Securing media streams in an Asterisk-based environment and evaluating the resulting performance cost

Submitted in fulfilment
of the requirements of the degree
of Master of Science
of Rhodes University

Bradley Clayton

8th January 2007

Abstract

When adding Confidentiality, Integrity and Availability (CIA) to a multi-user VoIP (Voice over IP) system, performance and quality are at risk. The aim of this study is twofold. Firstly, it describes current methods suitable to secure voice streams within a VoIP system and make them available in an Asterisk-based VoIP environment. (Asterisk is a well established, open-source, TDM/VoIP PBX.) Secondly, this study evaluates the performance cost incurred after implementing each security method within the Asterisk-based system, using a special testbed suite, named DRAPA, which was developed expressly for this study.

The three security methods implemented and studied were IPSec (Internet Protocol Security), SRTP (Secure Real-time Transport Protocol), and SIAX2 (Secure Inter-Asterisk eXchange 2 protocol). From the experiments, it was found that bandwidth and CPU usage were significantly affected by the addition of CIA. In ranking the three security methods in terms of these two resources, it was found that SRTP incurs the least bandwidth overhead, followed by SIAX2 and then IPSec. Where CPU utilisation is concerned, it was found that SIAX2 incurs the least overhead, followed by IPSec, and then SRTP.

Acknowledgements

My supervisors, Alfredo Terzoli and Barry Irwin have been invaluable. Their leadership, inspiration and raw passion for discovering new stuff has kept me focussed and enthusiastic even at my lowest moments. It is a credit to their dedicated guidance that there have been so many highs and remarkably few lows. I'm also grateful to Hannah Slay, for reading my work and giving my valuable input.

Dad, thank you for inspiring me and for nurturing my passion for technology. I especially appreciate your guidance and critical comments on my work from its early conceptual stages to the write-up. Mom, thank you for your love, support and belief in me. The world would have suffered a grump had you not reminded me to eat and sleep. My little brother Luke, thank you for threatening to beat me with a stick near the completion date. Your inimitable sense of humour kept me sane.

Thank you to my friends who make my world an awesome playground.

Kiran, you walked the same path hand-in-hand and you motivated me when I was in doubt. I especially appreciate your critical reading of my work, over and over again. Thank you also for stimulating me with your creative intelligence and for singing to me every day.

This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Comverse, Verso Technologies, Tellabs, StorTech, THRIP and the National Research Foundation and the German Academic Exchange Service (DAAD).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The problem

The growing speed of networks and the large scale of the Internet has enabled the development of real-time services over IP networks. One such real-time service is VoIP (Voice over Internet Protocol), where voice and video communication, which have traditionally run on the PSTN (Public Switched Telephone Network), are transported on IP networks. VoIP technologies are being adopted in the home and business environments as they provide cheaper call rates and greater flexibility. For example, a business is able to use their IP connections to link the telephone systems of physically separated offices, reducing call charges and so allowing better communications among the offices. At the same time, flexibility is attained through the ease by which services can be developed and integrated into existing data systems. For example, a system in a customer support centre could be implemented so that an incoming call triggers the caller's profile to be automatically opened on the support attendant's computer. However, despite these advantages, running voice services on IP networks brings about particular challenges, such as quality of service and security. This study focuses on the performance cost of adding security to a multi-user VoIP system. Quality of Service (QoS) for voice services is introduced next, followed by security for voice services.

1.1.1 Quality of voice services

Unlike VoIP, services built on the PSTN are able to guarantee a high level of quality. The QoS provided by the PSTN is achieved through the design of the network. The PSTN utilises *circuit-switched* technology in which a physical or, at least, a logical circuit is dedicated to each active session.

By contrast, IP networks are *packet-switched*, representing a different design that typically provides a *best-effort* level of service. A packet-switched network is shared by multiple users by dividing transported data into packets [33]. Quality of service can be achieved within an IP network but it requires additional logic (for example, a throttling mechanism built into the operating system), unlike a circuit-switched network where the QoS is inherent. However, packet-switched networks make more efficient use of available bandwidth. The performance of a VoIP service is related to the complexity of the system and influenced by other applications running on the same network: any addition to a VoIP system can potentially incur a performance cost to the overall system.

This study focuses on the performance cost of adding security to VoIP systems. Security for VoIP is introduced next.

1.1.2 Security of voice services

We are interested in three areas from the larger field of data security: confidentiality, integrity and availability (CIA) [25] and hence we do not discuss the protection of the VoIP infrastructure from malicious attacks (the reader is referred to a related study for details in this regard [52]). Eavesdropping is a breach of confidentiality, and integrity guards against the malicious alteration of communication by an external party. Availability guards against the denial of a service. Traditional voice and data services reside typically on independent infrastructure and are subject to different types of security vulnerabilities. VoIP inherits the security risks of traditional networks as well as the security risks found in data networks. This makes the development of adequate security for voice services particularly important and interesting. For example, the physical security and centralised control of the PSTN reduces the forging of identity. Identity in an IP network can be forged by simply falsifying the sender address on an IP packet. Furthermore, spoofed IP packets can be injected into any part of an IP network, whereas physical access to a telephone exchange is needed if such an attack is to be performed using the PSTN.

1.1.3 Specific problem statement

Methods exist for providing CIA for data transported on IP networks and these methods can be employed for voice services. However, early security methods were designed for securing data transport and not so much with real-time communication in mind. Therefore, many current security methods are appropriate for small scale communication systems but for large scale systems they might incur a considerable overhead, which could make them unusable.[1] With the adoption of VoIP for large

[1] A large scale VoIP system would be one which serves many clients concurrently, utilising shared servers and bandwidth, such as a VoIP-based call-centre.

scale systems, a need has emerged for new security mechanisms which are specifically designed for real-time applications.

This study focuses on the performance impact of various security methods, both generic and specific to VoIP, within a set VoIP environment. This environment is described next.

1.1.4 The VoIP environment

The VoIP environment in which this study is conducted is the iLanga telephone system, developed at the Rhodes University Computer Science department [36]. The architecture of this system comprises servers running Asterisk (an open source software PBX [53]), SER (the SIP[2] Express Router) and an innovative user interface [37]. The Asterisk platform provides three distinct advantages. Firstly, it supports TDM[3] interfaces as well as SIP, H.323,[4] IAX[5]and MGCP.[6] This study concentrates on SIP and IAX as these are interesting emerging protocols. Secondly, Asterisk is a software implementation of a PBX running on commodity PC hardware (specific TDM hardware is only required when Asterisk is connected to TDM networks). Finally, the Asterisk source code is licensed as open source, providing an open, flexible platform for this study [69].

iLanga is a multi-user system, where centralised hosts serve multiple communication sessions simultaneously. The centralised hosts act as gateways to the local PSTN and the university's internal telephone system. Other than the basic authentication measures for registering an IP device with iLanga, there is no security in place to provide confidentiality, integrity and continued availability for an active session. In response to the current lack of security in the iLanga system, this study investigates suitable security methods and their performance impact when added to an Asterisk-based system such as iLanga.

The specific research objectives are outlined in the following section.

1.2 Research Objectives

This study has two primary objectives:

[2]Session Initiation Protocol
[3]Time Division Multiplexing
[4]H.323 is an ITU standard for video-conferencing over packet-switched networks
[5]Inter Asterisk eXchange
[6]Media Gateway Control Protocol

1. To investigate appropriate methods of securing conversation streams in an Asterisk-based VoIP system, and implement them as a proof-of-concept system.

2. To examine the performance cost, in terms of CPU, bandwidth and overall quality, of the selected security additions to an Asterisk-based VoIP system. This was achieved through the use of a specially designed software suite that we called DRAPA.

The first objective seeks to identify appropriate methods of securing our VoIP environment. Appropriacy is measured in terms of three factors: a security method's provision of CIA, its ease of implementation into our environment, and the performance cost to real-time applications. Drawing from the literature, security methods are evaluated against the three pillars of security represented in the CIA model, evaluating their provision of confidentiality, integrity and availability [40, 25]. Adequate CIA depends on initial authentication, and so authentication is also outlined during the investigation. The appropriate security methods should be implemented within an Asterisk-based system as a proof-of-concept. The second objective undertakes to find a method of performance analysis. Part of this objective is the exploration of areas which affect the quality of VoIP and measures which can be used to rank the performance of security mechanisms added to a VoIP system. From this exploration a testbed can be built and used to investigate the overhead incurred as a result of adding the security methods to an Asterisk-based VoIP system.

1.3 Project scope

The environment in which the study is undertaken is iLanga, described in Section 1.1.4. The area of this research is specifically the IP interface of Asterisk. Of the VoIP protocols made available by Asterisk, only two are used in this study, namely the Session Initiation Protocol (SIP) [62] and the Inter-Asterisk eXchange (IAX2) protocol [70]. The SIP protocol merely creates a session and works in collaboration with other protocols, such as SDP (Session Description Protocol) [35] and RTP (Real-time Transport Protocol) [66] to negotiate session attributes and to transport the session media. These subsidiary protocols are the focus of the study.

This study does not attempt to provide a full security solution for VoIP. The securing of voice streams during a VoIP session is the primary area of investigation. The study will not include the securing of signalling and of the infrastructure supporting the VoIP streams.

The transportation of the media streams is the part of VoIP systems that is resource intensive. Securing the streams will ultimately result in performance costs. The investigation of these costs is within the scope of this study.

1.4 Thesis Outline

The thesis is structured as follows:

- Chapter 2 provides an overview of work related to this research. It begins by describing the real-time multimedia protocols used in this study in Section 2.2. For each protocol, basic security considerations are discussed. The chapter then introduces and describes three security methods for our VoIP environment in Section 2.3. Security mechanisms rely on cryptographic keys to perform confidentiality and integrity services. Methods for exchanging these keys before a communication session is established are explained in Section 2.4. In Section 2.5, the performance constraints on VoIP systems are introduced. The chapter concludes by discussing the metrics used for measuring VoIP quality, as well as the existing tools used to measure the performance of VoIP components.

- Chapter 3 describes the design and implementation of a testbed named DRAPA (Distributed Real-time Application Performance Analyser). The architecture of the testbed is described in terms of the entities which constitute DRAPA. Unlike a simulated testbed, DRAPA relies on a real implementation of a VoIP system. The testbed is managed by distributed agents. These agents are explained in detail in Section 3.4. Next, the central control, which is the core of DRAPA, is described in Section 3.5. This is followed by a description of user configurable modules, which provide flexibility to a user of DRAPA in Section 3.6. Section 3.7 describes the web interface which provides status information to the user. An initial evaluation of DRAPA is performed to ensure the performance results generated are comparable to existing studies. The evaluation is described in Section 3.8. The chapter concludes by identifying and addressing problems with the testbed in Section 3.9.

- Chapter 4 can be divided into three parts. The first part, Section 4.2, describes the configuration and implementation of three security mechanisms within an Asterisk-based VoIP system. The second part, Section 4.4 and Section 4.5, describes the use of DRAPA in an experiment to measure the performance cost placed on the secured VoIP system. The configuration of the testbed as well as preliminary theory and expected outcomes are discussed. The third part of the chapter, Section 4.6, presents and examines the results of the experiment. For each performance metric, the security methods are ranked in terms of their performance impact on the system. Finally, a summary is presented which takes into account the results from the experiment, and provides guidelines for best use of each security method.

- Chapter 5 presents the conclusion to this study. The chapter outlines two contributions this study has made to VoIP research in Section 5.2: 1) the addition of security to an Asterisk-based VoIP environment, 2) the performance analysis of VoIP systems. The chapter concludes by

reflecting on three limitations of this study and, in light of these, it suggests avenues for future research.

Through this research, the author has published four papers [18, 20, 19, 21].

Chapter 2

Related work

2.1 Introduction

Having introduced the environment in which this study takes place in the first chapter, a description of the VoIP protocols, security mechanisms and performance constraints for VoIP is presented in this chapter. Section 2.2 describes the protocols found in our environment. Section 2.3 describes security methods appropriate for securing real-time communication. Any security method needs a cryptographic key before it can provide CIA (Confidentiality, Integrity and Availability) services. Methods of key exchange for real-time communication are discussed in Section 2.4. This study focuses on the performance implications when security is added to a VoIP system. Finally, Section 2.5 describes the constraints placed upon real-time communication.

2.2 Real-time multimedia protocols

The life of a VoIP session can be divided into three stages. The first stage is to establish a session between two or more participants. The second stage is to transport session media between the participants. The third stage is to end the session and perform post session tasks such as billing. To facilitate these stages, two types of protocol are used in VoIP system, namely signalling and media transport protocols. An example of a signalling protocol is SIP (Session Initiation Protocol). SIP is responsible for the messaging needed to create a session. RTP on the other hand is responsible for transporting the session media.

The sub-sections which follow introduce and describe the VoIP protocols utilised in this study.

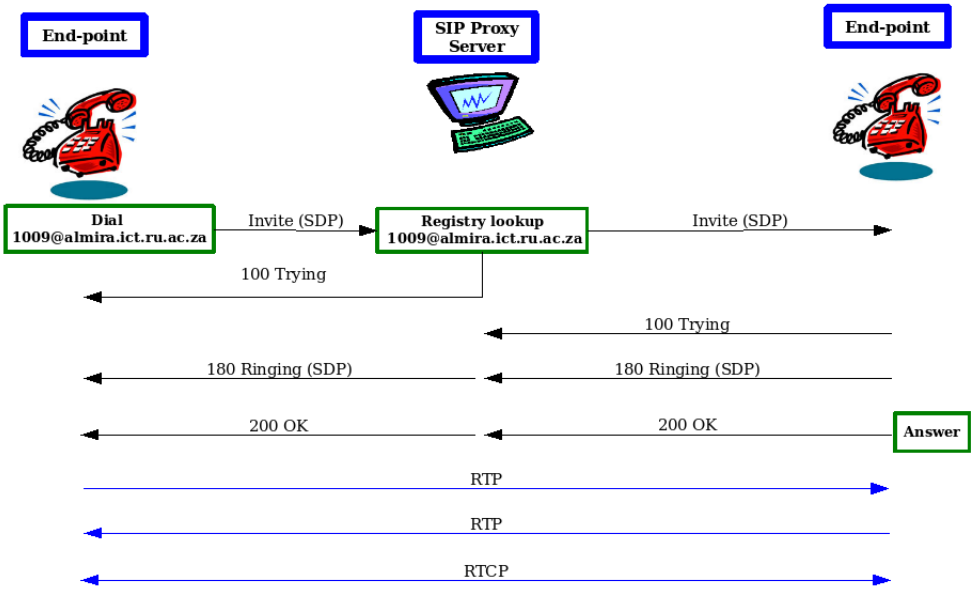


Figure 2.1: Typical SIP signalling when creating a session

2.2.1 Session Initiation Protocol

SIP is a signalling protocol which is responsible for creating media sessions between two or more participants [62]. The hardware or software used by a participant is referred to as an *end-point* in this study. Once a session has been negotiated, it is up to another protocol, for example RTP, to transport the media. In a multi-user environment, SIP relies on the infrastructure of proxy and registrar services to create communication sessions between end-points. The registrar service is a registration service for end-points, which allows multiple end-points to register with a single node so that registry information is centralised. A SIP registrar service maintains a database of end-points which a proxy service uses to facilitate session creation when one end-point dials another. Registrar and proxy services are often run on a single server. Hereafter, a *server* refers to a machine running the SIP registrar and proxy services. End-points are distinguished and addressed by their Uniform Resource Identifier (URI). A SIP URI consists of a user name and domain name, much like an e-mail address. For example, SIP:BRADLEY@SIP.ICT.RU.AC.ZA would identify the user BRADLEY at the domain SIP.ICT.RU.AC.ZA. When an end-point registers with a server, a unique URI-to-IP address mapping is added to the registrar's database. Using the previous example, if the user registered an end-point with the IP address 146.231.117.13, the following URI to IP address mapping would be created: 146.231.117.13:5060:60:BRADLEY@SIP.ICT.RU.AC.ZA.

Figure 2.1 shows the signalling involved in initiating a session between two end-points using a server. The originating end-point sends an INVITE message to the registrar that requests a session be started

with bradley@sip.ict.ru.ac.za. The registrar service searches for the IP address in its database and the proxy service sends an INVITE request to the destination end-point. The SIP INVITE messages encapsulate an SDP payload (described further in Section 2.2.2) which contains session information. This information would include a list of supported audio codecs and a request that RTP (see Section 2.2.3) should be used to transport the audio. The destination end-point replies with a TRYING message, to acknowledge the INVITE from the registrar, followed by a RINGING message when the destination end-point is generating a ringing sound. When the destination end-point is answered, two media streams (described further in Section 2.2.3) are created between the end-points. In this example, the RTP protocol is used to transport the media. The session media is not always transported directly between the end-points. Instead, it could be transported via the server. Transporting the session media via the server is advantageous in situations where the end-points do not support the same audio codec. In this case the server will perform *trans-coding*, whereby it converts one media format into another.

Securing SIP

Before methods for securing SIP are discussed, the difference between symmetric and asymmetric encryption needs to be explained. Symmetric and asymmetric cryptography differ in their key pairs. Symmetric cryptography uses the same key to encrypt and decrypt messages. Asymmetric cryptography uses two different keys to encrypt and decrypt a message. The keys are generated such that a message encrypted with the first key can only be decrypted with the second key. Therefore, one key is made public while the other is kept secret. The public key is used to encrypt a message which can then only be decrypted with the secret key [25].

SIP is an application layer protocol which utilises text messaging in a similar way to the Hyper Text Transfer Protocol (HTTP) [30] and the Simple Mail Transfer protocol (SMTP) [41]. Therefore, we are able to draw on existing security mechanisms to provide confidentiality, integrity and availability for SIP, using techniques such as: [40]

- HTTP digest authentication
- Secure MIME (Multipurpose Internet Mail Extensions)
- Transport Layer Security (TLS)

HTTP Digest authentication simply challenges a remote end by requiring a check-sum containing the following information: the user name, HTTP method, requested URI and a *nonce* value (a random number used to protect against replay or statistical attacks). The password is never sent as clear

text and the nonce value protects against replay attacks. However, HTTP Digest Authentication is susceptible to brute force attacks and is not recommended for securing SIP.

SIP already carries a MIME payload, enabling the use of Secure MIME (S/MIME) to provide confidentiality, authentication and integrity for SIP messages. S/MIME SIP tunnelling is available should one additionally wish to protect the SIP headers. S/MIME uses certificates to perform authentication, and public key cryptography for integrity and confidentiality. The use of certificates allows the third-party trust authority model to be followed [56, 12] which provides strong authentication. However, public key encryption is an asymmetric method of cryptography which incurs a higher performance cost than symmetric cryptography [25]. Since transport reliability is required for S/MIME, TCP (rather than UDP) is recommended as the underlying transport protocol, so that the recovery mechanisms within TCP can be used [27]. This could raise concern in terms of performance as TCP requires more detailed headers than UDP, creating a larger overhead. When VoIP media is being transported, the overhead incurred by transport headers is significant in relation to the size of the media [50].

Integrity and confidentiality of SIP messages can also be protected by the TLS protocol. TLS is referred to as a *hop-by-hop* method of security, because SIP messages are only secure while being transported on the network medium. Should a SIP message, protected by TLS, be passed from one end-point to another via a VoIP server, the server would decode the message and re-encode it with TLS before passing it on to the destination end-point. Like S/MIME, TLS requires a reliable transport protocol and so is only transported by TCP. However, Datagram Transport Layer Security (DTLS) is a new standard which could be used to provide TLS protection for SIP messages on top of the UDP transport protocol [60]. Other security methods, such as IPSec, would provide confidentiality, integrity and availability for SIP. (IPSec is discussed further in section 2.3.1.)

SIP can be used to facilitate the creation of a secure session, in which the session media is protected. A standard exists, RFC 3329, which describes the use of SIP for the negotiation of security mechanisms [6]. The standard is designed to be flexible so that it supports current and future security mechanisms. This standard can improve the performance of creating a secure VoIP session as it allows a security mechanism to be configured within the SIP protocol, rather than before or after a session is created. For example, the cryptography key needed by a security method can be transported within the SIP protocol. (These security methods, for example SRTP, are described further in Section 2.3.2.)

The SDP protocol is used by SIP to transport a list of session attributes. Security attributes can also be transported within SDP. The SDP protocol is described next.

o= (owner/creator and session identifier).
s= (session name)
i= (session information)
u= (URI of description)
m= (media name and transport address)
k= (encryption key)
a= (zero or more media attribute lines)

Figure 2.2: Selected portion of an SDP message [35]

2.2.2 Session Description Protocol

When establishing a real-time multimedia session, each party must agree on a set of attributes for the new session. Session attributes include the type of media to be transported, the transport protocol and the media format. These attributes are exchanged between end-points using SDP [35]. SDP is simply a format for describing the attributes of a session and its scope does not include session creation and media transport. The responsibility for creating a session falls upon another protocol, such as SIP. The transport of media can be performed by RTP, which is discussed in Section 2.2.3.

When SIP is used to initiate a session, the caller populates the attributes within an SDP description. This description is then encapsulated within the SIP INVITE message. Attributes are specified in *lines,* as seen in figure 2.2. Each line begins with an identifier followed by an equals sign and a value. The attributes in the caller's SDP message include, for example, a list of supported audio codecs. The callee selects preferred attributes from the caller's SDP message and uses them to populate a second SDP message. For example, one of the audio codecs from the caller's list will be selected. The callee passes the second SDP message to the caller using a SIP OK message. The codec format and transport type are listed using *a* lines as seen in figure 2.2.

The SDP standard has optional lines to describe attributes needed for media protection. In figure 2.2, the *k* line can be used to specify an encryption key which enables SDP to be used as a key exchange protocol. SDP is not inherently secure and relies on other protocols, such as S/MIME, to protect sensitive session attributes. A draft standard exists which describes the use of DTLS (Datagram TLS) for SDP security [31]. This standard would allow one to secure SDP messages with TLS on top of the UDP transport protocol. After a session is created between multiple end-points, a method for transporting session media is needed.

The next section describes the RTP protocol which is responsible for transporting session media.

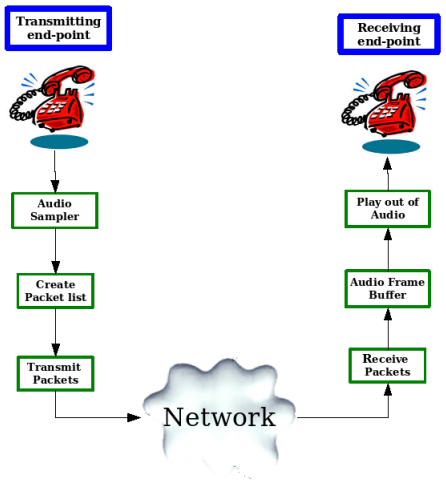


Figure 2.3: Lifetime of an RTP payload [22]

2.2.3 Real-time Transport Protocol

RTP is a protocol which facilitates the transport of data over a network in real-time applications [66]. RTP is intended to be used for applications such as audio and video conferencing, real-time systems control, and unicast or multicast services. After creating a session, the RTP protocol can be used to transport the session media (see figure 2.1 in section 2.2.1). Within the RTP protocol there is a second protocol, the RTP Control Protocol (RTCP). RTCP is responsible for providing feedback information on the quality of an RTP stream to the source of the stream. RTCP information is periodically transmitted through the same mechanism as the RTP stream. Figure 2.3 is an example of how RTP could be implemented in a one-way communication session. Audio is sampled in *frames*, each of which contains a portion of audio, usually 20 to 40 milliseconds in length. The frames are wrapped in RTP packets with headers indicating the audio format and order in which the audio should be played back. The RTP packets are transmitted over the network to the receiver. The receiver unwraps each frame and re-assembles the audio, using the order indicator in the RTP header, in a *playout* buffer. The audio is then played from the playout buffer. RTP utilises the unreliable IP transport protocol, UDP. In real-time applications, reliability is sacrificed in favour of real-time delivery. (If an RTP packet transporting audio is lost, it would be pointless to re-transmit the packet since its position in the playout buffer would get played before retransmission can take place.)

