1 Pair Average	20	4.01	79.75	0	0.000
Variance	-	0.0000	0.0000	0.2000	0.0000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38	20	4.01	79.75	0	0.000
38 Pairs Average	20	4.01	79.75	0	0.000
Variance	-	0.0000	0.0000	0.0000	0.0000
48	20	4.01	79.57	9	0.000
48	20	4.01	79.62	6	0.000
48	20	4.01	79.57	9	0.000
48	20	4.01	79.64	6	0.000
48	20	4.01	79.55	10	0.000
48 Pairs Average	20	4.01	79.59	8	0.000
Variance	-	0.0000	0.0015	3.5000	0.0000
49	20	4.01	79.65	5	0.000
49	20	4.01	79.69	3	0.000
49	20	4.01	79.59	8	0.000
49	20	4.01	79.54	11	0.000
49	20	4.01	79.63	6	0.000
49 Pairs Average	20	4.01	79.62	7	0.000
Variance	-	0.0000	0.0033	9.3000	0.0000
50	20	4.01	79.66	5	0.000
50	20	4.01	79.59	8	0.000
50	20	4.01	79.59	8	0.000
50	20	4.01	79.55	10	0.000
50	20	4.01	79.74	0	0.000
50 Pairs Average	20	4.01	79.66	6	0.000
Variance	-	0.0000	0.0056	15.2000	0.0000