RTP can be secured with transparent mechanisms such as IPSec. Alternatively, it can be secured with the Secure Real-time Transport Protocol (SRTP). Both of these security mechanisms have their own advantages and disadvantages and are discussed in Section 2.3. The last VoIP protocol relevant to this study, IAX, is discussed in the next section.

2.2.4 Inter-Asterisk Exchange Protocol

The Inter-Asterisk eXchange version 2 (IAX2) protocol [70, 69] was originally used to *trunk* VoIP traffic between Asterisk soft-switches. Trunking is a process whereby IP packets for multiple sessions are transported under a single IP header. This can be achieved if the source and destination for a group of streams is the same. IAX has since been adopted in client VoIP implementations and is regarded as an alternative to SIP. IAX transports media and signalling through a single stream and, unlike the SIP-SDP-RTP combination, it utilises a single IP port on which to transmit and receive.

From a security point of view, the design of IAX has two advantages over RTP and SIP. Firstly, IAX makes the configuration of a firewall simple because a single IP port can be opened, whereas RTP requires a range of ports to be opened. Secondly, IAX is able to traverse Network Address Translation (NAT) gateways without the need for STUN (Simple Traversal of UDP over NATs) [55] support on the gateway. (NAT can be seen as a simple and effective method of network security, as it only allows traffic to traverse the gateway after a connection has been initiated from within the network.)

As of version 1.2.4 of the Asterisk soft-switch, the IAX2 channel is able to perform authentication and encryption of signalling and media with MD5 hashing and 128 bit AES encryption respectively [74].

Specific details for securing the VoIP protocols introduced in this section are discussed further in the next section.

2.3 Appropriate mechanisms for VoIP security

Securing an IP network in such a way that its level of security matches that of the PSTN could be performed in two ways. An IP network fully, dedicated to the use of real-time services, can be physically protected and disconnected from any other data network. This follows the classical approach of PSTN networks. However, this makes for costly installation, and as such, it reduces the appeal of using VoIP. Virtual LAN technology (VLAN) allows for a logical separation of IP networks within one physical network [55]. This could be a more reasonable option for separating the real-time services from data services. The second way would be to apply current methods of data security to the real-time protocols. Not all current data security methods are suitable for real-time applications though. For example, securing a VoIP telephone conversation with a TCP-based VPN (virtual private network) would introduce a bandwidth overhead and varied latency (see Section 2.5.1) which would degrade the quality of the service. The methods discussed in this chapter were selected in two ways: methods which are suitable for real-time protection, and methods which are easily integrated into our

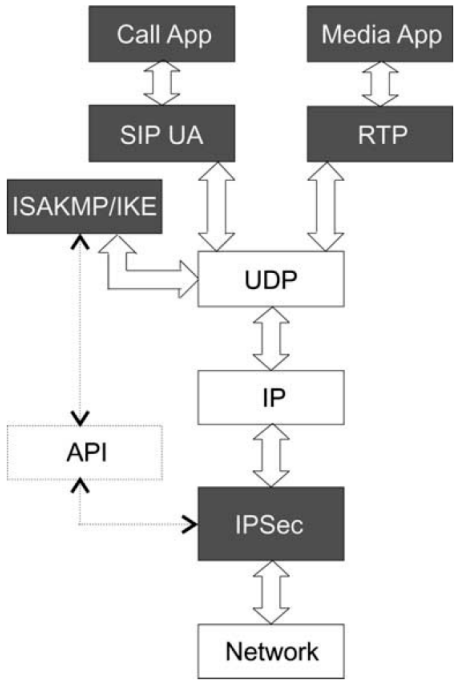


Figure 2.4: A VoIP protocol stack including IPSec [59]

environment. Section 2.3.1 presents IPSec, a transparent solution which resides within the operating system. Section 2.3.2 describes SRTP, which is an adaptation of the audio/video profile for RTP. Lastly, the AES security built into IAX2 is discussed in Section 2.3.3.

2.3.1 IPSec

This section is primarily based on two sources, [25, 55]. IPSec resides below the IP layer of the operating system. From the point of view of an application, data sent over the network is protected transparently, while data received from the network is unprotected transparently. A diagram of a VoIP protocol stack including IPSec is shown in Figure 2.4. IPSec utilises a range of protocols to accomplish confidentiality, integrity, availability, authentication and key exchanges . The first protocol used by IPSec is the Authentication Header (AH). The AH protocol provides integrity and authentication of IP packets (but does not provide confidentiality). This is realised by adding a header, called a *fingerprint* to each IP packet. The fingerprint is generated through the use of MAC (Message Authentication Code) algorithms. MAC algorithms and hashing algorithms are similar: given an original message they produce a fingerprint. The fingerprint is used to check the integrity of the message to ensure that it does not differ from the original. However, unlike a hashing algorithm, MAC algorithms generate the fingerprint from two inputs, namely the original message and a cryptographic key. By

```
spdadd LOCAL-ADDRESS REMOTE-ADDRESS any -P out ipsec
esp/transport/LOCAL-ADDRESS-REMOTE-ADDRESS/require;
```

Figure 2.5: Example of an IPSec policy

keeping the key secret, MAC algorithms enable integrity validation of communicated information. The AH protocol is run in one of two modes, transport or tunnel mode. In transport mode, an authentication header is added to the original IP packet. The authentication header is calculated for all the fields within the packet which do not change while on the wire. In tunnel mode, the original IP packet is encapsulated within a new IP packet and the authentication header is calculated so that it includes the entire original packet. Tunnel mode is suitable when two physically separated networks are linked via a virtual private network. The tunnel is managed by a gateway on each network.

Like AH, the Encapsulating Security Payload (ESP) protocol provides data integrity and authentication but adds confidentiality and anti-replay protection. Authentication is provided through MAC algorithms as with the AH protocol. Confidentiality can be provided by utilising a number of cryptographic ciphers, for example DES, 3DES, AES and Blowfish. ESP is also run in transport or tunnel mode. Static parts of the IP packet are secured in transport mode. Alternatively, the entire IP packet is encapsulated and protected in tunnel mode.

IPSec utilises the Internet Security Association Key Management Protocol (ISAKMP), which is responsible for maintaining security associations (SA). An SA is an agreement between communicating nodes which specifies attributes to use for an IPSec secured session. For example, an SA could state that ESP is to be used in transport mode, using the AES cipher for confidentiality, and the HMAC-SHA1 hashing algorithm for authentication. ISAKMP facilitates the creation of an SA between two communicating nodes. ISAKMP does not provide a method for exchanging keys during the creation of an SA. Instead, ISAKMP provides a generic framework which can be utilised by any key exchange protocol. The Internet Key Exchange (IKE) protocol does provide a method of key exchange, and uses the ISAKMP framework to create SAs for IPSec sessions. The keys exchanged are then used by AH and ESP for their authentication, integrity and confidentiality services. Key exchange protocols are discussed further in Section 2.4.

IPSec is configured by adding policies to the security policy database (SPD). A policy is simply a rule which governs which network channels are secured and what kind of security is utilised. Figure 2.5 is an example of a security policy. The first line of the policy instructs IPSec to intercept any traffic from LOCAL-ADDRESS to REMOTE-ADDRESS in the outgoing IP queue. The second line requires that the intercepted traffic is encapsulated within the ESP protocol in transport mode, and sent to the destination address REMOTE-ADDRESS from the source address LOCAL-ADDRESS.

The transparent method of protection provided by IPSec makes it easy to implement within our environment, providing security for networking applications. The disadvantage of IPSec is its method of encapsulation, as the headers added by IPSec will add to the overall bandwidth used. The SRTP and SIAX2 security mechanisms are built into the real-time protocols. The first of these mechanisms, SRTP, is discussed next.

2.3.2 Secure Real-time Transfer Protocol

Secure Real-time Transfer Protocol (SRTP) is an audio/video profile for the RTP protocol that offers confidentiality and integrity for RTP payloads [45]. While SRTP is considered a profile in its own right, in practise it represents an extension to the Audio/Video profile for RTP [14]. The Audio/Video profile has been modified so that the security implementation of SRTP resides between the RTP application and transport layers. RTP packets moving down the stack are intercepted and converted into SRTP packets before being passed to the transport layer. Conversely, SRTP packets moving up the stack are converted to RTP packets and passed to the RTP application layer. Likewise, Real-Time Control Protocol (RTCP) packets are converted into Secure Real-Time Control Protocol (SRTCP) packets when transmitted and vice-versa when received.

Like the AH and ESP protocols, SRTP uses the HMAC-SHA1 hashing algorithm for authentication and integrity. A secret key and the payload are combined to generate a fingerprint, which is in turn added as an authentication header to each RTP packet. SRTP counters replay attacks through the use of a sliding window and Replay List. The Replay List contains an index of all packets which have been received and authenticated. Upon receiving a packet, the packet's index is compared to a list of recently captured packet indexes. The packet is rejected if the index of the packet is smaller then the index of the last received packet, less the size of the sliding window.

Currently, SRTP only supports the AES cryptographic algorithm to provide confidentiality. RFC 3711 allows for the addition of new cryptographic algorithms. New cryptography algorithms should ideally have a low bandwidth overhead, a low computational penalty and a small footprint. A small footprint allows SRTP to be implemented, for example, on mobile and embedded devices such as telephone handsets. The NULL encryption algorithm is also supported for development and debugging purposes.

SRTP does not encrypt the RTP headers. RTP headers include the payload type, synchronisation source identifier and time-stamp. Any custom header extensions to RTP are also not encrypted. RFC 3711 suggests that IPSec, and hence AH or ESP, be used if headers are to be encrypted.

SRTP maintains availability (avoiding denial of service attacks) by only allowing seek-able stream ciphers. A seekable stream cipher does not depend on preceding packets to decrypt a current packet.

Therefore, if packet loss occurs, the cipher will still be able to decrypt the rest of the stream. We now turn to a discussion on the security built into the IAX2 protocol.

2.3.3 Secure Inter-Asterisk Exchange 2

Currently there is no detailed documentation which describes the security behind the IAX2 protocol. So, to gain a better understanding, the IAX2 source code was reviewed. IAX2 performs peer authentication through one of the following methods:

- A plain-text secret password provided by the peer,
- An MD5 hash of the peer's secret password,
- RSA encryption keys.

The first method, a plain text password, is the most insecure. The second method presents an authentication peer with an MD5 challenge. Asterisk provides the peer with a nonce value which is used, in conjunction with the secret password, to generate an MD5 hash. This method ensures that the password is not transmitted in the clear but, even with a nonce value, it is vulnerable to an offline brute force attack. The strongest peer authentication is achieved with RSA encryption keys. This method utilises asymmetric encryption. Authentication is achieved through a peer signing a message with its private key. Signing is performed by simply encrypting a message and making the cipher text and clear text message available. The authenticity of the peer can be checked by decrypting the cipher text message with the peer's public key. If decrypting the cipher text message reveals the clear text message, the authentication is successful.

Confidentiality is achieved by encrypting the IAX2 payload with an implementation of the AES [24] cryptographic algorithm written by Dr Brian Gladman [42]. Padding is added to the IAX2 payload so that it can be divided into even 16 byte blocks which are encrypted individually. The cryptographic key is derived using the fingerprint generated from a second MD5 challenge. Using an MD5 fingerprint to derive the cryptography key has two implications. Firstly, it forces the system to use the MD5 method of peer authentication if confidentiality is needed. Secondly, the confidentiality of a stream is weakened because of the relative ease with which a brute-force attack can be performed on the fingerprint. The IAX2 security mechanism requires a key exchange and method of integrity checking before it is suitable for securing production VoIP systems. An existing key exchange could be incorporated into the IAX2 protocol.

Existing key exchange protocols are described in the section that follows.

2.4 Key exchange protocols

There are several methods for providing a key to secure an end-to-end session. One method is to assign keys statically on each end of a communication channel. However, static key assignment does not allow for fresh key exchanges and results in a single set of keys being used for extended periods of time which is not regarded as good practice. (Exchanging a new key for each communication session is preferred as this limits the amount of information which could be obtained should a malicious third party gain access to the session key.) Another method is to use a public key cryptography infrastructure (PKI) [25]. PKI is based on asymmetric cryptography. Given a public and private key, data encrypted with a public key can only be decrypted with the private key. A PKI infrastructure can be used in the exchange of a symmetric cryptography key. (Symmetric cryptography incurs a lower performance cost than asymmetric cryptography.) Using the PKI method of key exchange requires that public keys are shared between communicating parties, or placed somewhere accessible.

A third method of key exchange enables two parties to derive a key without needing any pre-configured information, such as public keys. This method, named the *Diffie-Hellman* key exchange [71], is explained in the next section.

2.4.1 Diffie-Hellman key exchange

The Diffie-Hellman key exchange exploits the discrete logarithm problem, which assumes that given a prime number p, it is computationally infeasible to calculate x given $y^x mod P$ [9]. Two parties, *A* and *B,* are able to derive the same key through the process that follows. Both parties agree on two numbers which are made public, a prime number p and an integer α, such that α is a primitive root of p. *A* generates a secret integer α_a and calculates $y_A = \alpha^{\alpha_A} mod P$. *A* sends y_a to *B*. *B* does the same calculation and sends y_b to *A*. The values $y_a$and y_b are referred to as *Diffie-Hellman public values*. From the public values each party is able to generate the same shared secret key K. For example, *A* will calculate $K = (y_b)^{\alpha_a}$ [25].

A Diffie-Hellman key exchange is able to derive a key within three exchanges [71]. The ability to exchange a key within three messages is significant, as it fits into the communication pattern of SIP and SDP which also require three messages. This allows us to perform a key exchange with the SIP or SDP protocol which lowers the session initiation time. The sections which follow describe the key exchange options available to real-time applications.

```
rtpmap=0 PCMU/8000
rtpmap=97 iLBC/8000
rtpmap=3 GSM/8000
rtpmap=8 PCMA/8000
rtpmap=101 telephone-event/8000
silencesupp=off
crypto=1 AES_CM_128_HMAC_SHA1_80 inline:+dHGQ4ngRj7oz3kAnH0PmCx.......
```

Figure 2.6: Sdescriptions within an SDP payload

2.4.2 Secure descriptions

Secure descriptions (often referred to as sdescriptions) define a method of exchanging cryptographic keys within the SDP protocol [3]. Figure 2.6 shows a portion of an SDP packet, sent within a SIP INVITE packet. The SDP packet transports sdescriptions information. The `rtpmap` lines describe a list of audio codecs supported by the caller. `silencesupp=off` states that the caller is not making use of silence suppression. The `crypto` line tells the callee that the caller will be using 128bit AES to encrypt RTP payloads. It also declares 80bit HMAC SHA1 as the algorithm to ensure the integrity of RTP payloads. Finally, it defines the encryption key as "`dHGQ4ngRj7oz3kAnH0PmCxFO7V...`". The sdescriptions protocol allows us to perform a key exchange within a VoIP protocol. More importantly, it transports fresh encryption keys for every new session, which is considered better practice in comparison to static keys used for multiple sessions. However, SIP and SDP communication is performed in clear text, which exposes the method and key transported with sdescriptions. A second security mechanism must be used to secure SIP and SDP messages if sdescriptions is used. For example, IPSec could be used to secure SIP and SDP messages. However, instantiating an IPSec session requires its own key exchange. An IPSec key exchange is redundant as we merely wish to exchange a second key with sdescriptions. A preferred method of key exchange is one which is inherently secure, such as the MIKey protocol.

MIKey utilises the Diffie-Hellman key exchange method to exchange a symmetric cryptography key. MIKey is discussed next.

2.4.3 MIKey

The Multimedia Internet Keying (MIKey) protocol [5] is designed to address the key exchange problem specifically for real-time multimedia sessions using the Diffie-Hellman key exchange. Like sdescriptions, a MIKey key exchange can be performed within the SDP protocol. However, instead of simply transmitting the key, MIKey transports Diffie-Hellman public values between the participants.

This method of exchange enables them to derive the same key [1]. The key is never transmitted in the clear, and the use of intercepted information to generate the key is computationally infeasible [25]. It is important to note that without a method of authentication, Diffie-Hellman is vulnerable to man-in-the-middle attacks. A malicious third party, *M*, could intercept the traffic between two communicating ends, *A* and *B*. *M* can manipulate the Diffie-Hellman key exchange such that the cryptography keys are derived between *A* and *M,* and between *B* and *M* instead of between the legitimate participants, A and *B*. The man-in-the-middle vulnerability can be solved by authenticating the MIKey messages with a pre-shared key or PKI mechanism.

A key exchange method similar to SRTP, named ZRTP, is described next.

2.4.4 Zimmermin RTP (ZRTP)

ZRTP (Zimmermin RTP) was developed by Phil Zimmermin, the creator of PGP (Pretty Good Privacy). Like MIKey, ZRTP utilises the Diffie-Hellman key exchange mechanism to derive a common key between two communicating parties [76]. ZRTP is also designed to provide key exchange services for SRTP, described in Section 2.3.2. (ZRTP is an extension of RTP, while MIKey extends SDP.) The key exchange occurs after the signalling has taken place (SIP and SDP). ZRTP messages are embedded into RTP packets as added extensions. These extensions are ignored by an end-point unless it supports ZRTP. If both ends of a session support ZRTP, the policy attributes for an SRTP session are agreed upon and a Diffie-Hellman key exchange is performed [68]. A library, libZRTP, has been developed and implemented into a soft phone called Zfone. Zfone employs an innovative method of authentication: a SAS (short authentication string) is generated during the Diffie-Hellman key exchange. The SAS is used to generate a human readable string which is displayed on the Zfone GUI. Both parties in the session are required to read the string to one another to confirm authentication. This innovative method of authentication also solves the man-in-the-middle problem. If the generated string at each end is different, the communicating parties can assume their Diffie-Hellman key exchange has been tampered with.

Having discussed the VoIP protocols within an Asterisk-based environment and available methods for securing them, the effects on performance after adding security should be taken into account. The performance constraints on VoIP and current analysis tools are discussed in the next section.

2.5 VoIP Performance

The priority for real-time communication is for data to be transported as timeously as possible [63]. Any security addition to a real-time application would be of no value if it introduced a delay signif-

icant enough to degrade overall quality. When adding security to VoIP one also needs to be aware of the impact on bandwidth, as high quality media streams are already using more than 64Kb/s [16]. Moreover, security additions should make economical use of resources such as CPU and memory. This is an important consideration when security is implemented into an embedded system or soft-switch where many connections are handled concurrently. This section begins with an overview of performance constraints placed on real-time applications. This is followed by a description of metrics used to measure VoIP performance. Finally VoIP analysis tools are discussed.

2.5.1 Performance constraints on real-time communication

VoIP media streams are transported by best-effort networks which do not guarantee any level of service, for example Ethernet [15, 63]. However, IPv6 (IP version 6) furthers QoS controls found in IPv4 which could enable service provisioning for real-time applications over IP [26]. IPv6 includes a header which identifies a flow of data rather than individual packets. A router can use this header to provide a stable service per flow of data.

VoIP codecs incorporate mechanisms which can conceal the degrading effects of a best-effort network up to a certain level of network packet loss or latency. The extent to which VoIP can recover from degrading network conditions is described in this section along with recommendations from other studies. This allows the resilience of VoIP to be quantified.

Packet loss should always be expected in IP networks. ITU codecs, such as G.723.1 and G.729a, are able to manage a 5% packet loss without degrading the perceived quality of the encoded stream [43]. On the other hand, real-time communication is sensitive to latency and variation in latency. Variation in latency is often referred to as *jitter*. Latency is introduced at multiple points during the life-cycle of a real-time multimedia payload. For example, latency is added by digitally encoding audio into frames, by encapsulation of frames into packets, and by transporting a packet on a network. All these processes add to the overall delay experienced by a user. The ITU recommends that the maximum end-to-end delay (the latency experienced by two communicating persons) should be less than 150ms [17]. An end-to-end delay of 200ms to 800ms is deemed acceptable for short periods of time. A constant delay of more than 250ms leads to *talker overlap,* which occurs when one party's speech is interrupted by another party's delayed speech. Users of real-time communication are usually tolerant when their session exhibits minor delay. However, excessive jitter resulting from the fluctuation of delay during a conversation is considered intolerable [47].

To avoid quality degradation, transport protocols which induce varied latency, such as the sliding window in TCP [27], are rarely used in real-time applications. VoIP end-points can compensate for varying network latency by activating jitter buffers. Jitter buffers are adaptive real-time packet

Rating	Speech quality (MOS)	Level of distortion (DMOS)
5	Excellent	Imperceptible
4	Good	Just perceptible but not annoying
3	Fair	Perceptible and slightly annoying
2	Poor	Annoying but not objectionable
1	Unsatisfactory	Very annoying and objectionable

Table 2.1: Mean opinion scoring [72]

buffers which perform two tasks. Firstly, a real-time stream is monitored to ascertain the level of varied latency. Secondly, based on the level of latency variation, incoming packets are buffered. Buffering adds latency to the stream but provides a constant delay to the user rather than a varied one [29, 48]. Any addition to a VoIP system should not increase the total latency to an unacceptable level, keeping in mind that some allowance should be made for latency variation.

2.5.2 Metrics for measuring VoIP quality

Measuring the quality of a VoIP system is crucial to evaluating its success. MOS (Mean Opinion Scoring) is one method of quality analysis for voice services. A MOS analysis can be performed in one of two ways. The first way is through the use of SMOS (Subjective MOS) which requires a person to listen to a recording over a telephone and rate its quality [33]. The subject uses the ratings displayed in Table 2.1 to assign a score for two aspects of a telephone system, namely the quality of speech and the level of distortion [72]. The second method is OMOS (Objective MOS) which replaces the human subject with an algorithm. One example is the Perceptual Evaluation of Speech Quality (PESQ) method, an ITU standard for OMOS scoring [8]. Utilising an automated method of MOS scoring is advantageous as results are more objective, and therefore can act as a reliable benchmark in comparative studies. OMOS scoring also saves the cost of human resources.

The success of a VoIP system can also be assessed by measuring the capacity of the network on which the system is to be deployed. Preliminary testing should comprise measuring the available bandwidth and maximum latency of the network. From these measurements, the amount of voice traffic (usually measured by the total number of concurrent calls possible) can be calculated.

The next section describes tools which conduct these measurements as well as simulation tools which make use of MOS analysis.

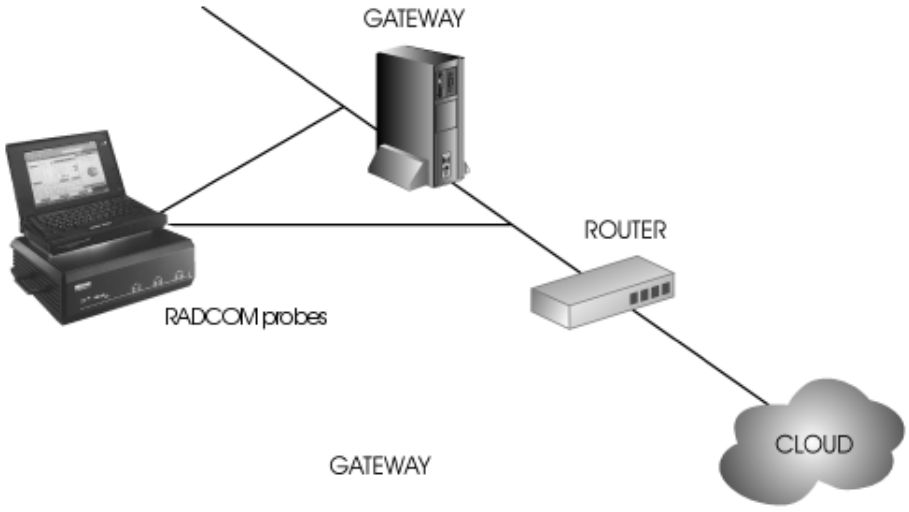


Figure 2.7: RADCOM's probes measuring the performance of a gateway [10]

2.5.3 VoIP Performance analysis tools

A study by Mier and Tarpley ranks seven commercial VoIP analysis tools [46]. Their study provides insight into interception and the data collection methods employed by these tools. Software and hardware mechanisms are used to intercept VoIP traffic, after which the intercepted traffic is analysed. The most common method of interception is *packet sniffing*. The packet sniffing solutions require that a mirrored port be configured to enable the interception of traffic on a switched network medium. Other solutions make use of a hardware device (referred to as a probe) that is placed between two or more communicating parties. The probe intercepts and forwards data to a central point where the analysis is performed. A tool developed by RADCOM performs a comparative analysis of data which is collected from multiple probes strategically positioned on a network [10]. Figure 2.7 shows how a RADCOM probe is placed on each side of a gateway. The data collected by each probe is correlated and used to analyse its performance.

The commercial tools enable the performance analysis of existing networks. These tools assist a network engineer in performance tuning. However, the tools exhibit limited functionality for testbed use where flexibility is needed. For example, an automated testbed would require that the method of analysis and the mechanism for data generation be centrally controlled. Simulation allows for such flexibility, as the virtual network and data collection processes are encompassed within a single system. Simulated performance analysis is discussed next.

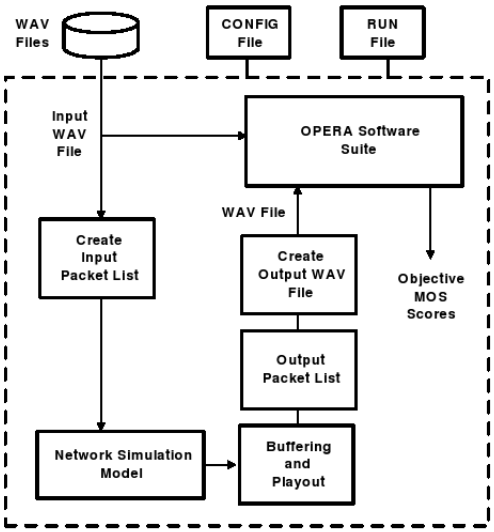


Figure 2.8: Software architecture of VoIP analysis tool by Conway & Zhu [22]

Automated MOS scoring

A VoIP analysis tool, developed by Conway & Zhu [23, 22] simulates the generation of network conditions and produces OMOS scores. The architecture of the tool is shown in figure 2.8. The architecture is based on a collection of VoIP sub-systems which process audio. The tool's input requires a collection of WAV files (a standard audio format [58]), and a CONFIG file. The audio files contain voice signals that are passed to each sub-system where they undergo alterations. The alterations represent a sample of the audio as if it were processed by a VoIP system. For example, one sub-system, the NETWORK SIMULATION MODEL is able to simulate packet loss and latency. The CONFIG file contains the configuration of each sub-system. Two instances of the audio signal are passed to the OPERA SOFTWARE SUITE, an untouched copy of the audio and the audio produced by the VoIP system simulation. The Opera software performs a comparative analysis of the two audio streams and produces a MOS score. There are two distinct advantages of this tool. Firstly, a simulator is always cheaper than implementing and probing a real VoIP system. Secondly the MOS analysis, traditionally performed by humans, is automated in software to provide a more objective result and eliminates the need for human resources.

OPNET is a network simulation tool which allows a user to model the physical layout of a network and then simulate traffic on it [28]. OPNET allows a user to measure the load of each node in their virtual network by analysing a wide range of metrics. A study by Han and Chung uses OPNET to measure the performance of VoIP audio codecs in relation to network queueing models [34]. The physical layout of their OPNET simulation is shown in figure 2.9. Han and Chung simulated packet

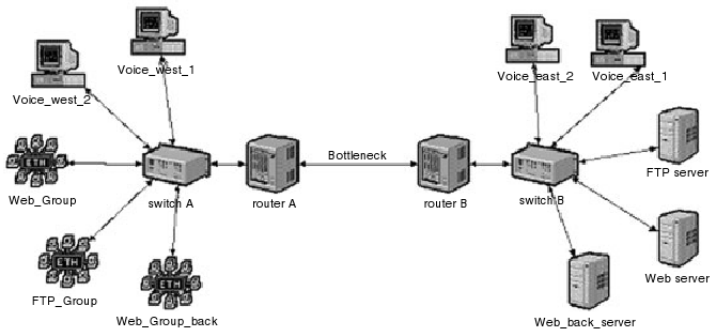


Figure 2.9: Physical network layout simulated with OPNET [34]

loss, latency, background traffic (HTTP, FTP and electronic mail) and bandwidth limits. Having set up a virtual representation of the real network, Han and Chung simulated voice traffic based on the G.723 and G.729 codecs. The simulator measured the effects on the voice traffic by various queueing methods used on the router. OPNET enabled Han and Chung to adjust variables in their simulator, such as the routers' queueing methods, the amount of voice traffic, the amount of background traffic, and QoS techniques, such as setting the TOS (Type Of Service) bit in IP packets. Han and Chung found that the G.729 codec in conjunction with a WFQ (Weighted Fair Queueing) model on the router produced best delivery of voice data. The flexibility of OPNET makes it a viable performance analysis tool for this study. However, a tool which could analyse our physical environment would produce results that better represent the real-world characteristics of our system.

2.6 Summary

In this chapter, the VoIP protocols found in our environment have been explained along with initial security mechanisms for each protocol. Appropriate security methods for our VoIP environment were described, namely IPSec, SRTP and SIAX2. The problem of providing a key to each end of a communication session so that a security method can perform cryptographic operations was identified. Solutions to the key exchange problem, within the context of real-time communication, were discussed. Finally, the area of VoIP quality for this study was addressed. This included a description of the constraints on real-time communication such as latency, jitter, packet loss and limited bandwidth. Metrics for measuring the quality of VoIP systems were also investigated, and a review of available tools that measure VoIP performance was performed. The predominant method of VoIP analysis was determined to be simulation. Tools which investigate live VoIP systems are available, but are not flexible enough to allow the control of a testbed. An ideal analysis tool should be able to control the nodes within a testbed as well as collect specific performance information related to the environment

under study. Therefore, analysis software was designed and implemented specifically for this study. The software, named DRAPA, controls a testbed and facilitates data collection.

DRAPA is described in the next chapter.

Chapter 3

Development of a performance analysis testbed - DRAPA

3.1 Introduction

Of the available VoIP analysis tools discussed in the second chapter, none were able to manage the devices within our environment so that automated testing could be conducted. In order to perform a performance analysis of security methods implemented within an Asterisk-based VoIP environment, a testbed had to be designed and implemented. The testbed software is called DRAPA, the Distributed Real-time Application Performance Analyser. Two types of testing are possible: real-domain and simulation. Real-domain testing refers to an analysis of a physically implemented system, rather than a virtual, simulated one. An advantage of real-domain testing is that it generates accurate results. However, a disadvantage of real-domain testing is that physical resources such as a network and VoIP devices are required. The results of previous studies, discussed in Section 2.5, are predominantly obtained through simulation. Our research focus is the real-world performance of a secure VoIP system. However, to extend its use DRAPA was designed to be sufficiently flexible to analyse any aspect of a VoIP system.

DRAPA identifies and addresses three key areas which are needed for real-domain VoIP performance analysis. Firstly, it defines a method of controlling physically distributed nodes on a network. VoIP phones and servers are two examples of such nodes. Secondly DRAPA allows us to make use of existing resources, such as laboratory computers, and it incorporates functionality to handle nodes as shared resources. Thirdly, DRAPA provides a clean boundary between testbed logic and specific analysis logic. The boundary exposes an API which provides the flexibility to test different aspects of a VoIP system. DRAPA took six months to design and implement, and consists of 1411 lines of code.

To ensure that the data generated by DRAPA is correct, a simulated study by Salah and Alkhoraidly [64] was reproduced using DRAPA . The CPU usage of the VoIP server and number of voice packets generated were used as comparative metrics. The results ensured that the data generated by DRAPA is in line with other studies.

3.2 Chapter structure

DRAPA is made up of physical nodes and software agents distributed over an IP network. Section 3.3 describes the architecture of each node and the physical layout of the testbed. Section 3.4 describes the distributed software agents which control testbed nodes and facilitate the collection of data. The Test Management Server (TMS), where DRAPA's central control is performed, is described in Section 3.5. Flexibility is provided to a DRAPA user through an API which is in the form of pluggable modules. The pluggable modules are described in Section 3.6. The collection and processing of data is described in Section 3.6.2. To ensure that DRAPA generates results comparable with other studies, an initial validation of DRAPA is presented in Section 3.8. Problems were encountered during the initial validation of DRAPA. These problems and the consequent improvements to DRAPA are discussed in Section 3.9. Finally, a summary of this chapter, highlighting contributions and areas of use are presented in Section 3.10. A discussion of possible future improvements on DRAPA can be found in the last chapter of this document in Section 5.3.

3.3 DRAPA's architecture

DRAPA has been designed to comprise five node types:

- end-points;
- VoIP servers;
- traffic shapers;
- a web interface;
- and a centralised controller.

The testbed used in this project utilises laboratory computers as end-points, where a total of 100 end-points were available. The testbed also includes one VoIP server and one traffic shaper. Figure 3.1 is a functional schematic of the logical layout of these nodes. This section introduces each type of node and describes their architecture, relationship and role within DRAPA.

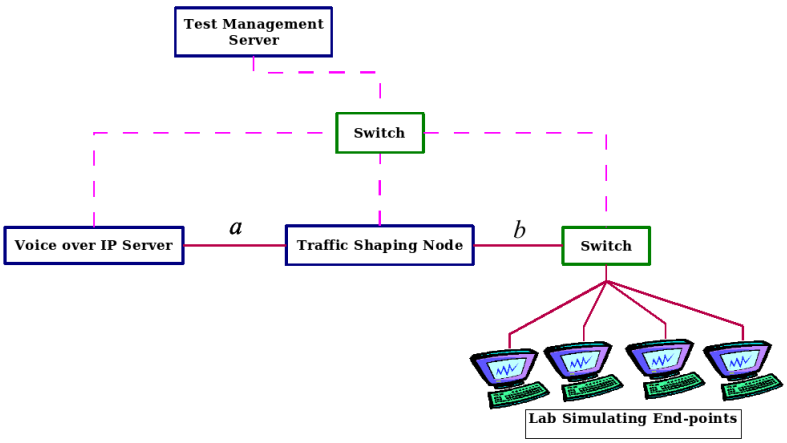


Figure 3.1: Functional schematic of DRAPA

3.3.1 VoIP server

The VoIP server runs Asterisk on top of the Linux operating system. The VoIP server is equipped with two network interfaces (see figure 3.1). Having two network paths to the server allows DRAPA to perform monitoring and control operations without interfering with the traffic generated on the testbed itself. The separation is achieved by using the solid interface for analysed VoIP traffic and the dotted interface for monitoring and control. In figure 3.1, solid lines refer to network links which carry monitored VoIP traffic and dashed lines correspond to network links which are used for monitoring and testbed control. During a test, real-time streams are created between the end-points and the VoIP server. The flow of real-time multimedia is routed via the traffic shaping node.

3.3.2 End-points

DRAPA performs data capture in a real domain rather than a simulated one. Therefore, physical resources are required so that a complete system can be implemented. Fortunately, the author's university has a high availability of student computers grouped in laboratories, which DRAPA can utilise as end-points. The laboratory machines are, therefore, a shared resource between DRAPA and the students. Collected data cannot be relied upon if a machine is simultaneously being used by DRAPA and a person in the laboratory. This creates an important design constraint: DRAPA can only make use of idle laboratory computers. The mechanism for the exclusion of end-points which are being used for another function is explained in Section 3.4.

Each laboratory computer, like the VoIP server, runs Asterisk on top of the Linux operating system. Asterisk was also used on the end-points to ensure uniformity, which makes the overall design and

maintenance easier. DRAPA is capable of running other end-point software, for example a soft-phone. The end-points are connected to a single switch which is linked to the traffic shaping node and in turn to the VoIP server (figure 3.1).

3.3.3 Traffic shaper and accounting node

The traffic shaping node (TSN) performs two tasks. Firstly, it can induce controlled network conditions. Secondly, it can measure the load at a point on the network. In our implementation, a TSN is a machine running FreeBSD. The FreeBSD kernel is re-compiled to enable the Internet Protocol Firewall (IPFW), bridging, and Dummynet [61, 73]. This combination allows us to simulate a variety of network conditions. For example, we are able to drop packets, induce latency, duplicate packets and limit bandwidth. The network load is monitored by using TSNs. For example, a TSN could be used to record the rate of data sent to and from a default router and its network. In this case, a TSN would monitor the bandwidth and count the packets which pass any of the links to the router in question. The TSN would be physically connected to the router and to the network switch and the measurement is achieved through the bandwidth and packet counters provided by IPFW. Address headers, IP protocols and port numbers can be used to differentiate and monitor separate traffic flows. During the data collection process, the bandwidth and packet count information is captured from the IPFW counter rules and stored in a database (this is explained further in Section 3.6.2). The TSN in figure 3.1 has three network interfaces. The first and second network interfaces on the TSN form an Ethernet *bridge*. The bridge connects to two other network nodes, a and b, such that the TSN intercepts IP traffic between a and b on the Ethernet layer. An IP address is bound to the third interface, which is connected to the main network switch. This third interface is used to send monitoring information to the test management server and to receive control requests from the same server. As in the case of the VoIP server, the third interface allows data to be collected from the TSN without interfering or relying upon the monitored links, hence reducing the effect of Observer's Paradox.[1] One or more TSNs can be placed at any physical point on the network. Our testbed makes use of a single TSN which is situated between the VoIP server and network switch.

3.3.4 Test management server

The last entity of DRAPA is the Test Management Server (TMS). The TMS is broken into three components. The first component is the core of the TMS which is responsible for coordinating the testbed. The TMS facilitates the actions of the end-points, the VoIP server, and the traffic shaper.

[1]The Observer's Paradox describes the phenomenon where the act of observation influences the results of the experiment.

These actions form part of the *central management system* which is explained in Section 3.5. The second component of the controller provides the flexibility of DRAPA's customisation. The flexibility is achieved through the implementation of user configurable pluggable *action modules*. (The pluggable action modules are explained in Section 3.6.) An action module contains instruction sets which are used by the central management system to control each element of the testbed. To collect data and facilitate control of the system, the controller makes use of distributed agents which report on and manage each DRAPA node. The distributed agents are discussed in the next section.

3.4 DRAPA's distributed management system

Having described the architecture of DRAPA and introduced each node type, the logic which makes up the DRAPA software can be described.

3.4.1 End-point management

To maintain control of the end-points, each end-point is tagged with a state. End-points are always in one of the following three states:

- AVAILABLE - The end-point is online and ready to be used in a test;
- WORKING - The end-point is online and currently used in a test;
- USED - The end-point is online but is being used by a person in the laboratory.

These states are displayed on the web interface, which is discussed further in Section 3.7. Note that an end-point which is offline (because it is not switched on, or is booted into the Windows operating system instead of Linux) is not registered in the list. Therefore, an OFFLINE state is not required.

Two agents, the DRAPA daemon (DRAPAD) and DRAPA slave (DRAPAS) manage the end-points. The daemon ensures that each AVAILABLE end-point is running up-to-date DRAPA software and performs upgrades when software is found to be old. The slave is responsible for maintaining a register of end-points and their current states.

The daemon can be run as an independent entity within the DRAPA testbed. However, it is possible to run the daemon in conjunction with the controller on a single machine (for example, the TMS). The daemon iteratively performs the following actions:

1. It queries all end-points in the AVAILABLE state to ensure that they are still online. An end-point which does not respond is removed from the list of registered end-points. (End-points usually de-register with the TMS when shut down properly.)

2. While checking AVAILABLE end-points, the daemon ensures that they are up-to-date with the latest version of the slave agent, with Asterisk binaries, and Asterisk configurations. Any out-of-date software or configuration is upgraded by the daemon.

3. Lastly, the daemon probes predefined IP addresses for hosts which may not have registered properly when coming online. Any host found is registered as an online end-point and its state is set to AVAILABLE, unless it is in use. After this search, the daemon begins its cycle again with the first action.

The slave agent runs on each end-point and performs the following tasks:

1. It informs the TMS when an end-point comes online or goes offline.

2. It informs the TMS when an end-point is in use by a person in the laboratory.

3. It ensures that all testing processes are closed down should a person begin to use the end-point. The slave also notifies the central controller when the state of the end-point on which it runs changes.

When the central controller is alerted to any of the above events, an appropriate action is taken. For example, if a laboratory host is currently being used in a test and a student logs into the machine, DRAPA discards the current test. This functionality enables DRAPA tests to co-exist with normal laboratory usage. Should a test be discarded, DRAPA restarts the test ensuring that any laboratory machine in use is not incorporated into the new test. The logic processing these decisions resides on the TMS and is explained further in Section 3.5.

3.4.2 Data collection agents

Data collection agents are responsible for monitoring a testbed node and making performance information available to the DRAPA controller (see Section 3.5). Distributed data collection agents can reside on VoIP servers, TSNs and end-points. The DRAPA user configures each agent to specify the type of data which is to be collected. The customisation of data collection agents allows DRAPA to be used to analyse any specific aspect of a VoIP system. For example, if a new jitter buffer were studied, the jitter buffer may log performance indicators which a customised data collection agent would

monitor and make available to the DRAPA controller. Other examples of monitored performance metrics are bandwidth usage, CPU time allocation and memory usage. A minimal configuration of each agent is a sampling interval which determines the frequency at which an agent probes for performance information. Data collected by an agent is stored on the host node's hard-disk for collection by the DRAPA controller. The transfer of collected data from the distributed agents to a central database is discussed in Section 3.6.2.

3.5 DRAPA's Central Management System

The logic behind DRAPA is contained within the DRAPA controller (DRAPAC). The controller is responsible for issuing instructions to the end-points, to the VoIP servers, and to the traffic shapers. The controller also makes decisions in order to utilise available resources effectively and to maintain an even distribution of data. The section that follows describes four processes which are performed iteratively by the controller. Each iteration is referred to as a *test cycle* and consists of the creation of communication sessions within a VoIP system, followed by the measurement and collection of data, and subsequent closing of the sessions. Test cycles are repeated until interrupted by the user. The repetition of test cycles is necessary to produce results which better represent the phenomena within the VoIP system. For example, the repetition of test cycles helps to smooth out any ad-hoc spikes caused by network interference, operating system house-keeping, and factors outside the control of the DRAPA testbed.

3.5.1 Two modes of operation

The controller begins a test by calculating the number of sessions to be created. The number of sessions relates directly to the load placed on the system. The session load calculation starts by taking into account the number of online end-points which are in the AVAILABLE state, x. DRAPA is run in one of two modes which define the extent to which each end-point is used. The first mode only allows one session per end-point. Therefore, the total number of sessions is $x/2$. For example, 10 available end-points would see a maximum of 5 sessions being created, one session between two end-points.

The second mode allows end-points to create more than one session, allowing DRAPA to distribute a larger number of sessions over a given set of end-points. The first mode is preferred, as it better represents a realistic VoIP system, but it has two limitations. Firstly, the total number of sessions is bound by the number of available end-points. Secondly, unless the use of end-points is dedicated to the testbed, the maximum number of possible sessions will vary during an experiment in relation

to the end-points serving other tasks, whereas the second mode allows an experiment to test the maximum number of sessions with fewer end-points. The second mode sacrifices some real-world authenticity but it allows DRAPA to better utilise available resources. The second mode is configured with a `low limit`, `high limit` and `step value`. The `low limit` defines the minimum number of calls created while the `high limit` defines the maximum number of calls to be made. The `step value` is used to configure the number of calls created at one time. When DRAPA is run in the second mode the minimum number of calls are initially created. After every test cycle the current number of sessions is increased by the step value. The sessions are closed down when the maximum number of sessions is reached and the test starts again.

In contrast to the second mode, the first mode reviews the collected data and makes a decision on the number of sessions to test next. This process is described next.

3.5.2 Load calculation

An identifier named TYPE is used to distinguish the data collected in each experiment. The name of the action module associated to an experiment is used to set the value of TYPE (see Section 3.6). During a test cycle, the number of sessions to be created in using the first mode is calculated as follows: If $x/2$ is greater than any previously recorded number of sessions for the current TYPE, all end-points are used to create $x/2$ sessions. If $x/2$ is less than or equal to the largest recoded number of sessions for the current TYPE, the controller searches the collected data to find a record generated from y number of sessions, where y satisfies two conditions. Firstly, $0 < y <= x/2$, and, secondly, the number of tests associated with y appears the least number of times. In other words, the controller looks for the number of sessions that has been tested the least. If the distribution of tests for each value of y is the same, the maximum possible value for y is selected. This makes use of all the end-points while they are still available (the command line dialogue of DRAPA run in mode 1 can be found in Appendix C).

The second mode utilises a simpler selection method, as y is not as strictly bound by the number of available end-points. It simply iterates between one and the maximum limit. Another variable, r, is used to limit the number of sessions allowed per end-point. The value for r should be set such that the load on each end-point is restricted to a value smaller than its maximum capacity. The limit ensures that any one end-point is not overloaded with sessions. The total number of sessions, y, is calculated such that $0 < y < x * r/2$.

3.5.3 Test cycle control

After calculating a value for y, the controller executes user-defined functions, starting with the creation of sessions. The user-defined functions are encapsulated in a pluggable module which is explained further in Section 3.6. After creating y number of sessions, the controller prepares the data collection agents and then enters a sleep state for a pre-configured amount of time, like 60 seconds. While the controller waits in sleep state, the VoIP system holds its state and the distributed agents monitor and log the performance of specific entities, for example the CPU and network load of a VoIP server. When the wait period expires, the controller performs a test to ensure that y number of sessions still exist. If there has been a problem and sessions were lost, through, for example, the rebooting of an end-point, the current test is discarded. If the number of existing sessions is correct, the controller probes each agent and collects data generated during the current test cycle. The data collection is discussed further in Section 3.6.2. After the data has been saved to a database, the sessions are closed down and another test cycle begins.

A more detailed discussion of the functions performed during a test cycle is found in the next section. These functions are defined by the user in the form of modules plugged into DRAPA.

3.6 DRAPA's pluggable action modules

DRAPA's flexibility lies in its pluggable action modules. A module defines sets of instructions which are used to control nodes during each test cycle. A module is made up of functions. Each function defines a set of instructions which perform a specific task within the testbed. The modules, like the controller and distributed agents, are written in Perl [11] and are a template of basic operations which is customised for each specific test before DRAPA is run (the module template can be found in Appendix B).

3.6.1 Modules

There are eight user defined functions in each module. The first function is `create_session`. The controller passes three arguments to this function: a caller address, a callee address and an extension number which the caller should dial. These arguments are used to instruct the caller to create a VoIP session to the callee. The two end-points are selected by the controller, as described in Section 3.5. The second function, `close_sessions`, instructs end-points and VoIP servers to close down their active VoIP sessions. The `count_session` function should return the number of active VoIP sessions by querying end-points and any VoIP servers. The controller executes the

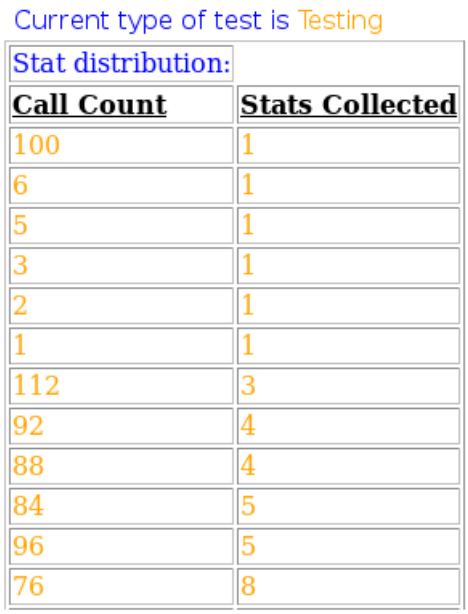
`count_session` function before collecting data from the distributed agents. If the number of sessions returned by the `count_session` function is less than what was expected, the current test cycle should be discarded. The controller executes the `discard_test` function which performs any action required in discarding a test. For example, data already written to the database can be purged. The last three functions, `init_counters`, `final_counters` and `do_math` perform data collection tasks. These functions are discussed in Section 3.6.2.

The use of customisable modules provides significant advantages. A DRAPA user is able to switch between different experiments with minimal overhead. For example, if the performance of SRTP and IPSec are being investigated, two modules would be written. Each module would contain specific instructions and configurations associated with the analysis of each security method. For example, the `create_sessions` function for IPSec might instantiate security associations between the end-points before creating a session. The flexibility provided by the action modules extends to physical resources too. One module could utilise laboratory computers as end-points while another could utilise hard-phones, hence template modules could be provided for architecturally different testbeds.

3.6.2 Data Collection

Data collection and processing is customised by the DRAPA user within an action module. An action module contains three functions dedicated to data management, `init_counters`, `final_counters` and `do_math`.

The actions required for collecting data from distributed agents are defined in `init_counters` and `final_counters`. `init_counters` is executed by the controller after the required number of VoIP sessions have been created. `final_counters` is executed after the controller's wait period has expired. This allows a comparison of the data collected before and after the completion of a test cycle. One could make use of the SNMP [13] or SSH [75] protocols to fetch data collected by agents running on the VoIP server and a traffic shaper. The author selected SSH as it also enables the controller to execute commands on the remote nodes. Remote collection requests to the VoIP server could collect CPU information while requests to the traffic shaper could collect bandwidth information. The last function executed during a test cycle is `do_math`. Database house-keeping and preliminary data processing is performed by the `do_math` function. For example, values in the database could be converted to a standard metric to facilitate the comparison of results.



Current type of test is Testing

Stat distribution:

Call Count	Stats Collected
100	1
6	1
5	1
3	1
2	1
1	1
112	3
92	4
88	4
84	5
96	5
76	8

Figure 3.2: Test type and data distribution table

3.7 Web interface

DRAPA features a web interface, shown in figures 3.2 and 3.3. The interface displays three types of status information. The first two are seen in figure 3.2. Firstly, the interface displays the type of test currently being performed by the DRAPA controller. Secondly, the Stat distribution table displays the distribution of collected data for each set of sessions.

The third area, the Online endpoint status table (figure 3.3) lists the symbolic address and state of each online end-point. (Offline end-points are not listed in the table.) When an end-point is in the WORKING state, the call type and remote peer fields are also displayed. The call type indicates whether the host is the caller or callee. The remote peer displays which end-point is on the other side of the session.

3.8 Initial evaluation of DRAPA

After developing an experimental tool such as DRAPA, it is important to validate it. For this exercise, an investigation, via simulation, into the performance of the G711 audio codec [38], by Salah and Alkhoraidly [64], was reproduced using the DRAPA framework. The investigation by Salah and

Online endpoint status:

Host	State	Call Type	Remote Peer
ug101.ict.ru.ac.za	Used		
ug107.ict.ru.ac.za	Available		
ug125.ict.ru.ac.za	Working	caller	ug133.ict.ru.ac.za
ug132.ict.ru.ac.za	Available		
ug133.ict.ru.ac.za	Working	calee	ug125.ict.ru.ac.za
ug141.ict.ru.ac.za	Available		
ug142.ict.ru.ac.za	Used		
ug147.ict.ru.ac.za	Available		
ug149.ict.ru.ac.za	Available		

Figure 3.3: End-point status table

Alkhoraidly uses the OPNET network simulation tool [28] to study the feasibility of adding a G711-based VoIP system to a small office network. Their simulation is described in Section 3.8.1.

3.8.1 Testbed setup

DRAPA must be set up to match the study by Salah and Alkhoraidly. The topology of the simulated network is based on a case study of a small business located in a triple storey building. Figure 3.4 shows the network layout comprising three local area networks, one per floor, interconnected with two managed switches and a router. All network entities are linked at a speed of 100Mb/s. Salah and Alkhoraidly incorporated two other attributes from the case study network. Firstly, they measured the network utilisation during peak usage time and, secondly, they reserved 25% of the available network capacity for future growth. Their aim was to identify the number of sessions which could be placed on the network given the remaining network capacity. These two attributes were not incorporated into the DRAPA analysis as our aim was simply to ensure that our results correlated with those of the simulation. However, the interfering data could be generated by any of the DRAPA nodes and the reservation of network capacity could be enforced by a bandwidth-shaping node.

To compare the simulated results to those generated by DRAPA, the number of voice packets generated per second was investigated. DRAPA is validated if the two investigations generate the same number of voice packets, in relation to the number of concurrent sessions. (The simulator used by Salah and Alkhoraidly is hereafter referred to as the S&A simulator.)

The S&A simulator was configured to create three VoIP sessions every two seconds. To mimic this, a DRAPA action module is configured to do the same. Every two seconds, an end-point is chosen and

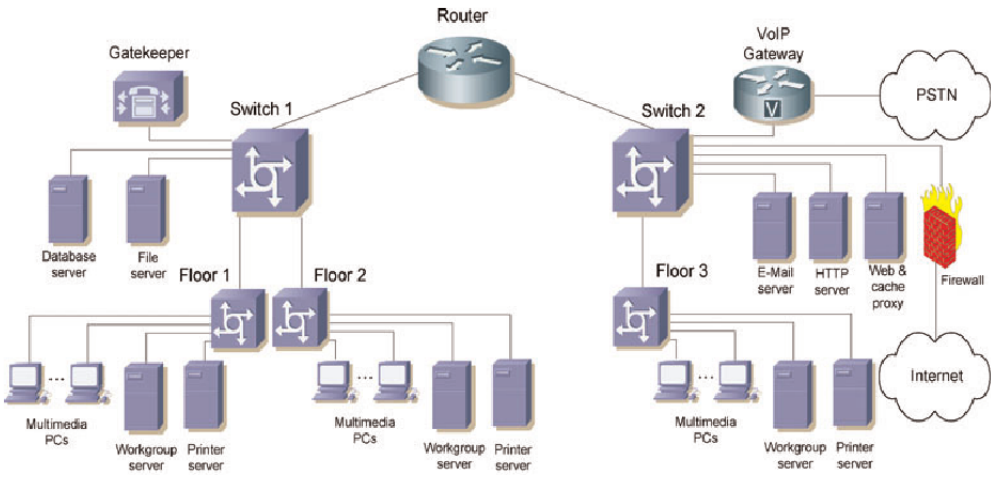


Figure 3.4: Schematic network layout from simulation (source [64])

made to place three sessions to another end-point, via the VoIP server. The end-point is selected such that the number of sessions is distributed evenly over each end-point. Routing the sessions via the VoIP server mimics the load placed on the router in the S&A simulation. The results documented by Salah and Alkhoraidly focuses only on the router as it was found to be the bottleneck in the system. Therefore, the number of packets passed through the router in the S&A simulation is compared to the number of packets passed through the DRAPA VoIP server. The DRAPA action module was configured to collect data two seconds after each addition of three sessions to the system.

3.8.2 Results

Figures 3.5, 3.7 and 3.8 are taken from Salah and Alkhoraidly. Figures 3.6, 3.9 and 3.10 are generated from data collected by DRAPA. These two groups of graphs plot the same type of data, enabling the comparison of DRAPA to the S&A simulator. The graphs generated by the S&A simulator are plotted in relation to the simulated network time, which was eight minutes in total. The number of concurrent sessions, x, at any time, m:s (minutes:seconds), can be calculated as $x = 3 + (m * 60 + s - 70/2) * 3$. (70 is subtracted to account for a delay of 70 seconds at the beginning of the simulation before the first three sessions were created.) Figure 3.5 plots the total number of packets per second (pps) generated by the S&A simulation. For example, at 2 minutes and 30 seconds 123 sessions existed, generating 12,300 pps. Figure 3.6 plots the total number of packets per second generated by DRAPA. At 123 sessions DRAPA was generating 12,363 pps which differs from the simulated analysis by 63 pps. Inspecting the data generated by the simulator we found that exactly 100 pps per session were generated. (It looks like the S&A simulator is applying the exact theory of the g711 codec and generating 100 pps per session, 50 pps per stream and two streams per session). There is a difference

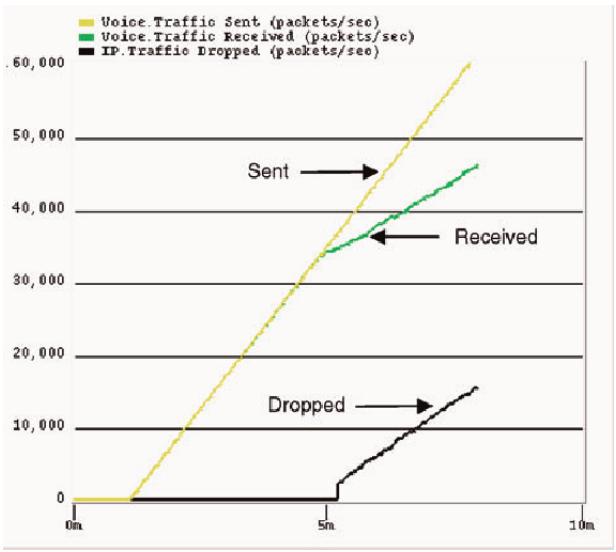


Figure 3.5: Total packets per second from simulation [64]

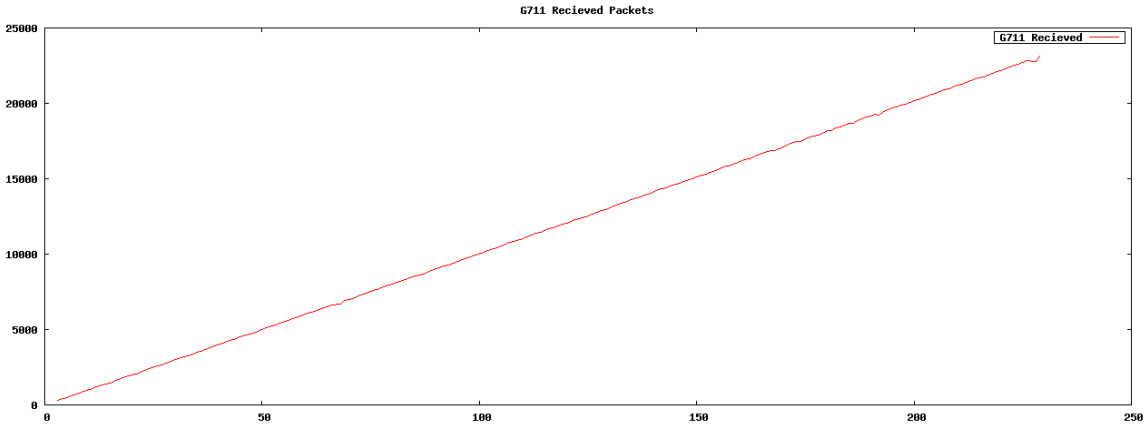


Figure 3.6: Total packets per second from DRAPA

[h]

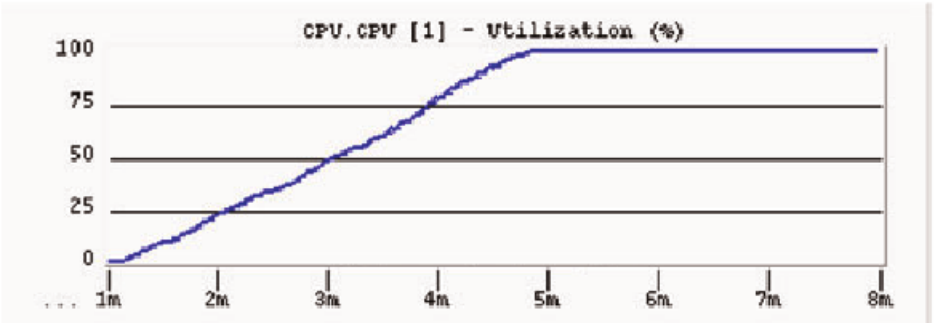


Figure 3.7: Simulation of CPU utilisation of router [64]

of 0.35% between the number of packets generated by DRAPA and the number generated by the S&A simulator; DRAPA is producing results in line with the S&A simulator. The small increase in packet rate on the DRAPA testbed is caused by additional traffic from IP supporting protocols, such as ARP [54] requests.

3.8.3 Resource Exhaustion

To determine the potential number of sessions on our network and to further demonstrate the use of DRAPA, sessions were added until a resource had been exhausted. Both the S&A simulator and DRAPA found the CPU resource to be the bottleneck in the system. Figure 3.7 plots the CPU utilisation of the router in the S&A simulation. At 4 minutes and 48 seconds, or 330 sessions, the router's CPU became saturated and began to drop packets. The number of packets received, sent and dropped is plotted in figure 3.8.

Similar results are evident when using DRAPA. Figure 3.9 plots the CPU usage of the DRAPA VoIP Server. The usage is measured with the kernel *tick*, which is explained in Section 4.6.2. (In short, a tick represents 10 milliseconds of accounted for CPU time.) Figure 3.9 plots the number of ticks for each CPU state on the VoIP server. The predominant CPU state is SYSTEM, because the Asterisk process runs as a system process. The graph shows that at 240 sessions, the number of IDLE CPU ticks reaches zero. At this point the CPU is saturated and thus the VoIP server does not accept any new sessions. When using the G711u codec, the largest number of concurrent sessions we can achieve is 240. Figure 3.10 plots the number of packets received and sent by the VoIP server. Figure 3.8 shows that the number of ingress packets, in relation to the router, continue to increase while the number of egress packets remains the same. In the case of DRAPA, figure 3.10 plots an equal number of ingress and egress packets per second until the 240 sessions is reached. If the VoIP server were able to accept sessions after 240, the plots in this graph would exhibit a similar pattern to that generated by the S&A simulator. However, unlike a simulator, DRAPA is limited to physical resources.

[h]

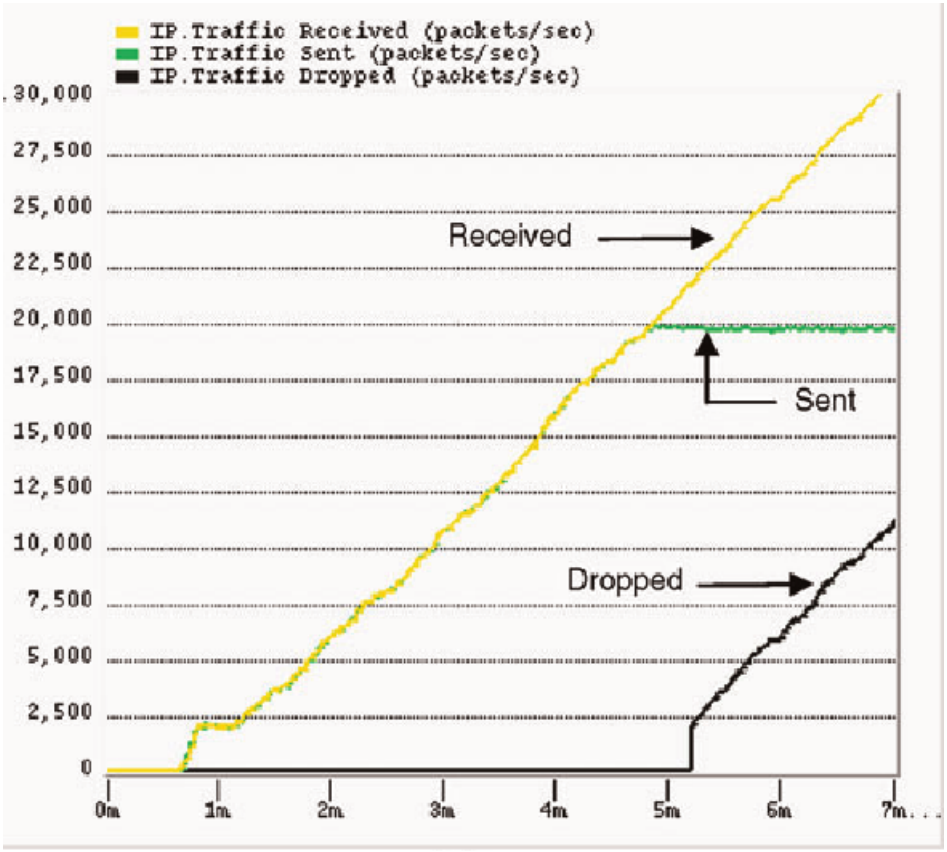


Figure 3.8: Packets per second received and sent by the router [64]

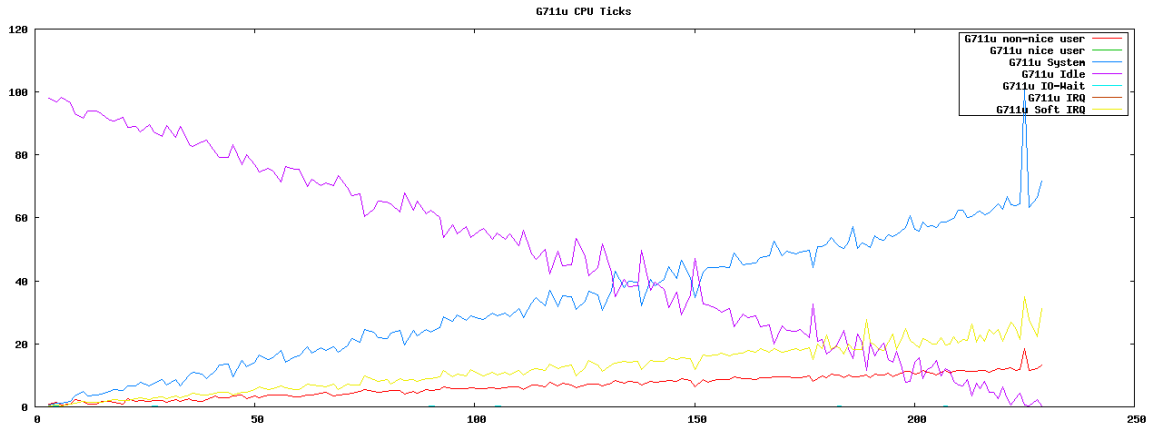


Figure 3.9: CPU usage for the DRAPA VoIP server

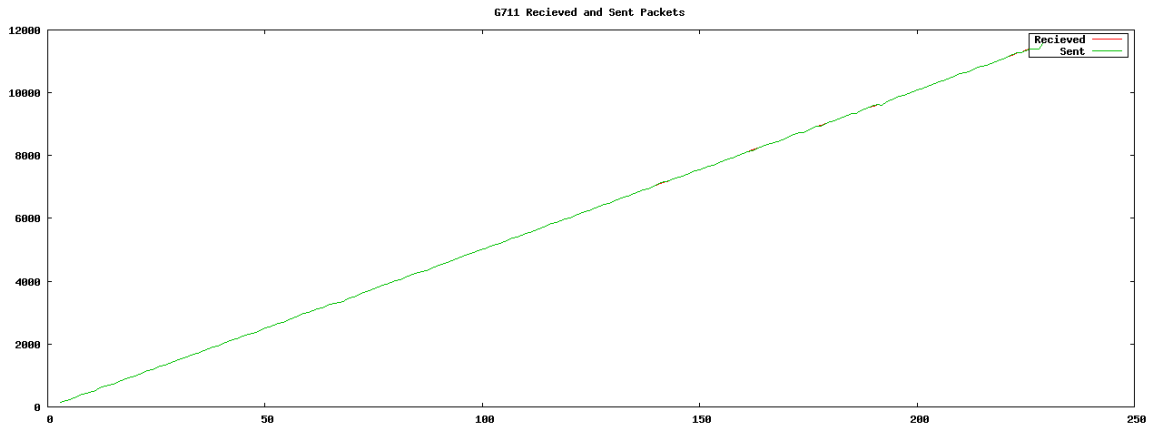


Figure 3.10: Sent and received packets per second by DRAPA VoIP server

3.9 Limitations and improvements to DRAPA

During design, implementation and validation of DRAPA, several problems were encountered and alterations were made to DRAPA to address these problems. Significant issues experienced and their solutions are discussed in the following sections.

3.9.1 Lengthy experiments

The first problem faced during the validation was the length of time taken to complete a test cycle. After x number of sessions are created and initial counter values are collected, the system sustains its state for a pre-configured amount of time, while the distributed agents monitor and log data. The pre-configured waiting time is inescapable: if the monitoring period of the testbed is set to 60 seconds, the total execution time of a test-cycle is 60 seconds plus the time taken to create and close down the sessions (overhead time). Assuming a monitoring period of 60 seconds, a test cycle could take a total of three minutes. The time taken to start and close down each test cycle was decreased by creating and closing down sessions concurrently. This reduced the overhead to less than five seconds for any number of sessions during a test cycle. On average, the time taken to complete a full experiment with a period of 60 seconds and sessions from one to 240, is 4 hours and 15 minutes. Before the addition of concurrent session management, twelve hours were needed to complete the experiment.

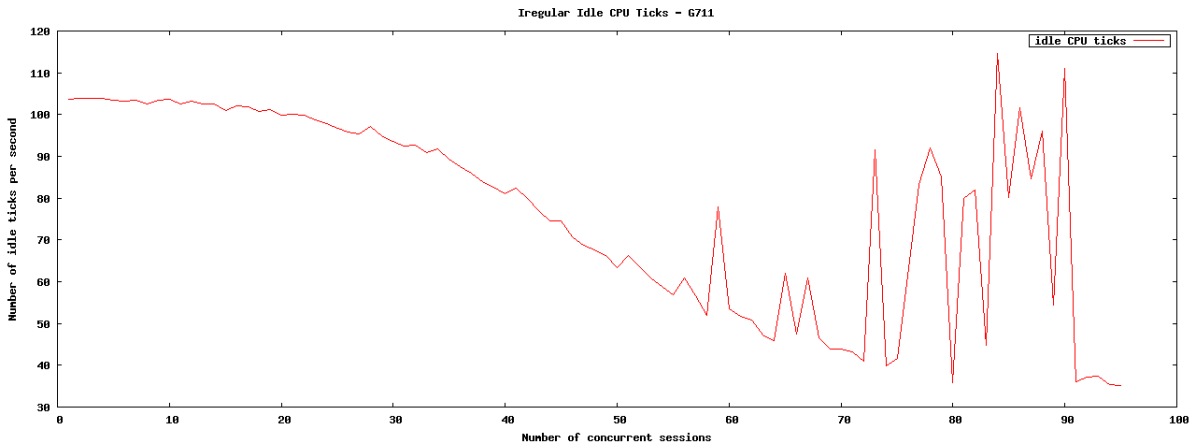


Figure 3.11: Inaccurate CPU Graph

3.9.2 Scattered patterns in graphs

The second problem became apparent upon reviewing the initial results generated by DRAPA. The bandwidth and CPU graphs exhibited a regular, linear trend up to 56 concurrent sessions. For any number of sessions after 56, the graphs exhibit a scattered pattern. Possible causes of this problem were:

1. Exhaustion of a resource such as CPU or bandwidth, creating a bottleneck in the system.
2. Interference from external traffic and/or processes on the network.
3. Incorrect data collection.
4. Faulty switching infrastructure between the end-points and asterisk server.

Figure 3.11 plots the CPU load of the VoIP server. At 56 concurrent sessions there were 60 available idle CPU ticks per second after which the data became scattered. This number of ticks was high enough to rule out the possibility of saturating the CPU.

Figure 3.12 plots the bandwidth load: less than 32Mb/s were used for 56 concurrent sessions, which is well below the maximum speed of the network (100Mb/s). The bottleneck of the network used in this specific analysis is the 100Mb/s link between the switching infrastructure and the VoIP server. To be on the safe side, we measured the available bandwidth between the VoIP server and a machine acting as an end-point using the BING tool [39, 57]. `Bing` sends two sizes of ICMP `echo_request` packets to a pair of remote hosts and attempts to measure the available bandwidth between two

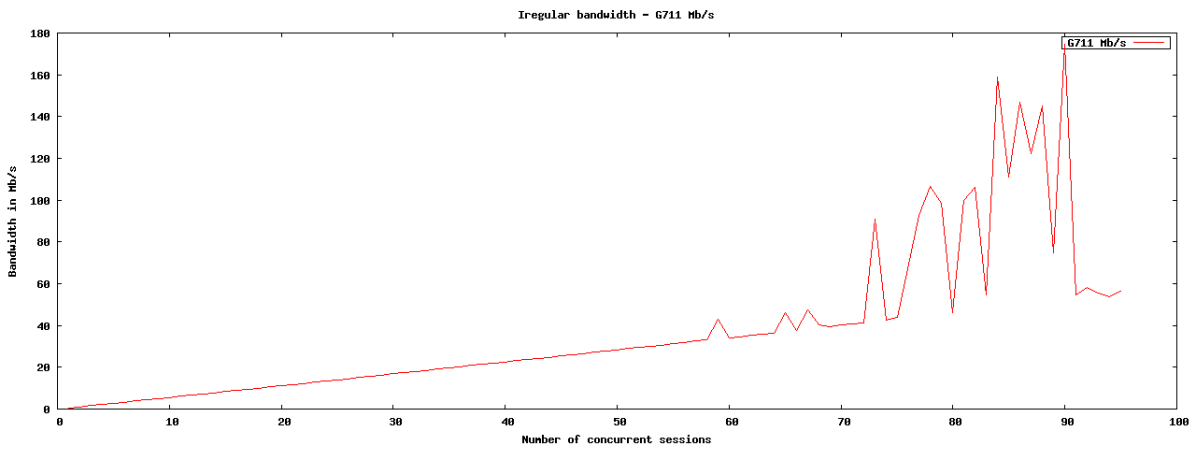


Figure 3.12: Inaccurate Bandwidth Graph

points on a network. The first host should reside on the VoIP server's local network, while the second should reside on the second point of the network. In this case, the bandwidth manager and one of the laboratory machines acting as an end-point were selected as the first and second points respectively. Figure 3.13 shows the output generated by Bing. Bing reported that the link between the VoIP server and the selected end-point was 102.4Mb/s. Therefore, we can safely rule out the notion that a choke on the available bandwidth between the end-points and VoIP server caused the irregular results.

The data collection agents included logic to flag irregular data by detecting unexpected conditions, such as counters that roll over and divisions by zero. However, to completely exclude the possibility of incorrect data collection, the graph generation software was made to perform an extra round of checks. This confirmed that data was being collected properly.

Lastly, we considered the possibility of network interference and faulty equipment. Two changes were made to the testbed before a second round of results were collected. The first change was at the physical network layer. To minimise the possibility of faulty network equipment between the VoIP server and the end-points, the VoIP server was re-patched so that it resided on the same switch as the end-points, hence bypassing the core switching infrastructure of the building. The second change addressed interfering network traffic. Disconnecting the nodes utilised by DRAPA from the departmental network was not feasible because the end-points are shared between DRAPA and students. Therefore, the distributed agents and action modules were extended to measure interfering traffic. The amount of interfering traffic is taken into account during the analysis of collected data. Any test cycle which is found to exhibit a significant amount of interfering traffic is discarded. Re-patching the VoIP server resulted in elimination of the scattered points beyond 56 concurrent sessions, proving that there was a problem with the switching infrastructure between the end-points and the Asterisk server.

```
root@almira:~# bing -z -P soekris1.ict.ru.ac.za
ug150.ict.ru.ac.za
BING soekris1.ict.ru.ac.za (146.231.121.207) and
ug150.ict.ru.ac.za (146.231.127.150)
44 and 108 data bytes (1024 bits)
ug150.ict.ru.ac.za:  73.143Mbps 0.014ms 0.013672us/bit
ug150.ict.ru.ac.za:  56.889Mbps 0.018ms 0.017578us/bit
ug150.ict.ru.ac.za:  60.235Mbps 0.017ms 0.016602us/bit
ug150.ict.ru.ac.za:  78.769Mbps 0.013ms 0.012695us/bit
ug150.ict.ru.ac.za:  93.091Mbps 0.011ms 0.010742us/bit
ug150.ict.ru.ac.za:  93.091Mbps 0.011ms 0.010742us/bit
ug150.ict.ru.ac.za:  102.400Mbps 0.010ms 0.009766us/bit
--- estimated link characteristics ---
host          bandwidth          ms
ug150.ict.ru.ac.za 102.400Mbps 0.215
```

Figure 3.13: `Bing` used to measure available bandwidth to the VoIP server

3.10 Summary

DRAPA facilitates the study of VoIP systems that comprise multiple end-points served by a finite number of VoIP Servers. A VoIP system includes SIP proxy servers and gateways to other types of network, for example the PSTN. The management of physical shared resources is addressed by DRAPA, minimising the cost of real-domain analysis when physical nodes are already available but shared with other users.

DRAPA has been compared to an industry standard simulator, OPNET, and found to collect, as one would expect, more realistic results. For example, DRAPA recorded miscellaneous bandwidth used by supporting IP protocols such as ARP. The availability of a real-world testbed as opposed to a simulated one opens the possibility of better results when compared to simulated studies. DRAPA can also be used in a commercial environment, where the performance of a VoIP system affects costs and revenue planning.

DRAPA has been formulated with flexibility as a key design consideration. The user customisation of the data collection agents and action modules gives DRAPA two important advantages. Firstly, DRAPA can be used to investigate multiple VoIP environments. Secondly, a single DRAPA testbed can be used by multiple users, where the time taken to switch from one user's work to another is negligible.

The next chapter describes the use of DRAPA to investigate the performance of security mechanisms implemented within an Asterisk based VoIP system to guarantee confidentiality, integrity and

authenticity to media streams.

Chapter 4

Performance of a secured Asterisk environment

4.1 Introduction

With respect to the quest to fulfil the research objectives stated in the first chapter, two tasks have been completed thus far. Firstly, appropriate security methods for an Asterisk-based VoIP environment were investigated and SRTP, SIAX2 and IPSec identified as suitable. The DRAPA system was then developed to support the performance analysis of a VoIP system that uses these security mechanisms. This chapter completes the set research objectives. Firstly, SRTP, SIAX2 and IPSec are made available into an Asterisk-based VoIP system. Then the extent to which the implemented security mechanisms affect the overall performance of the VoIP system is studied using DRAPA.

Section 4.2 describes the configuration of IAX2 and IPSec, and the implementation of SRTP into Asterisk. Section 4.3 introduces the experiments, a performance analysis of the security mechanisms within an Asterisk-based environment. The configuration of DRAPA and the implementation of data collection methods for the experiments are described in Section 4.4. Preliminary theory and expected results are discussed in Section 4.5. Finally, the results of the experiments are presented and discussed in Section 4.6.

4.2 Securing an Asterisk-based VoIP system

Mechanisms for securing VoIP media streams have been discussed in Section 2.3 in terms of their ability to provide sufficient confidentiality, integrity and Availability (CIA). Three security methods

were selected and implemented or configured for use in an Asterisk-based VoIP system, namely SIAX2, IPSec and SRTP. IPSec tools needed to be installed and configured before it was activated in the Linux kernel. SIAX2 was already available in the stable branch of Asterisk and simply needed to be configured. The configuration of SIAX2 and IPSec is explained in more detail in Section 4.2.1. At the time at which the secure testbed was built, a SRTP implementation for the Asterisk soft-switch was not available. Hence, the author implemented SRTP into Asterisk. The SRTP implementation is described in Section 4.2.2.

The next section describes the configuration of SIAX2 and IPSec.

4.2.1 Configuration of SIAX2 and IPSec

SIAX2 exists as a native security mechanism within Asterisk. To enable SIAX2, the option `encryption=aes128` is added to the Asterisk IAX2 configuration. This change is made on the VoIP server and end-points before the SIAX2 test is run.

IPSec (described in Section 2.3.1) is set up in two phases: firstly, a key-exchange mechanism is set up, and then policies which govern the IPSec stack within the Linux kernel are configured. To perform key exchanges, version 0.6.6 of *Racoon*, the IKE keying daemon [7] were installed on end-points and the VoIP server. Each installation of Racoon was configured with a preset key which is used by the end-points and VoIP server to authenticate one another. The key exchange was configured to use the IKE protocol and 160 bit HMAC-SHA1 was configured as the authentication algorithm. SHA1 can generate a larger fingerprint in comparison to MD5 (128 bit), which provides better authentication. Unlike SRTP and SIAX2, IPSec supports a range of cryptographic algorithms. The Blowfish cipher was chosen and configured to use its largest key size of 448 bits. The large key was chosen as it provides stronger encryption and allows a range of cryptographic strengths to be analysed in the experiments.

The second phase in configuring IPSec is to declare policies. Figure 4.1 shows a policy configured on one of the end-points. A similar policy was configured on the other end-points and a converse policy for each end-point was configured on the VoIP server. Each policy, from the point of view of an end-point, instructs the Linux kernel to intercept traffic to and from the VoIP server. Intercepted traffic is encapsulated within the ESP protocol in transport mode (described in Section 2.3.1).

4.2.2 Implementing SRTP within Asterisk

When initial performance testing was being conducted, a mechanism for securing media streams with SRTP in an Asterisk-based environment did not exist. Consequently, the author implemented SRTP

```
spdadd 146.231.127.174 146.231.123.45 any -P out ipsec
esp/transport/146.231.127.174-146.231.123.45/require;
spdadd 146.231.123.45 146.231.127.174 any -P in ipsec
esp/transport/146.231.123.45-146.231.127.174/require;

voip-server address:  146.231.123.45
end-point address:  146.231.127.174
```

Figure 4.1: End-point IPSec security policy.

into Asterisk as a proof-of-concept, and this implementation was used for initial performance testing. Later (October 2005) the Asterisk community implemented SRTP into Asterisk and started an official branch of the Asterisk source code specifically for SRTP support. The official branch was adopted by this project because it ensures it is up-to-date with the latest version of Asterisk in our testbed. Moreover, it also enables us to produce results against a globally accepted branch of SRTP secured Asterisk. The results of the experiments presented in Section 4.6 were generated from the community branch of SRTP implemented into Asterisk.

The two implementations, the one developed for this study and the Asterisk community's, are very similar in that they both make use of an SRTP library maintained by David McGrew (also co-author of the SRTP RFC [45]) from Cisco Systems. This section briefly describes the incorporation of the SRTP library into the Asterisk RTP stack. The implementation is described in three phases. Firstly the SRTP library is introduced. Secondly, affected areas of the Asterisk source code are identified. Lastly, the adjustments to Asterisk are discussed. A more in-depth discussion of this work, which includes problems encountered during the development appears in a paper by the author [19].

The sections which follow refer to the author's proof-of-concept implementation, unless otherwise stated as belonging to the Asterisk community's implementation.

The SRTP library

The SRTP library, named libSRTP [44] is licensed in the open source domain with the aim to promote the use of SRTP in various applications. The library supports all mandatory features defined in the SRTP Request for Comment [45] (RFC3711).

libSRTP addresses RTP traffic as *streams;* more than one concurrent stream is termed a *session*. Cryptographic options, keys and stream-addressing are configured through the declaration of policies. A policy is associated to a stream with a Synchronisation Source Identifier (SSRC), a field already

used in the RTP protocol. The SSRC value is used to match incoming or outgoing RTP packets with stream policies. The following policy attributes are specific to a stream:

- `cipher_type` represents the type of cipher that should be used for confidentiality;
- `cipher_key_len` represents the length of the key to be used in the cipher;
- `auth_type` denotes the authentication function to be used;
- `auth_key_len` holds the length of the authentication function key;
- `auth_tag_len` contains the length of the authentication tag;
- `sec_serv` is a flag representing security services to be applied.

A policy per stream is configured. One or more policies are grouped into a session. The streams within a session share these policy attributes:

- `ssrc_t ssrc` stores the synchronisation source identifier;
- `crypto_policy_t_rtp` points to the specific policy for RTP data;
- `crypto_policy_t_rtcp` points to the specific policy for RTCP data;
- `octet_t_key is` the cryptography key.

The `rtp` and `rtcp` attributes contain the specific policies for the real-time media and real-time control data respectively. `key` contains the cryptography key for encrypting or decrypting the RTP and/or RTCP data.

libSRTP exposes several functions. The first is `srtp_init(),` which is called to initialise the library. Then, to create a new SRTP session, `srtp_create()` is called and passed an SSRC value and a configured policy. Once a session is created, streams can be added to the session by invoking `srtp_add_stream()` and passing it the existing session SSRC value and a new stream policy. SRTP streams are de-allocated when `srtp_remove_stream()` and `srtp_dealloc()` are called and passed an SSRC value.

Once a session is created with one or more stream policies in place, RTP and RTCP data can be protected. The `srtp_protect()` function is responsible for the authentication and encryption of RTP and RTCP payloads. A payload is passed to `srtp_protect()`, along with the SSRC of the payload's stream. The function returns a pointer to the encrypted and authenticated payload. On the

receiving end of an RTP or RTCP stream, `srtp_unprotect()` is used to authenticate and decrypt the payload. Like the protect function, a payload is passed to `srtp_unprotect()` along with SSRC of the payload's stream. The function returns a pointer to the decrypted and authenticated payload. `srtp_unprotect()` will authenticate each packet and return an `err_status_auth_fail` enumerated code if the packet has been tampered with. If packet replay is detected, the `err_status_replay_fail()` code is returned. These functions and policy attributes are used within Asterisk to provide SRTP functionality.

Having described the basic functions of the SRTP library, the next section identifies parts of the Asterisk RTP stack which handle the creation of RTP streams, the transport of RTP media, and the closing down of RTP streams.

Identifying SRTP's place in Asterisk

To understand Asterisk's RTP architecture, the Asterisk source code was reviewed. Areas of the code relevant to integrating SRTP support into Asterisk are explained in this section. Asterisk performs all RTP processing in rtp.c. The following were found to be important functions:

- `ast_rtp_init()` initialises the RTP system. This function is called when Asterisk is started.
- `ast_rtp_new_with_bindaddr()` is called and passed the destination's IP address when creating a new RTP session. Usually the Session Initiation Protocol (SIP) and Service Discovery Protocol (SDP) systems in Asterisk are responsible for obtaining the destination's IP address and RTP ports. This function creates an RTP session and assigns it an SSRC value.
- `ast_rtp_stop()` is executed when an RTP session is not needed any more.
- `ast_rtp_reset()` sets all the default RTP session attributes to their initial values, effectively recreating the RTP session.
- `ast_rtp_raw_write()` is called when there is an RTP packet that needs to be sent to an RTP receiving node. This function is responsible for passing RTP packets to the computer's network stack.
- `ast_rtp_read()` the read function is called when the computer's network stack has received an RTP packet. The read function accepts RTP packets, checks for basic data corruption and passes their contents to Asterisk where a decoder converts RTP data into audio or another encoded format for retransmission.

These functions were marked as areas where libSRTP could be used to create or stop secure sessions and to protect or unprotect RTP streams. The modifications to enable SRTP support are described in the section which follows.

Implementing SRTP into Asterisk

After understanding the SRTP library and reviewing the Asterisk source code, the author implemented SRTP into Asterisk. Before the SRTP library can perform cryptographic procedures, a key needs to be supplied to both ends of a session. A static key was used, which is adequate for our study since we are only interested in the performance cost when media is streamed with SRTP and are not focusing on the key exchange. The community branch utilises sdescriptions, described in Section 2.4.2, to perform a key exchange within the SDP protocol.

Policies are configured when the RTP stack in Asterisk is initialised. `ast_rtp_init()` is used to initialise libSRTP when Asterisk is started and when the RTP system is reloaded. The policy is placed in `ast_rtp_new_with_bindaddr`(). Each SRTP session is created with the following attributes (the policy assignment code can be found in Appendix D):

- The SSRC value assigned by Asterisk is used to address the new SRTP session.
- A cryptography key is statically assigned.
- The cipher type is set to AES_128_ICM, currently the strongest cipher available to libSRTP.
- For authentication, the function is set to HMAC_SHA1.
- The policy is configured to provide confidentiality and authentication.

The `ast_rtp_stop()` function was amended so that an SRTP session is removed when the RTP session it is protecting is closed down. Likewise, `the ast_rtp_reset()` function was amended to recreate SRTP sessions when RTP sessions are recreated.

RTP packets are intercepted within `ast_rtp_raw_write`(). Before the RTP packet is sent to the network stack, its payload is moved to a new buffer which is passed to `srtp_protect`(), the protect function provided by libSRTP (the protect code can be found in Appendix E). The payload is encrypted and an authentication header is added. The buffer is effectively converted from an RTP payload into an SRTP payload. The new buffer is passed to the network stack where it is sent to its destination, for example another end-point.

On the receiving end of an SRTP stream, `ast_rtp_read()` was modified to unprotect incoming SRTP packets. When an SRTP packet is received, it is copied into a buffer which is passed to `srtp_unprotect()`, the unprotect function provided by libSRTP (the unprotect code can be found in Appendix F). This function converts the SRTP payload into an RTP payload and it checks the authentication header to ensure integrity. Finally, a pointer is addressed to the contents of the new buffer, which is used by Asterisk for further processing.

Having described the configuration and implementation of the security mechanisms into a Asterisk-based testbed, the experiments can be described. The experimental procedure, definition of terms and metrics used for performance measurement are introduced in the next section.

4.3 Experiments introduction

Once the secure testbed was built, the DRAPA nodes and parts of Asterisk were altered so that performance information could be collected during a series of tests. This section describes the type of data collected and defines the terms used in the experiments.

A collection of test cycles is simply referred to as a experiment. Five experiments were completed:

- Unprotected RTP;
- RTP protected by SRTP;
- Unprotected IAX2;
- IAX2 protected by SIAX2;
- RTP protected by IPSec.

A comparative analysis is done by comparing the data collected during the testing of a secure protocol to the data collected during the testing of the non-secure version of the protocol. For the three security mechanisms, RTP is protected by SRTP and IPSec, while IAX2 is protected by SIAX2. IPSec could protect RTP or IAX2, but for this study IPSec protects RTP. During each test, data was collected for between 1 session and the maximum number of sessions possible given the VoIP Server's CPU capacity. However, the data displayed in the plots presented in this chapter have been capped at 96 sessions for better readability. The length of time for which DRAPA held the concurrent sessions during a test cycle is 60 seconds. The data collected from each test has been divided by 60 and so it is presented as a per second average. The following resources were chosen to be monitored for each test:

- Bandwidth: bandwidth is collected as total bytes sent to and received from the VoIP server. This is converted into megabits per second.
- CPU time: The number of active CPU ticks on the VoIP server. A CPU tick is described further in Section 4.6.2.
- Encryption and decryption times: for the SRTP and SIAX2, the process of encrypting or decrypting each packet was timed. The maximum, minimum and average time taken for encryption and decryption was recorded.
- Latency: two latency measurements are employed. Firstly, before the concurrent sessions during a test are closed down, a measurement of the network latency is taken. Secondly, the time between received packets on the VoIP server for one of the concurrent monitored streams is measured. These methods are explained further in Section 4.4.3.
- Interfering data: due to the problems discussed in Section 3.9, a bandwidth counter which records the amount of external data received was added to the testbed. The interfering bandwidth counter is checked and a test is discarded if a significant amount of interfering traffic is present.

4.4 Configuration of the VoIP Server, end-points and DRAPA

The G711 codec [38] is used to encode and decode the media transported in these experiments. Other codecs, such as GSM [67], sacrifice CPU computation time in favour of saving bandwidth. The G711 codec does not attempt to optimise the bandwidth usage, and so it uses a significant amount of bandwidth (64kb/s) in comparison to GSM (13kb/s). During the validation of DRAPA (Section 3.8), the CPU resource was exhausted before the available bandwidth. Hence, for these experiments we chose the least CPU intensive codec, G711.

DRAPA's management of shared resources was relied upon for preliminary experimentation. However, to ensure the accuracy of the results, a computer laboratory was reserved for our sole use for the investigation, whose results are presented here. The laboratory machines were utilised as end-points which create sessions to one another via a VoIP server, placing the VoIP server under load. In total 44 end-points were available. During each test, the number of sessions was divided evenly over the available end-points. A maximum of six sessions, three outgoing and three incoming, were used per end-point, creating a total of 100 concurrent sessions at a maximum. The sections which follow discuss the configuration of both the end-points and the VoIP server. They also describe the configuration of the DRAPA modules and the data collection agents.

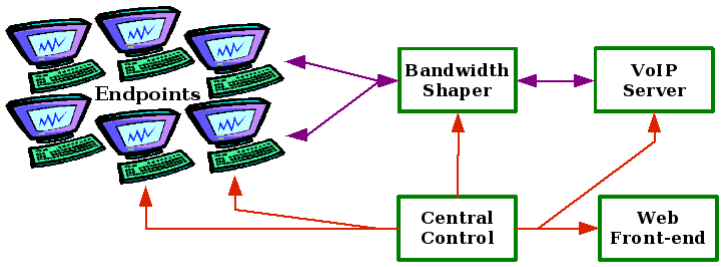


Figure 4.2: Physical layout of the DRAPA testbed.

Extension	Priority	Application
XXX	1	Answer
XXX	2	Set(_SIP_SRTP_SDES=1)
XXX	3	exten => _XXX,2,Dial(SIP/9${EXTEN}@ug${EXTEN}.ict.ru.ac.za)
XXX	4	Hangup

Table 4.1: VoIP server dialplan

4.4.1 Media routing

Two media routing strategies are possible after a VoIP session has been created between two end-points. The pair of end-points could route the session media directly between one another, or the session media could be routed via the server. The second strategy is used in the experiments described here, because we are interested in the load placed on the server by multiple sessions. In Asterisk, session routing is done in the *dialplan*. Table 4.1 shows a segment of the dialplan used for routing SRTP sessions. Session routing in Asterisk is triggered by extension numbers; the portion of the dialplan shown is triggered by an end-point dialling any three digit extension (an X denotes any digit). The flow of the dialplan is governed by the priority numbers shown in the second column of the table. The applications, displayed in the third column, are run iteratively according to the priority values. An end-point initiates a session to the VoIP server by dialling a three digit extension number. The session is answered by the server executing the `Answer` application priority at 1. Priority 2 enables SRTP protection for the outgoing channel. The third priority creates a second session to a destination end-point. The server uses the number dialled by the initiation end-point to route the received session to a second end-point. For example, if the server received an incoming session on extension `123`, it would create an SRTP session to the extension `9123` at the host `ug123.ict.ru.ac.za`. The VoIP server simply unprotects incoming media from one end-point, protects it again and sends it to the other, implementing a hop-by-hop security mechanism. For each session, four media streams, or channels, are handled by the VoIP server, two per end-point. Table 4.2 shows the dialplan on each end-point (each end-point runs a copy of Asterisk). The application at priority 2, `saynumber`,

Extension	Priority	Application
9XXX	1	Answer
9XXX	2	SayNumber(123456789)
9XXX	3	Goto(local,${EXTEN},2)

Table 4.2: End-point dialplan

plays a series of audio files (one for every number, one to nine). The `saynumber` application is repeatedly executed by the `goto` application at priority 3, thereby providing a constant stream of media during each test. This design places the load required to generate media onto the end-points, keeping unnecessary load off the server.

4.4.2 Per-packet timing of cryptographic logic

The analysis code was placed into the copy of Asterisk running on the VoIP server. The code allows the testbed to collect data from the server while a test is in progress. What follows is the code segment which measures the delay incurred by the SRTP cryptographic operation, which was added to the incoming RTP stack.

```
// Get the starting time
gettimeofday(&beforedec,&tz);
// Perform the cryptographic operation
for (i = 0; i < 2; i++) {
  srtp_hdr_t *header = buf;
  res = srtp_unprotect(srtp->session, buf, len);
  if (res != err_status_no_ctx)
    break;
  if (srtp->cb && srtp->cb->no_ctx) {
    if (srtp->cb->no_ctx(srtp->rtp, ntohl(header->ssrc), srtp->data) <
    }
  } else {
    break;
  }
}
...
// Get the finishing time
gettimeofday(&afterdec,&tz);
// Accumulate
```

```
curdec = (int) (afterdec.tv_usec - beforedec.tv_usec);
totdec += curdec; // Total number of unprotected packets
countdec++;
if (ticktockde == 500) { // log the results every 500th packet
  ast_log(LOG_EVENT, "SRTP Unprotect Cur/Min/Max/Avg/Tot/Cnt:
          *%d*%d*%d*%ld*%ld*%ld*%d*\n",
          curdec, mindec, maxdec,
          totdec / countdec, totdec, countdec,waitqueuedec);
  if (curdec > maxdec)
    maxdec = curdec;
  if (curdec < mindec)
    mindec = curdec;
  ticktockde = 0;
}
ticktockde++;
```

The timing code was used four times, to measure the time taken to protect and unprotect packets when securing media for the SRTP and SIAX2 protocols. This routine calculates the minimum, maximum and average time of each security operation in microseconds.

The SRTP and SIAX2 protect and unprotect functions are implemented within areas of Asterisk which are called by multiple threads. Therefore, the memory used to store the timing calculations may be accessed by more than one thread at the same time. To avoid data being corrupted, a mutex is used to block parallel processes trying to access and update the same value simultaneously. This method of data management raises the concern of the Observer's Paradox as it can skew the results of a test. By blocking access to a shared resource, we risk creating a queue of waiting processes. A queue of waiting process significantly affects the delivery time of media packets, thus affecting the results of the test. To ensure that this situation does not arise, an integer, `waitqueue` is used to monitor the number of blocked processes. A test is discarded if `waitqueue` is ever found to be greater than one.

The SRTP and SIAX2 implementations differ from IPSec in that they are integrated into Asterisk, whereas IPSec provides transparent protection within the TCP/IP layer. A clear boundary exists between normal logic and added security logic in the Asterisk code. This boundary allows us to place timing logic into areas of the Asterisk code which have been augmented with security operations. It is more difficult to time the logic used to secure real-time multimedia with IPSec and to obtain results comparable with SRTP and SIAX2. Hence, the per-packet encryption and decryption times

were only collected and analysed for the SRTP and SIAX2 protocols.[1]

4.4.3 Latency measurements

Two methods of latency testing were employed. The first simply uses the UNIX *ping* command [39] which utilises a 64 byte packet ICMP [57] to find the minimum, mean, maximum and standard deviation in latency of the network. Ping uses the ICMP protocol's ECHO REQUEST/ECHO RESPONSE data-grams to measure round-trip times between the server and one of the end-points. The measurement is taken before all the sessions are closed down. The aim of this test is to measure any added latency of the network in response to the network load created by the current number of sessions.

Measuring the latency of the network allows us to check that bandwidth has not been consumed in its entirety. However, the focus of this study is the performance of secured real-time media streams and so a second latency measurement was added. The second measurement times the gap between each received media packet on the server. The delay between each packet indicates latency of the network combined with latency within the VoIP system, and so, is referred to as *in-band* latency. The in-band latency should remain stable unless a resource is exhausted, which would result in a difference in in-band latency. What follows is a code segment listing the logic added to Asterisk to measure the in-band latency for RTP.

```
if (monitoredssrc == rtp->ssrc) { // Wait for the monitored stream
  waitqueue++;
  while(!ast_mutex_trylock(&statlock));
  if (nextseqno == rtp->rxseqno) {
    gettimeofday(&after,&tz); // Get the finishing time
    totpack++;
    microseconds = microseconds + after.tv_usec - before.tv_usec;
  }
  if (rtp->rxseqno > nextseqno) { // Detect dropped packets
    dropped = dropped + (rtp->rxseqno - nextseqno);
  }
  nextseqno = rtp->rxseqno;
  nextseqno++;
  // Get the starting time for the next incoming packet
```

[1] This, naturally, does not prevent the collection of data relating to the impact of IPSec on bandwidth, server CPU and network latency of media streams

```
    gettimeofday(&before,&tz);
    waitqueue--;
    ast_mutex_unlock(&statlock);
  }
  if (ticktock == 500) { // Log results every 500th packet
     ast_log(LOG_EVENT, "::Monitored channel:: ,totlat,totpack,totdrop
              *%f*%f*%f*\n",microseconds,totpack,dropped);
     ticktock = 0;
  }
  ticktock++;
```

During a test, the first stream created is used to perform the in-band latency measurement and is referred to as the *monitored channel*. The monitored channel is identified by the SSRC value of the stream. For every packet received by the RTP stack, the SSRC value is checked and a timer, `after`, is recorded if the packet belongs to the monitored channel. The time in microseconds between each received packet is added to a total, `microseconds,` and a packet counter, `totpack`, is incremented. A start time, `before`, is recorded so that the time for the next incoming packet can be recorded. The order of packets is identified by each packet sequence number. `nextseqno` is always set to the sequence number of the next packet to be received. If the sequence number of a received packet is greater than `nextseqno`, the logic assumes that packets have been lost and increments `dropped` to record the number of dropped packets. (This assumption is realistic within a LAN.) Like the cryptographic timing code, a mutex is used to ensure that data is not modified by more than one process. The `waitqueue` counter is used to report if a queue of blocked processes has resulted from the presence of the mutex. Lastly, the following values are logged every 500th packet: the total time in microseconds, the total number of received packets and the total number of dropped packets. Note that these values are logged every 500th general packet and not every 500th monitored packet to correlate the values associated with the cryptographic logging.

Having amended Asterisk with monitoring code, the DRAPA modules were configured to enable the tests and collect data. The DRAPA module configuration is described in the next section.

4.4.4 DRAPA Module configuration

Five action modules were created, one for each test: `rtp`, `srtp`, `iax2`, `siax2` and `ipsec`. The DRAPA modules differ in terms of the logic used to create and close down sessions. However, the logic used to collect data was kept as similar as possible to ensure that the results could be easily compared. The structure of the test environment is illustrated in figure 4.2. A test entails the

creation of x number of sessions. Then, data measuring the security performance of each session is collected. Upon initialisation, the TMS selects a test based on the distribution of tests already performed so that an even distribution of tests is maintained for each test. The selected number of sessions determines the number of laboratory machines instructed to create a single session to the server. This achieves a realistic real-world scenario where many VoIP end-points are communicating via a central soft-switch. The load on the soft-switch is held for 60 seconds. During this time, data is collected from three points: 1) the timing code placed into Asterisk, 2) the bandwidth monitor on the bandwidth shaper node and 3) the CPU usage monitored on the VoIP server, maintained directly by the operating system. Before instructing the used end-points to close their media streams, the data generated at these three points was collected and saved in a MySQL database [49].

4.5 Preliminary theory and expected results

Having explained the configuration of the testbed, the experimental results can be discussed. This section describes the preliminary theory behind the experiments and presents hypotheses on their expected outcomes.

4.5.1 Bandwidth

We expect bandwidth usage to increase when any security method is utilised. In particular, we expect IPSec to cause the greatest bandwidth increase due to its method of encapsulating packets with the Encapsulating Security Payload (ESP) [25, 4]. By encapsulating a packet within a second protocol, we are adding a second set of headers. (This is analogous to placing a postcard into an envelope when the postcard could be addressed and sent without the corresponding envelope, but at the risk of being read by someone other than the intended recipient. A postcard in an envelope increases the bulk of the overall package.)

SRTP and SIAX2 protect the data at the application layer, in contrast to IPSec. The payload of a real-time media packet is protected before being encapsulated into an IP packet. The protection process merely alters the contents of the media packet so that the previously unprotected media is encrypted, and an authentication header is added. This method of protection is preferred in general as the size of the overall packet should exhibit a smaller increase.

4.5.2 CPU Usage

We expect the CPU utilisation of the VoIP Server to increase when any of the three security methods are applied. The overhead in CPU usage is expected to differ for each security method due to the use of different cryptographic algorithms and key sizes. In the experiments IPSec is configured to use the Blowfish cryptographic algorithm [65] with a 448 bit key, while SRTP and SIAX2 were configured to use the AES cryptographic algorithm [24] with a 128 bit key. Both Blowfish and AES are block cryptographic algorithms. Cryptographic studies show that the Blowfish cipher performs its operations faster than the AES cipher [32]. However, in these experiments Blowfish is used to protect an entire IP packet, while AES protects the payload only of RTP and IAX2 packets. Thus it is expected that Blowfish will consume more CPU time to perform its operations.

4.5.3 Encryption and Decryption times

SRTP and SIAX2 utilise the AES cipher to encrypt the media payload. Comparing SRTP and IAX2, we expect the average time taken for each protocol to protect a packet to be similar. If there is a difference, we expect it to be linear in relation to the number of concurrent sessions. (Encryption and decryption times were not collected for IPSec experiments.)

4.5.4 Latency

In both methods, the variation of latency is expected to remain constant in all tests, unless a resource becomes scarce. The saturation of a resource should see an increase in latency due to real-time media payloads queueing up to be processed. In identifying a potential bottleneck, preliminary tests were run on the testbed and bandwidth and CPU data was collected. The testbed was configured to use the GSM codec which requires 57.25Kbs of Ethernet bandwidth [50]. One hundred concurrent sessions were created, which generated 5.59Mb/s of bandwidth. Therefore, there should be no increase in latency due to the high availability of bandwidth (100Mb/s). To investigate the availability of CPU time, IPSec was configured to secure the GSM Codec. The availability of free CPU time amounted to 33 ticks per second at 95 sessions encrypted with IPSec. Therefore, sufficient CPU time is available too. However, other security methods could require more CPU time or could introduce a shared resource, such as a cryptographic engine. A shared cryptographic engine under significant load would result in a queue of packets waiting to be processed and this would increase latency.

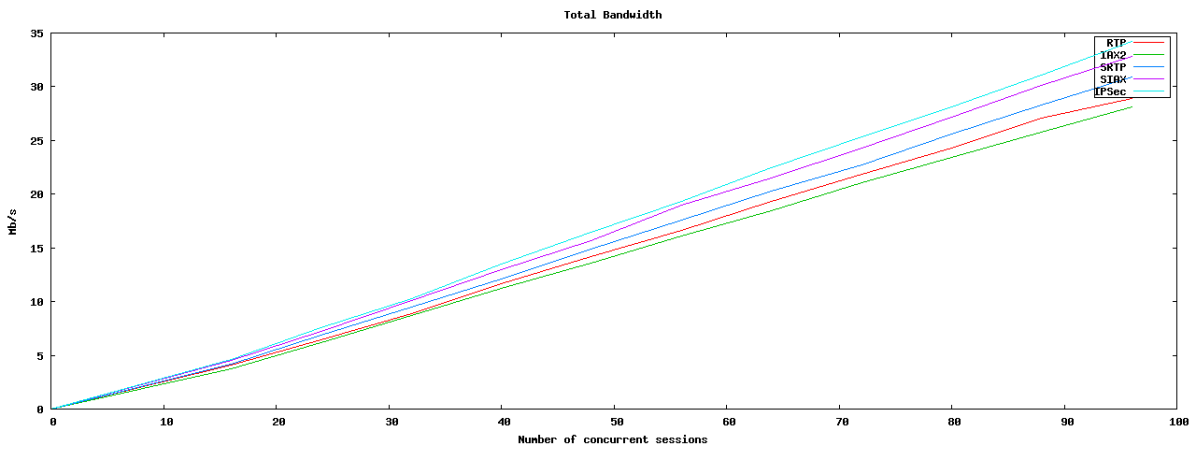


Figure 4.3: Total bandwidth consumed

4.6 Results and discussion

Before analysis was undertaken, the data collected was processed to produce the graphs shown in this section. This process aggregates the results to reveal the trend for each secured or untouched transport method. The aggregated data was used to generate two categories of graphs: the first are *comparison* graphs and the second are *overhead* graphs. The comparison graphs plot resource usages for all tests performed. The overhead graphs plot the difference in the resource usage between the secured and unsecured version of each transport protocol. This section introduces and explains each graph in terms of metrics and interesting anomalies. For all the graphs, the x axis shows the increase in the number of sessions made from end-points to the VoIP server.

4.6.1 Bandwidth

The graph in figure 4.3 plots the total bandwidth consumed for each test in megabits per second. One should note that the total bandwidth used does not directly reflect the bandwidth 'on wire' used by each security mechanism or media transport protocol. For every session, the VoIP server handled four media streams: two bi-directional streams to each end-point. Therefore, the bandwidth generated by a single stream for any of the tests is a fourth of what is plotted in figure 4.3.

The initial comparison confirms the expectations described in Section 4.5, namely that SRTP and SIAX2 require more bandwidth than the unprotected RTP and IAX2 protocols. The security mechanisms can be ranked in terms of total bandwidth consumption, where IPSec requires the most bandwidth, followed by SIAX2 and then SRTP. However, this ranking is dependent on the bandwidth

Security Mechanism	Ethernet bandwidth overhead incurred
IPSec	16%
SIAX2	17%
SRTP	5%

Table 4.3: Percentage overhead in bandwidth

usage of the unprotected protocol. For example, RTP requires 4% more bandwidth than IAX2. A true reflection of the bandwidth overhead induced by each security mechanism is shown in figure 4.4, which plots the difference between the protected and unprotected version of each protocol. Each security mechanism added a consistent bandwidth overhead, and the percentage of each overhead is shown in Table 4.3. Note that the percentage overheads in Table 4.3 were calculated with reference to the original Ethernet packet. Theoretical percentages calculated with reference to the payload of a packet are higher, because the IP and Ethernet headers are not taken into account.

The larger overhead of IPSec and SIAX2 are due to their methods of protection. IPSec, utilising the ESP protocol, adds a security parameter index (4 bytes), sequence number (4 bytes), authentication header (20 bytes) and a variable length of padding (0 - 255) to the original payload [25]. Therefore, the total overhead ESP adds to a packet is between 28 and 283 bytes, depending on the amount of padding needed. In the case of these experiments, DRAPA revealed an average overhead of 30 bytes added to the original 192 byte RTP packet by the ESP protocol.

SIAX2 used the AES cipher and encrypted each IAX2 payload in blocks of 16 bytes. The SIAX2 source code was reviewed to evaluate the theoretical bandwidth overhead incurred. The original size of an IAX2 payload using the G711 codec is 162 bytes. To make the payload consist of even 12 blocks of 16 bytes, 30 bytes of padding is added. The total size of the payload becomes 194 bytes which equates to a 19% overhead in relation to the original IAX2 payload size. The data collected by DRAPA revealed a 30 byte overhead too. In relation to the size of an IAX2 packet encapsulated within IP and Ethernet headers, the overhead is 17%.

Like SIAX2, SRTP uses the AES cryptographic algorithm but also includes a HMAC-SHA1 80 bit authentication header. The authentication header adds 10 bytes of overhead to each packet, which is originally 172 bytes in size. The total secured payload, 182 bytes, brings about a 6% overhead. DRAPA, measuring the Ethernet size of each packet, found the bandwidth used per SRTP packet to be 202 bytes, and bandwidth per RTP packet to be 192 bytes. DRAPA's data confirms the presence of the 10 byte authentication header and revealed a 5% bandwidth increase in relation to the size of an RTP packet encapsulated within IP and Ethernet headers. SRTP proved to be the most bandwidth efficient security mechanism.

The computational cost of each security method is discussed next.

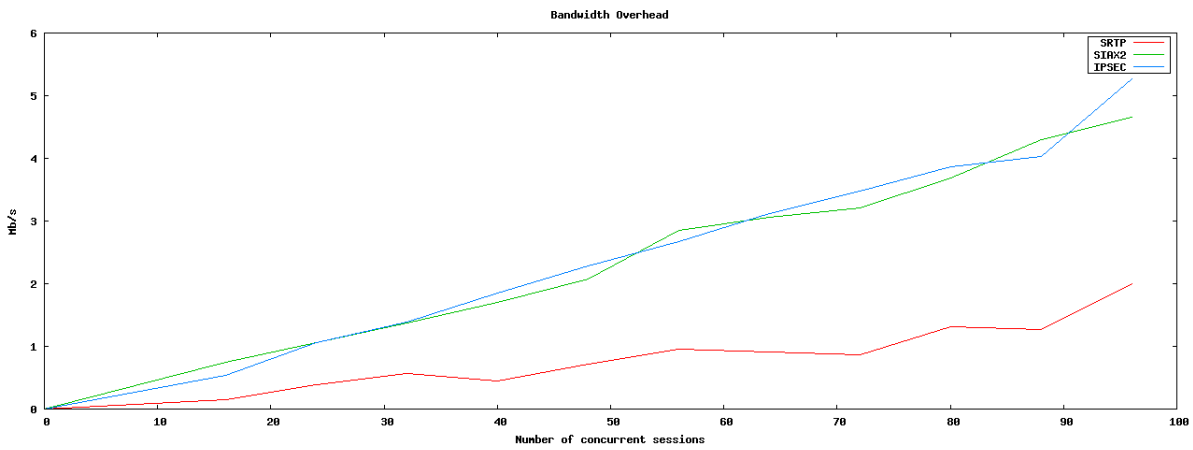


Figure 4.4: Bandwidth Overhead incurred by each security method

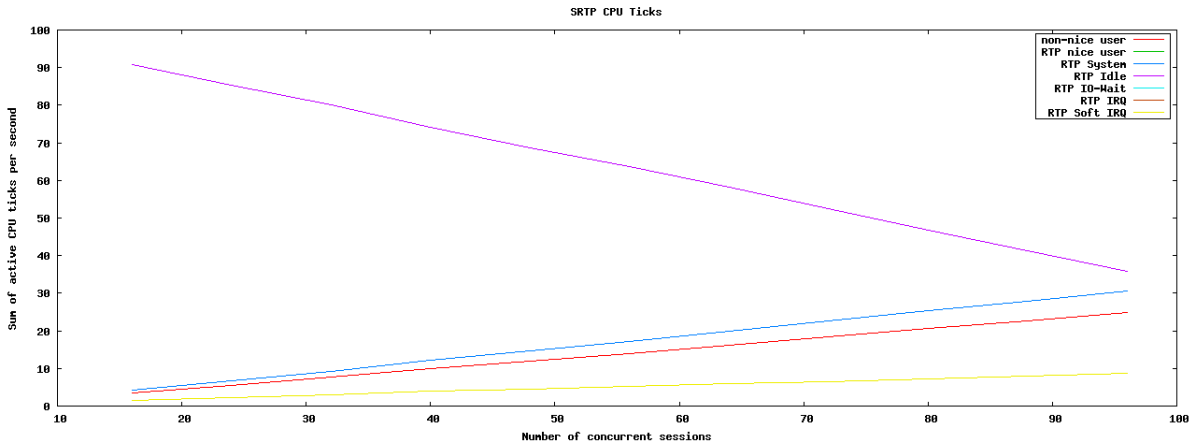


Figure 4.5: CPU usage for SRTP

4.6.2 CPU Usage

The CPU utilisation on a Linux machine is grouped into seven types: `non-nice`, `user`, `nice-user`, `system`, `idle`, `io-wait`, `irq` and `soft-irq`. From these types, one or more must be chosen as a metric to measure the CPU utilisation for our experiments. To make this decision, the SRTP test was plotted in figure 4.5. The unit used to measure CPU time spent on each of the seven states just listed is called *tick*. A tick is actually an interrupt generated at a period of $10ms$ by the Programmable Interval Timer (PIT) on the x86 architecture [2, 51]. The Linux kernel increments counters to log the number of CPU ticks spent on each of the seven states. The graph in figure 4.5 clearly plots a decrease in `idle` ticks and a converse increase in `system` ticks. However, there is also a less prominent increase in `irq` and `soft-irq` ticks which should be taken into account. So to take all active CPU

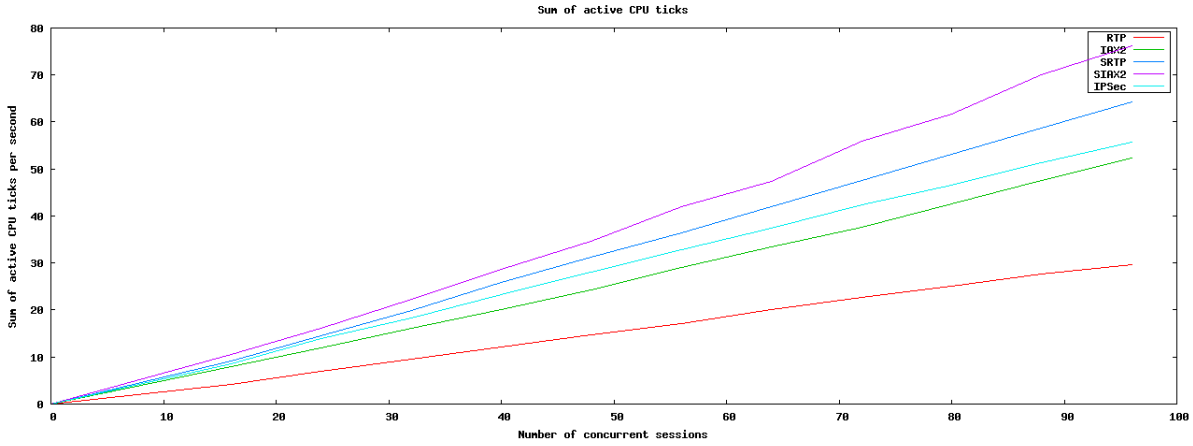


Figure 4.6: Comparison of active CPU ticks

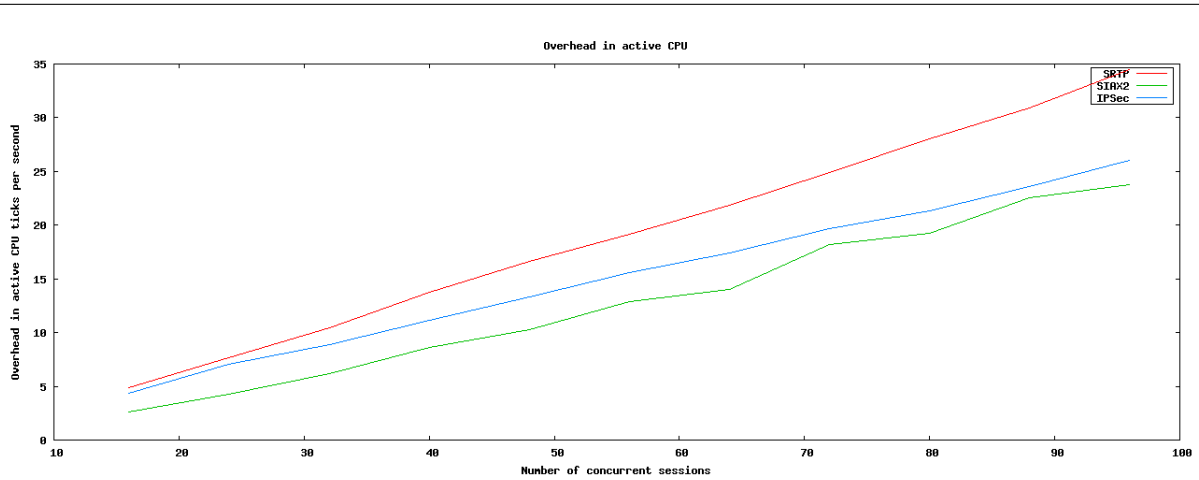


Figure 4.7: Overhead in active CPU ticks

Security Mechanism	CPU overhead incurred
SRTP	111%
SIAX2	46%
IPSec	78%

Table 4.4: Percentage overhead in CPU ticks relative to the CPU usage of the unprotected transport protocols

activity into account, the comparative and overhead CPU utilisation graphs are plotted using the sum of `user`, `nice-user`, `system`, `io-wait` `irq` and `soft-irq`.

The total number of active CPU ticks are plotted for all the protocols under study in figure 4.6. (Note that CPU usage takes into account all the processing on the server and not just ticks used by Asterisk. This provides a preferred measure of the load placed on the VoIP server as a whole.) The first phenomenon to observe is the large difference in required ticks for unprotected RTP and IAX2: RTP is 70% more efficient than IAX2. The large difference can be attributed to the design of each protocol. RTP uses the RTCP protocol and SIP to perform out of band stream control and signalling. Therefore, Asterisk is able to route RTP packets from one end-point to another without opening the RTP payload and performing any inspection. On the other hand, IAX2 transports its stream, stream control and signalling within a single protocol. This forces Asterisk to inspect every IAX2 packet, resulting in a higher computational cost.

The percentages shown in Table 4.4 are calculated in relation to the CPU usage of unprotected RTP and IAX2. SRTP is seen to incur a 111% overhead due to the very efficient RTP routing just described. This is not a true reflection of the computational performance of SRTP as, like IAX2, Asterisk is requires to inspect each SRTP packet during the unprotection and protection stage in the RTP stack. The 111% overhead is due to the loss of the RTP routing efficiency through security. Section 4.6.4 provides a better measure of the computational overhead incurred by SRTP's cryptographic operations through timing the cryptographic logic directly.

Unlike the SRTP measurement, the SIAX2 computational overhead of 46% is more accurate. The computational overhead of IPSec, 78%, is also accurate as the "transparent" security provided by IPSec allowed the RTP routing efficiency to still take place. The ranking of security methods with regard to computational overhead is seen in figure 4.7. SIAX2 incurs the least overhead, followed by IPSec and then SRTP. This rank is based on the premise that the routing efficiency of RTP can be utilised, which is not always true. For example, if the VoIP server were required to transcode incoming frames into another audio format, the computational cost of RTP would increase, hence the performance cost in using SRTP would decrease.

Having analysed bandwidth and computational costs, the latency measurements are discussed next.

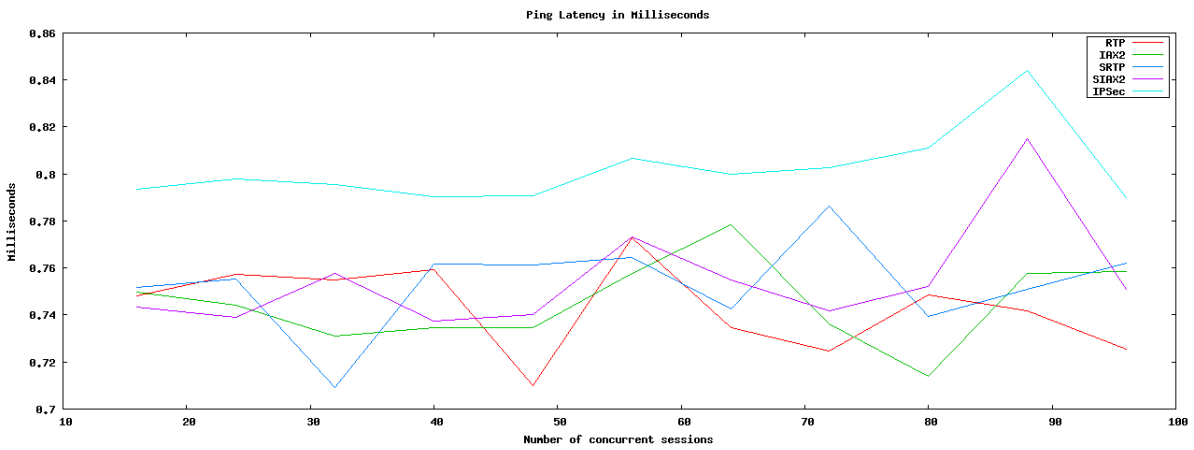


Figure 4.8: Ping Latency

4.6.3 Latency

Two types of latency were measured. The graph in figure 4.8 plots the ICMP echo round trip time for each transport method as recorded using the UNIX ping command. The graph in figure 4.9 plots the time between received packets on the server (discussed in Section 4.4.3). The plots in both graphs are linear in relation to the number of concurrent sessions when the scale of the graphs is taken into account. The ping latency for each test is no more than 1 millisecond, which means there was no bottleneck on the network during any of the tests. Congestion on the network would result in larger echo round trip times and this would cause the plots in the ping graph to increase in relation to the number of concurrent sessions.

The interval between two consecutive packets received by the VoIP server (figure 4.9) for all tests is on average 20 milliseconds. This is a large value in comparison to the speed at which the network is capable of transporting packets: the latency of the network can be measured at half the ping latency, so it is around 0.5 milliseconds. However, 20 milliseconds in between the packet arrival times is correct when considering the mechanism used to encode G711 audio into frames, encapsulating frames into packets and then passing the packets to the network stack. The tests in these experiments used the G711 codec which samples audio in 20 millisecond intervals. Therefore, Asterisk was able to transmit an audio sample every 20 milliseconds.

For each security mechanism, the overhead in-band latency is plotted in figure 4.10. From the scale of the graph we can conclude that the variation in latency for any of the three security methods is no more than 0.1ms. Therefore, the latency and jitter added by the security mechanisms is negligible.

The last measurement, the timing of cryptographic operations, is discussed next.

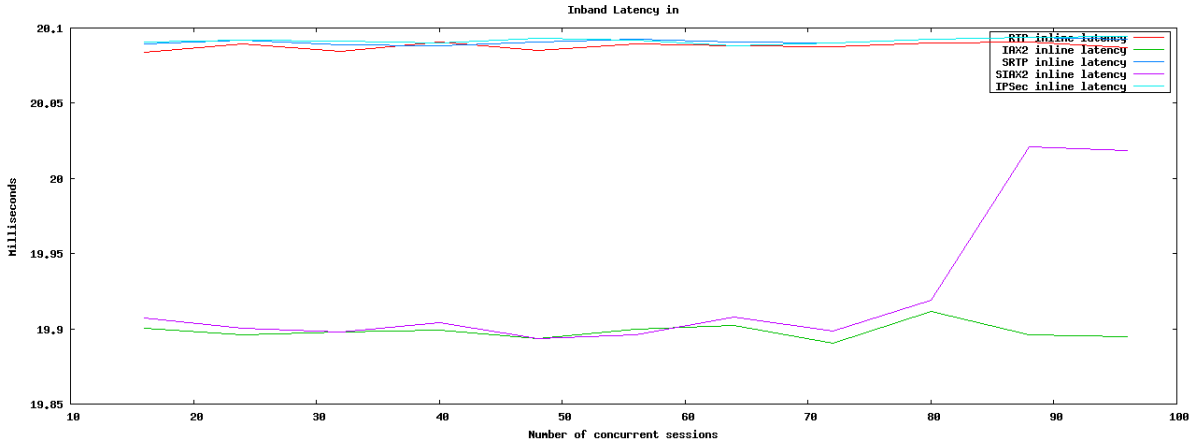


Figure 4.9: In-band latency

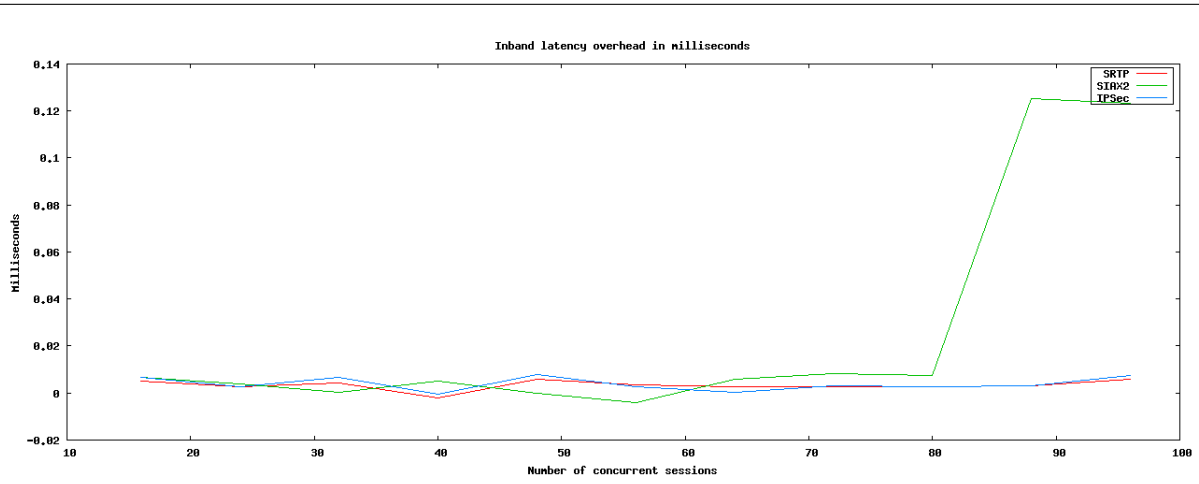


Figure 4.10: Overhead of in-band latency

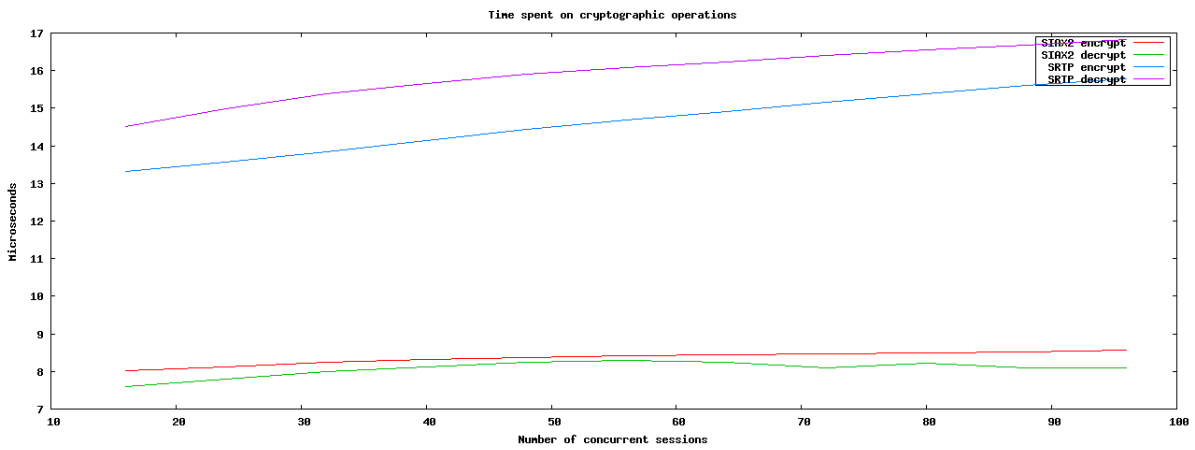


Figure 4.11: Cryptographic Operations

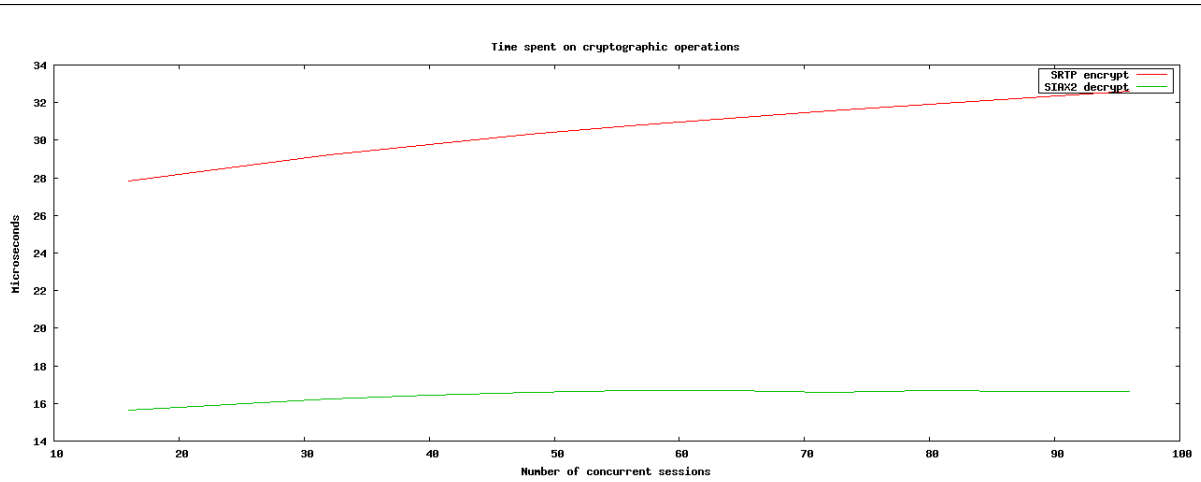


Figure 4.12: Total time spent on cryptography operations

4.6.4 Cryptographic overhead

This section considers the SIAX2 and SRTP protocols only (discussed in Section 4.4.2). The time taken to perform the protect and unprotect cryptographic operations on a single packet for SRTP and SIAX2 is plotted in figure 4.11 against the number of concurrent sessions.

Figure 4.12 displays the total time, an aggregation of the protection and unprotection times, in microseconds for SRTP and SIAX2 packets. In both cases the difference in time taken to protect and unprotect data was negligible. The graph confirms a constant overhead: SRTP and SIAX2 exhibit a variation of no more than $4\mu s$. SRTP requires an average of $31\mu s$ and SAIX2 an average of $16\mu s$ to perform protection and unprotection functions. SIAX2 completes its cryptographic operations in a shorter time than SRTP, by an average of $14\mu s$. The cryptographic operations for SRTP and SIAX2

add 0.03 milliseconds and 0.02 milliseconds of latency per packet respectively, which is considered negligible.

4.7 Summary

The first part of this chapter described the implementation of three security mechanisms into an Asterisk-base VoIP system. SIAX2 is natively implemented within Asterisk and simply needed to be enabled. IPSec is implemented at an operating system level and required the configuration of policies and, in these experiments, pre-shared keys. A method of using the SRTP protocol to secure the testbed was initially unavailable, so the author implemented SRTP functionality into Asterisk. Like IPSec, SRTP is also configured through policies.

The second part of this chapter explained the configuration of a DRAPA testbed for analysing the performance impact of the three security methods. A series of experiments were conducted to measure bandwidth usage, CPU utilisation, latency induced and cryptographic overheads when securing media streams with SRTP, SIAX2 and IPSec. From the measurements, Bandwidth and CPU resources appear to be affected the most by applying security.

These results can be used to plan the implementation of security methods into specific environments. For example, where bandwidth is limited more than CPU, the SRTP method of security should be used. Where CPU is limited more than bandwidth, for example in an embedded system, SIAX2 should be used.

Chapter 5

Conclusion

5.1 Introduction

At the beginning of this study, the following two research objectives were set:

1. To investigate appropriate methods of securing conversation streams in an Asterisk-based VoIP system, and implement them as a proof-of-concept system.
2. To examine the performance cost, in terms of CPU, bandwidth and overall quality, of the selected security additions to an Asterisk-based VoIP system. This was achieved through the use of a specially designed software suite that we called DRAPA.

In this section, each objective is discussed in terms of how it was achieved

To address the first objective, three security methods are discussed in terms of their provision of CIA in Section 2.3. IPSec was selected as a traditional method of data security. SRTP and SIAX2 were selected as they are designed with the aim of protecting real-time data. The implementation of the secured system entailed the configuration of SIAX2 and the installation and configuration of IPSec. At the time, there was no available method for protecting Asterisk-based VoIP systems with SRTP. To provide SRTP protection for Asterisk, the author implemented SRTP functionality into Asterisk as a proof-of-concept. This implementation was successful and was used for initial performance evaluations of SRTP.

To address the second objective, the DRAPA software was designed and implemented. During the design stage, the scope of DRAPA was extended, so that it is able to analyse any aspect of a VoIP

system. DRAPA was used to perform a qualitative analysis which found that bandwidth and CPU were primarily affected by the addition of security to our environment. Security methods were ranked in terms of their impact on these two resources. It was found that, where bandwidth is limited, SRTP should be used, as it adds the lowest bandwidth overhead. Where CPU is limited and there is a reason for Asterisk to inspect an audio stream (for example, transcoding or recording the stream), SIAX2 should be used as it incurs the smallest overhead in CPU usage. However, if Asterisk does not need to inspect the audio stream, IPSec should be used to protect the RTP transport protocol, as this combination used the least amount of CPU overall.

5.2 Contributions of this study

This study has made three main contributions.

The first is the development of DRAPA, a suite of flexible testbed software which enables the performance analysis of a physically implemented VoIP system, as opposed to a simulated one. The flexibility of DRAPA allows a user to measure the specific aspects of a VoIP system that are of interest. For example, DRAPA could be used to investigate the performance of codecs instead of security methods. Furthermore, the results produced by DRAPA have been validated through comparison with the A&S simulator described in the literature.

The second contribution is the proof-of-concept implementation of SRTP within an Asterisk-based environment. Discussions, by the author, of this implementation on the Asterisk development mailing list led to the Asterisk community implementing SRTP functionality into an official branch of Asterisk (October 2005).

The third contribution is the performance analysis of SRTP, SIAX2 and IPSec within an Asterisk-based environment. The results of this study provide a baseline against which the performance of future security mechanisms can be compared. The results can also be used to evaluate the cost/benefit ratio of implementing security within VoIP systems.

5.3 Recommendations for future research

5.3.1 Extensions to DRAPA

Three extensions to DRAPA are suggested as possible future work.

Currently, the scope of DRAPA does not include the processing of data after it has been stored in a database. The first extension should address the processing and presentation of the collected data. The current design of the action modules could be extended to include data processing functions to describe the actions needed to extract, aggregate and present collected data. However, it may be advantageous if the data processing functions exist in separate modules. This would allow a user to prepare a single data processing module which can then be utilised by many action modules. The user would specify an appropriate data processing module within each action module.

The second extension could extend the web interface to facilitate the control and configuration of the testbed. To better support a multi-user environment, the DRAPA web front-end should be converted from a page of static information into a portal. The portal could support user profiles, which would enable a DRAPA user to authenticate and gain access to their action modules online. The portal could manage a booking schedule for each user to reserve time for their experiments. Electronic mail notifications could be added, where DRAPA notifies users when the reserved testing session is completed.

The third extension to DRAPA would be the addition of an interactive testbed mode. Currently DRAPA supports two modes, one which restricts the load to a single session per end-point, and another which allows more than one session per end-point. A third mode could enable the user to adjust, via the web-interface, the number of concurrent sessions within the testbed and, hence, receive real-time feedback. The real-time interaction of this mode would give a user better control of the experiment and could be used, for example, to demonstrate VoIP system design in real-time to a class of students.

5.3.2 Security extensions

Of the security methods used in this study, IPSec is the only one which provides complete protection to communication streams including signalling. IPSec achieves this through its location in the network stack, rather than within a real-time protocol.

SIAX2 does provide confidentiality to both signalling and media transport but it is limited by the weak cryptographic key used. The SIAX2 key is created through the process of the VoIP server requesting an end-point to compute an MD5 of their authentication password and a nonce value (a random number used to protect against replay or statistical attacks). Should an eavesdropper capture both the nonce value and the computed hash during the authentication of the end-point, an offline brute-force attack would allow the end-point's password to be discovered. Thereafter, the eavesdropper could calculate their own copy of the cryptographic key. The SIAX2 protocol also neglects to provide integrity checking. Future work could entail the development of a more secure key exchange. In

addition, mechanisms for authentication, availability and integrity checking could be added to the SIAX2 protocol. These improvements would enable SIAX2 to provide complete protection for real-time communication.

The implementation of SRTP within Asterisk currently provides confidentiality, authentication, availability and integrity protection for media transport only and as a result, the cryptography key used by SRTP is exchanged in the clear. We propose two strategies to address this problem. Firstly, IPSec could be employed to secure the SIP exchange only. Thereafter, SRTP could secure the media session. This method would provide complete protection of the session initiation phase, communication stage and closing down of the session. However, it would also introduce an extra overhead as a result of the instantiation of an IPSec security association [59]. The second strategy would be to implement one of the secure key exchange methods discussed in Section 2.4. For example, MIKey would exchange a key in a secure manner, which can then be used by SRTP. The second strategy would incur a smaller performance cost during the session initiation process, but would naturally leave the SIP exchange in the clear. This would allow an eavesdropper to ascertain session information, such as the caller and callee URIs, but would provide sufficient protection to the media stream.

5.3.3 Future performance experiments

The experiments in this study analyse the performance cost of three security methods. Currently the cryptographic and authentication algorithms available for SRTP and SIAX2 are limited. By contrast, IPSec supports a rich set of cryptographic and authentication algorithms. IPSec also supports different encapsulation methods and modes of use. Different configurations of IPSec could be analysed with DRAPA and compared to the results in this study. For example Blowfish, a strong cryptographic cipher, was utilised by IPSec in this study. A future experiment could re-examine IPSec when configured to use a weaker cryptographic cipher. This configuration may result in a lower computational cost and a change in bandwidth usage.

5.4 Conclusion

The ability to provide voice services on an IP network with software-based switches has accelerated the development of voice and telephony services [36]. An important consideration in these services is quality, which makes the addition of mechanisms to protect confidentiality, integrity and authentication less appealing, because of their cost on performance. This study has provided realistic measures of this cost and concluded that the addition of CIA to real-time communication is viable.

References

[1] I. Abad. Secure Mobile VoIP. Master's thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, June 2003.

[2] L. Abeni, A. Goel, C. Krasic, J. Snow, and J. Walpole. A measurement-based analysis of the real-time performance of the linux kernel. Technical report, Department of Computer Science and Engineering, Oregon Graduate Institute, Portland, 2002.

[3] Flemming Andreasen, Mark Baugher, and Dan Wing. Session Description Protocol Security Descriptions for Media Streams. Technical report, Cisco Systems, Inc, September 2005.

[4] Zahid Anwar, William Yurcik, Ralph E. Johnson, Munawar Hafiz, and Roy H. Campbell. Multiple Design Patterns for Voice over IP (VoIP) Security. Available at http://citeseer.ist.psu.edu/anwar06multiple.html [Last Accessed: 15 December 2006].

[5] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman. MIKEY: Multimedia Internet KEYing. Technical report, Ericsson Research, August 2004.

[6] J. Arkko, V. Torvinen, G. Camarillo, A. Niemi, and T. Haukka. RFC 3329 - Security Mechanism Agreement for the Session Initiation Protocol (SIP). Technical report, RTFM, Inc. Stanford University, January 2003.

[7] B. Buesker and K. Kovisto and B. Nottingham and C. Saout and R. Spenneberg. IPSec-Tools, 2006. Available at http://ipsec-tools.sourceforge.net/ [Last Accessed: 15 December 2006].

[8] J. G. Beerends, A. P. Hekstra, A. W. Rix, and M. P. Hollier. Perceptual Evaluation of Speech Quality (PESQ), the new ITU standard for end-to-end speech quality assessment. Part II Psychoacoustic model. Technical report, ITU-T, October 1998.

[9] J. Bilien. Key Agreement for Secure Voice over IP. Master's thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, December 2003.

[10] Y. Boger. Fine-tuning voice over packet services. Technical report, VP Business Development, RADCOM Ltd, June 1999.

[11] J. Bowls and S. Vance. Scripting with Perforce. Technical report, Swinburne University of Technology - Center for Advanced Architectures, April 2005.

[12] F. Cao and C. Jennings. Providing Response Identity and Authentication in IP Telephony. In *First International Conference on Availability, Reliability and Security (ARES 06)*. IEEE, 2006.

[13] J. Case, M. Fedor, M. Schoffstall, and J. Davin. RFC 1067 - A Simple Network Management Protocol (SNMP). Technical report, University of Tennessee at Knoxville, NYSERNet, Rensselaer Polytechnic Institute and Proteon Inc, August 1988.

[14] S. Casner and H. Schulzrinne. RFC 3551: RTP profile for Audio and Video Conferences with Minimal Control. Technical report, Columbia University, Packet Design, July 2003.

[15] F. Chatzipapadopoulos, G. De Zen, T. Magedanz, I.S. Venieris, and F. Zizza. Harmonised Internet and PSTN service provisioning. *Computer Communications*, 23:731–739, 2000.

[16] R. Chen, W. Lee, and J. Lin. Optimal Bandwidth Allocations for VoIP Latency Guarantees. *Information Science and Engineering*, 20:869–884, 2004.

[17] C. Chuah. Providing End-to-End QoS for IP-Based Latency-sensitive Applications. Technical report, University of California at Berkeley, October 2006.

[18] B. Clayton, B. Irwin, and A. Terzoli. Securing Real-time multimedia: A brief survey. In Jan Eloff, Hein Venter, Les Labuschagne, and Mariki Eloff, editors, *Proceedings of the ISSA 2005 from Insight to Foresight Conference (ISSA 2005)*, Sandton, South Africa, 2005.

[19] B. Clayton, B. Irwin, and A. Terzoli. Integrating Secure RTP into the Open Source VoIP PBX Asterisk. In Jan Eloff, Hein Venter, Les Labuschagne, and Mariki Eloff, editors, *Proceedings of the ISSA 2006 from Insight to Foresight Conference (ISSA 2006)*, Sandton, South Africa, 2006.

[20] B. Clayton, A. Terzoli, and B. Irwin. Performance Cost in Securing Confidentiality, Integrity and Authenticity of VoIP Communications. In *Proceedings of SATNAC 2005 - Convergence - Can technology Deliver*, September 2005.

[21] B. Clayton, A. Terzoli, and B. Irwin. DRAPA - a flexible framework for evaluating the quality of VoIP components. In *Proceedings of SATNAC 2006 - Convergence - The network @ work*, September 2006.

[22] A. E. Conway and Y. Zhu. A simulation-based methodology and tool for automating the modeling and analysis of voice-over-IP perceptual quality. *Performance Evaluation*, 54(2):129–147, 2003.

[23] Adrian E. Conway and Yali Zhu. Analyzing Voice-over-IP Subjective Quality as a Function of Network QoS: A Simulation-Based Methodology and Tool. In *TOOLS '02: Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, pages 289–308, London, UK, 2002. Springer-Verlag.

[24] Joan Daemen and Vincent Rijmen. *AES Proposal: Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[25] C. R. Davis. *IPSec. Securing VPNs*. Osborne/McGraw-Hill, Berkeley, California, 2001.

[26] S. Deering and R. Hinden. RFC 2460 - Internet Protocol, Version 6 (IPv6). Technical report, Cisco and Nokia, December 1998.

[27] M. del Rey. RFC 793 - Transmission Control Protocol (TCP). Technical report, Information Sciences Institute, University of Southern California, September 1981.

[28] M. W. Dixon and T. W. Koziniec. Using OPNET to Enhance Student Learning in a Data Communications Course. In *IS2002 Proceedings of the Informing Science and IT Education Conference*, pages 349–355. ACM Press/Addison-Wesley Publishing Co., June 2002.

[29] D. Ferrari. Design and application of a delay jitter control scheme for packet-switching internetworks. In *Proceedings of the second International Conference on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Germany, 1991.

[30] R. Fielding, UC Irvine, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. Technical report, UC Irvine, Compaq/W3C, W3C/MIT, Xerox, Microsoft, W3C/MIT, June 1999.

[31] J. Fischl and H. Tschofenig. Session Description Protocol (SDP) Indicators for Datagram Transport Layer Security (DTLS). Technical report, CounterPath Solutions, Inc. and Siemens, February 2006.

[32] Creighton T. R. Hager, Scott F. Midkiff, Jung-Min Park, and Thomas L. Martin. Performance and energy efficiency of block ciphers in personal digital assistants. In *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pages 127–136, Washington, DC, USA, 2005. IEEE Computer Society.

[33] T. A. Hall. Objective Speech Quality Measures for Internet Telephony. In *Proceedings of SPIE*, volume 4522, 2001.

[34] Jeong-Soo Han, Seong-Jin Ahn, and Jin-Wook Chung. Study of delay patterns of weighted voice traffic of end-to-end users on the voip network. *Int. J. Netw. Manag.*, 12(5):271–280, 2002.

[35] M. Handley and V. Jacobson. RFC 2327: SDP: Session Description Protocol. Technical report, ISI, LBNL, April 1998.

[36] J. Hitchcock. Decorating asterisk: Experiments in service creation for a multi-protocol telephony environment using open source tools. Master's thesis, Department of Computer Science (Rhodes University), Grahamstown, South Africa, March 2006.

[37] J. Hitchcock, J. Penton, and A. Terzoli. A multilanguage, open source approach to connecting user interfaces to a next-generation PBX. In *Proceedings of SATNAC 2004 - Next Generation Networks*, September 2004.

[38] ITU-T. Pulse code modulation (PCM) of voice frequencies. Technical report, International Telecommunication Union, 1988.

[39] W. Jiang. Detecting and measuring asymmetric links in an IP network. Technical report, Columbia University, New York, January 1999.

[40] D.R. Juhn, T.J. Walsh, and S. Fries. Security Considerations for Voice Over IP Systems. Technical report, US National Institute of Standards and Technology, January 2005.

[41] J. Klensin. RFC 2821 - Simple Mail Transfer Protocol (SMTP). Technical report, AT&T Laboratories, April 2001.

[42] H. Lipmaa. AES Candidates: A Survey of Implementations. Technical report, Kuberneetika AS, Tallinn, Estonia, February 1999.

[43] E. Mahfuz. Packet Loss Concealment for Voice Transmission over IP Networks. Master's thesis, Department of Electrical Engineering (McGill University), Montreal, Canada, September 2001.

[44] D. McGrew. libSRTP 1.4 Overview and Reference Manual. Technical report, Cisco Systems, Inc, 2001.

[45] D. McGrew, E. Carrara, M. Baugher, M. Naslund, and K. Norrman. RFC 3711: The Secure Real-time Transport Protocol (SRTP). Technical report, Cisco Systems, Inc and Ericsson Research, March 2004.

[46] E. Mier, D. Mier, and R. Tarpley. VoIP analysis tools: Picking up VoIP-specific tools for the network management workbench. Technical report, Network World, March 2003.

[47] Daniel Minoli and Emma Minoli. *Delivering Voice over IP Networks*. Wiley, New York, 1998.

[48] A. Muthukrishnan and D. Manjunath. On Incorporating Playout Adaptation and Loss Recovery in VoIP Applications. Technical report, Department of Electrical Engineering, Indian Institute of Technology, November 2006.

[49] MySQL. The world's most popular open source database, 2006. Available at http://www.mysql.com/ [Last Accessed: 15 December 2006].

[50] Newport Networks. White paper - voip bandwidth calculation. Technical report, Newport Networks, Castlegate Business Park - Monmouthshire - United Kingdom, 2005.

[51] G. J. Nutt. *Operating Systems: a modern perspective*. Addison Wesley Longman, Inc, Boulder, Colorado, 2001.

[52] J. Opie. Securing soft-switches from malicious attacks. Master's thesis, Department of Computer Science (Rhodes University), Grahamstown, South Africa, January 2007.

[53] J. Penton and A. Terzoli. Asterisk: A Converged TDM and Packet-based Communications System. In *Proceedings of SATNAC 2003 - Next Generation Networks*, September 2003.

[54] D. C. Plummer. RFC 826 - An Ethernet Address Resolution Protocol (ARP). Technical report, MIT-MC, November 1982.

[55] T. Porter, B. Baskin, L. Chaffin, M. Cross, J. Kanclirz jr, A. Rosela, C. Shim, and A. Zmolek. *Practical VoIP Security*. Syngress Publishing, Inc, Rockland, MA, 2006.

[56] J. Posegga and J. Seedorf. Voice Over IP: Unsafe at any Bandwidth? In *ZISC Information Security Colloquium*, 2005.

[57] J. Postel. RFC 792 - Internet Control Message Protocol. Technical report, ISI, September 1981.

[58] Collaborative Digitization Program. Digital Audio Best Practices. Technical report, Digital Audio Working Group, November 2005.

[59] M. K. Raganathan and L. Kilmartin. Performance analysis of secure session initiation protocol-based VoIP networks. *Computer Communications*, 26:552–565, 2003.

[60] E. Rescorla and N. Modadugu. RFC 4347 - DTLS: Datagram Transport Layer Security. Technical report, RTFM, Inc. Stanford University, April 2006.

[61] Luigi Rizzo. Dummynet: A simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, 1997.

[62] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC 3261: SIP Session Initiation Protocol. Technical report, Dynamicsoft, Columbia U., Ericsson, WorldCom, Neustar, ICIR, AT&T, June 2002.

[63] S. Chen and K. Nahrstedt. An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions. *IEEE Network*, 12(6):64–79, 1998.

[64] K. Salah and A. Alkhoraidly. An OPNET-based simulation approach for deploying VoIP. *Int. J. Netw. Manag.*, 16(3):159–183, 2006.

[65] Bruce Schneier. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop*, pages 191–204, London, UK, 1994. Springer-Verlag.

[66] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889 - RTP: A Transport Protocol for Real-Time Applications. Technical report, GMD Fokus, Precept Software Inc, Xerox Palo Alto Research Center, Lawrence Berkeley National Laboratory, January 1996.

[67] J. Scourias. Overview of the Global System for Mobile Communications. Technical report, University of Waterloo - Shoshin Research Group, October 1997.

[68] S. Sotillo. Zfone: A New Approach for Securing VoIP Communication. Technical report, East Carolina University, June 2006.

[69] M. Spencer, M. Allison, and C. Rhodes. The Asterisk Handbook. Technical report, Digium, Inc, March 2005.

[70] M. Spencer and F. Miller. Inter-Asterisk eXchange (IAX) Version 2. Technical report, Digium (Inc), Cornfed Systems (LLC), July 2005.

[71] Michael Steiner, Gene Tsudik, and Michael Waidner. Refinement and extension of encrypted key exchange. *SIGOPS Oper. Syst. Rev.*, 29(3):22–30, 1995.

[72] J. Turunen, P. Loula, and T. Lipping. Assessment of objective voice quality over best-effort networks. *Computer Communications*, 28:582–588, 2005.

[73] W. A. Vanhonacker. Evaluation of the FreeBSD Dummynet network performance simulation tool on a Pentium-4 based Ethernet Bridge. Technical report, Swinburne University of Technology - Center for Advanced Architectures, December 2003.

[74] voip-info.org. IAX encryption, 2006. Available at http://www.voip-info.org/wiki/view/IAX+encryption [Last Accessed: 15 December 2006].

[75] T. Ylonen and C. Lonvic. RFC 5251 - The Secure Shell (SSH) Protocol Architecture. Technical report, SSH Communications Security Corp, Cisco Systems Inc, August 2006.

[76] P. Zimmermann, A. Johnston, and J. Callas. ZRTP: Extensions to RTP for Diffie-Hellman Key Agreement for SRTP. Technical report, Zfone Project, Avaya, PGP Corporation, October 2006.

Appendix A

Glossary of terms

AES Advanced Encryption Standard. An iterated block cypher, sometimes called Rijndael.

AH Authentication Header. An encapsulation method employed by IPSec which provids authentication for IP packets.

API Application Programming Interface.

ARP Address Resolution Protocol.

ATM Asynchronous Transfer Mode. A high bandwidth, fixed-size packet-switching network.

Bing A tool used to compute point-to-point throughput using two sizes of ICMP ECHO_REQUEST packets to pairs of remote hosts.

CIA Confidentiality, Integrity and Avalibility are three areas which should be addressed when securing network data.

Circuit-switched A circuit-switched network is one which maintains a logical circuit for each session. This provides true QoS.

CPU Central Processing Unit.

DES The Data Encryption Standard is a cryptographic block cypher.

DMOS A Distortion Mean Opinion Score rates the amount of distortion during a telephone call.

DRAPA The Distributed Real-time Application Performance Analyser. A real-domain testbed which measure the performance of VoIP systems.

DRAPAC The DRAPA Controller manages the testbed by performing tasks such as device control and data collection.

DRAPAD The DRAPA Daemon is responsible for maintaining resources within a DRAPA testbed. For example, it ensures each node is running the latest DRAPA software.

DRAPAS The DRAPA Slaves manage the state of shared resources within the DRAPA testbed. A slave will inform the central controller when a resource is being used for an external function or if it is available for DRAPA to use.

DTLS Datagram Transport Layer Security. Currently a draft specification which describes a method of TLS for the UDP transport protocol.

ESP The Encapsulation Security payload is a method employed by IPSec provide confidentiality, authentication and integrity for IP packets.

FreeBSD Free-BSD is an advanced variant BSD UNIX operating system.

FTP The File Transfer Protocol is used to move files between a server and client.

GSM The Global System for Mobile communication is a digital cellular system.

GUI Graphical User Interface.

HMAC Hashed Message Authentication Codes are similar to hashing functions but they require a secret key in conjunction with the plain text. They are used to authenticate a communicated message.

Hop-by-hop A hop-by-hop method of security protects data while on the wire. The data is unprotected when lifted off the wire, and protected again when put back on the wire by each node until its destination is reached.

HTTP The Hyper-Text Transfer Protocol is typically used to publish web-pages.

IAX The Inter-Asterisk eXchange protocol used by Asterisk. Initially IAX was a trunking protocol specifically for Asterisk. However, it is available in VoIP clients too.

ICMP The Internet Control Message Protocol utilises error, control and information messages to manage IP connections.

IKE The Internet Key Exchange protocol provides a method of key exchange for security protocols.

IPFW The IP Firewall for BSD.

IP The Internet Protocol.

IPv4 The Internet Protocol, version 4.

IPv6 The Internet Protocol, version 6.

ISAKMP The Internet Security Association Key Management Protocol is responsible for maintaining security associations.

ITU International Telecommunications Union. An international standards body for telecommunications.

Jitter Varying latency in a real-time media stream.

LAN A Local Area Network is a computer network covering a local area, for example an office.

MAC Message Authentication Codes accompany a message. Their role is to prove the authenticity of the message.

MD5 Message Digest 5 is a hashing function that converts data into a fixed sized digest.

MGCP The Media Gateway Control Protocol is used to connect VoIP networks to legacy PSTN networks.

MIKey The Multimedia Internet Keying exchange protocol.

MIME Multipurpose Internet Mail Extensions is a standard for transporting files within mail and web protocols .

MOS A Mean Opinion Score is used to rate the quality of a telephone call.

NAT Network Address Translation on an internet gateway allows many devices on a local network to share a single internet-facing IP address.

Nonce value A random number added to a secret before the secret is encrypted or hashed. This protects against replay or statistical attacks.

OMOS An Objective Mean Opinion Score is produced by software, whereas a Subjective MOS is made by a person.

OPNET A popular network modeler and simulator.

OSI The Interconnection of Open Systems is a standard consisting of seven well-defined layers. This standard is used for inter-computer networking.

PABX Private Automatic Branch eXchange.

Packet-switched A packet switched network is one which divides data into packets. This typically provides a best-effort service.

PESQ Perceptual Evaluation of Speech Quality is the current ITU-T objective speech quality measurement method. PESQ is an objective method for producing a MOS score.

PGP Pretty Good Privacy.

PIT A programmable interval timer. A hardware device that produces an interrupt after a specified amount of time has elapsed. Such a device is used by the Linux scheduler and produces an interrupt at an interval of ten milliseconds.

PKI Public Key Infrastructure uses a public and private key pair to provide confidentiality and authentication.

PSTN Public Service Telephone Network. A circuit switched voice network which provices QoS .

RFC Request For Comment.

RSA A public key cryptographic algorithm named after its inventors: Rivest, Shamir and Adleman.

RTP The Real-time Transport Protocol is used to transport session media within a real-time system.

RTCP The Real-time Transport Control Protocol is responsible for the management of RTP streams.

SA A Security Association is established between two devices within the IPSec method of security.

SAS Short Authentication String. A string employed in Zfone's mechanism to ensure authentication.

SDP Service Description Protocol is used by end-points of a communication session to agree on attributes for the session, such as which audio codec to use.

SHA The Secure Hash Algorithms is a group of cryptographic hashing functions. For example, SHA-1.

SIAX2 The Secure Inter-Asterisk eXchange protocol provides confidentality to IAX2 payloads.

SIP The Session Initiation Protocol is responsible for creating sessions between end-points.

SIP Proxy A SIP entity which acts as a server and client. A SIP Proxy makes requests on behalf of a client, allowing SIP routing via a central point.

SMOS A Subjective MOS rating is produced by a person, whereas an objective MOS rating is produced by software.

SMTP The Simple Mail Transfer Protocol is used to transfer electronic mail between mail servers.

SNMP The Simple Network Management Protocol provides a means to monitor and control network devices.

SPD Security Policy Database. The database which holds IPSec policies .

SQL Structured Query Language. This acronym is pronounced "sequel" and is a language which controls relational database systems.

SRTP Secure RTP provides confidentiality, integrity and authentication for the real-time transport protocol.

SRTCP Secure RTCP provides confidentiality, integrity and authentication for the real-time transport control protocol.

SSH Secure Shell is a protocol used to gain secure terminal control on a remote computer.

SSRC The Synchronisation Source Identifier is found within the RTP protocol. It is used to identify an RTP stream.

STUN Simple Traversal of UDP over NATs is a protocol communicated between an application (which utilises UDP) and a NAT gateway. It allows the gateway to create NAT rules which allow UDP to be transported through the gateway.

Talker overlap When two parties interrupt one another due to large delay in their communication channel.

TCP Transmission Control Protocol. A reliable IP transport protocol.

TDM Time Division Multiplexing.

Tick A metric which, in this study, represents 10 milliseconds of accounted for CPU time.

TLS Transport Layer Security is the latest version of SSL (Secure Socket Layer) security for the TCP transport protocol.

TMS DRAPA Test Management Server.

TOS Type of Service. A flag in the IP header which can be used to label a packet as containing real-time data.

TSN A Traffic Shaping Node is utilised in a DRAPA testbed. It simulates network conditions and collects traffic flow data.

UDP User Datagram Protocol. A connectionless, unreliable IP transport protocol.

UNIX An operating system originally developed by AT&T Bell Labs. Many varients of UNIX exist today, such as Linux and BSD.

URI A Uniform Resource Identifier is a formatted string used to identify resources for Internet services. For example, an electronic mail address is a URI.

VLAN A Virtual LAN is a network which exists as one logical network when its physical infrastructure is in fact separated.

VoIP Voice over Internet Protocol is the transport of voice data over an IP-based packet network.

VPN A Virtual Private Network allows a user to make use of the Internet to gain access to resources within a protected network. A VPN is usually protected so that confidentiality, integrity and authentication is ensured .

WFQ The Weighted Fair Queueing algorithm represents a method of stream identification so that multiple streams traversing a router can be serviced equally. This is achieved by inspecting the IP packets traversing the router and grouping them into streams.

ZRTP The Zimmermin RTP protocol provides a key exchange method for real-time multimedia systems, using the Diffie-Hellman key exchange algorithm.

Appendix B

Example skeleton of a DRAPA module

```
#!/usr/bin/perl
use strict;
sub main::dbq() {
        my $queryin = shift;
        # This function performs a 'full database query' to the database.
        # This function should return a single value in $result.
        return $result;
}
sub main::dbqq() {
        my $queryin = shift;
        # This function performs a 'quick database query' to the database.
        # This function should return a single value in $result.
        return $result;
}
sub main::create_session() {
        my $caller;
        my $calee;
        my $i;
        ($caller, $calee, $i) = @_;
        print "\t",$caller," --> ",$calee," (",$i,")\n";
        # This function should create a session between $caller and $calee.
}
sub main::stop_sessions() {
        # This function should close down any active sessions.
}
sub main::test_sessions() {
        my $calls = `ssh root\@almira.ict.ru.ac.za asterisk -rx "show\\ channels" |
        $calls = (split(/ /,$calls))[0];
        return $calls;
}
sub main::init_counters() {
```

```
        my $type;
        my $pcallcount;
        ($type,$pcallcount) = @_;
        # This function is called before the sampling period begins to expire.
        # It should perform any database house-keeping.
        # $type will be set to the name of loaded module.
        # $pcallcount will set to the current number of concurrent calls.
}
sub main::final_counters() {
        my $type;
        my $pcallcount;
        ($type,$pcallcount) = @_;
        # This function is called after the sampling period has expired.
        # It should perform any database house-keeping, for example
        #       it should collect data from distributed agents and write
        #       the data to the database.
        # $type will be set to the name of loaded module.
        # $pcallcount will set to the current number of concurrent calls.
}
sub main::discard_test() {
        # This function should perform any hous-keeping needed
        # if the current test cycle is aborted.
}
sub main::do_math() {
        # This function is called after a test cycle is completed.
        # It should perform minor house-keeping on recently collected data.
}
```

Appendix C

Demonstrated command line usage of DRAPA

```
root@hobbes:/home/labs# ./drapa-mode1.pl 3-SRTP-Twoway.pm
Getting max calls over all test statistics .. 112
Do we have an unusually large number of slaves on line? .. No - 2
Are there call counts which have not been tested at all? .. No
Is there a stat that lacks testing ..  Yes - 2
About to perform a test of 2 calls for type 3-SRTP-Twoway.
Putting the following slaves to work:
        ug198.ict.ru.ac.za --> almira.ict.ru.ac.za (1)
        ug199.ict.ru.ac.za --> almira.ict.ru.ac.za (2)
```

Appendix D

SRTP policy for Asterisk implementation

```
/* Create stream policies */
srtp_policy_t policy;
octet_t key[30];
sec_serv_t sec_servs = sec_serv_conf_auth; /* configure:
                                              confidentiality and
                                              using AES_128_ICM and
                                              HMAC_SHA1  */
/* Set the pre shared key */
memcpy(key,"c1eec3717da76195bb878578790af71c4ee9f859e197a414a78
                                                    d5abc7451",64);

crypto_policy_set_rtp_default(&policy.rtp);
crypto_policy_set_rtcp_default(&policy.rtcp);
policy.ssrc.type  = ssrc_specific;
policy.ssrc.value = rtp->ssrc;
policy.key = key;
policy.next = NULL;
policy.rtp.sec_serv = sec_servs;
policy.rtcp.sec_serv = sec_serv_conf;

/* Add the stream policy to a session */
session = malloc (sizeof(session));
srtp_status = srtp_add_stream(session, &policy);
/ * Check that the sesson was created */
if (srtp_status) {
    ast_log(LOG_ERROR, "SRTP failed to add the session %d with
                          error code %d\n, rtp->ssrc, srtp_status);
} else {
    ast_log(LOG_NOTICE, "Session created.\n");
}
```

Appendix E

libSRTP protect() added to Asterisk

```
if (rtp->them.sin_port && rtp->them.sin_addr.s_addr) {
    /* Provide SRTP protection */
    packet_len = f->datalen + hdrlen;
    /* Create a buffer to perform SRTP protection. */
    memcpy(buffer, rtpheader, packet_len);
    /* Call libSRTP */
    srtp_status = srtp_protect(session, &buffer, &packet_len);
    if (srtp_status) { /* Report the error code if protect was not succsessfull */
        ast_log(LOG_ERROR, "SRTP protect ERROR %d
                    with error code %d\n", rtp->ssrc, srtp_status);
    }
    /* Modify the call to 'sendto()' */
    //res = sendto(rtp->s, (void *)rtpheader, f->datalen + hdrlen,
              0, (struct sockaddr*)&rtp->them, sizeof(rtp->them));
    res = sendto(rtp->s, (void *)buffer, packet_len,
             0, (struct sockaddr *)&rtp->them, sizeof(rtp->them));
    ...
}
```

Appendix F

libSRTP unprotect() added to Asterisk

```
/* Provide SRTP unprotection */
int packet_len = sizeof(rtp->rawdata);
char buffer [packet_len + SRTP_MAX_TRAILER_LEN];
/* Create a buffer to perform SRTP unprotection. */
memcpy(buffer, rtp->rawdata, packet_len);
srtp_status = srtp_unprotect(session, &buffer, &packet_len);
if (srtp_status) {
    ast_log(LOG_ERROR, "SRTP failed to unprotect steam %d with error code
                                                %d\n", rtp->ssrc, srtp_status);
}
/* Move the unprotected packet back to rtp->rawdata */
memcpy(rtp->rawdata, buffer, packet_len);
```

Appendix G

Use of the accompanied CD

On the CD you will find five folders:

- DRAPA Code: Contains the Perl source code for the DRAPA system along with a suitable MySQL database structure. A template of a pluggable module is also included.
- Graphs: Contains digital copies of the graphs generated by the DRAPA system for this project.
- Literature: Contains copies of the electronic literature cited in this project. Each file is named using the title of the paper.
- SRTP-Asterisk Code: Contains the two implementations of SRTP into Asterisk. Two sub-folders are found, one containing the version maintained by the Asterisk community. The other contains the version developed during this project.
- Thesis: Contains a digital copy of this document.