NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

OPNET PERFORMANCE SIMULATION OF NETWORK SECURITY SERVICES

by

Frederick R. Carlson

June 1999

Principal Advisor: John C. McEachen

Associate Advisor: Dan C. Boger

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE OPNET PERFORMANCE SIMULATION OF NETWORK SECURITY SERVICES	5. FUNDING NUMBERS

6. AUTHOR(S)
Carlson, Frederick R.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Information Warfare Activity 9800 Savage Road Fort Meade, Maryland 20755	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis conducts a performance simulation of Asynchronous Transfer Mode (ATM) and Kerberos security solutions. Specifically the study will build a working modeling framework of the Kerberos security service and the CellCase ATM encryption service. The model will be used to look at how these services will affect throughput by inserting waiting times in a series of queues in a small sized network.

The algorithms for calculating cryptographic delay are then inserted in an OPNET model and examined against a control model for validation. These models assume a linear relationship between the cryptographic service time and the throughput. Further relationships between service time and throughput are suggested for use in other security systems.

This thesis concludes that the modeling framework presented is viable for creating higher fidelity performance simulations of network security services. Further, this thesis suggests model validation directions for future research.

14. SUBJECT TERMS Joint Command, Control, Communications, Computers and Intelligence (C^4I), Optimized Network Evaluation Tool (OPNET), Kerberos, Asynchronous Transfer Mode (ATM)	15. NUMBER OF PAGES 99
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFI- CATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298 102

i

OPNET PERFORMANCE SIMULATION OF NETWORK SECURITY SERVICES

Frederick R. Carlson
Captain, United States Army
B.E.T., University of South Florida, 1989
B.A., University of South Florida, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY

from the

NAVAL POSTGRADUATE SCHOOL
June 1999

ABSTRACT

This thesis conducts a performance simulation of Asynchronous Transfer Mode (ATM) and Kerberos security solutions. Specifically this study builds a working modeling framework of the Kerberos security service and the CellCase ATM encryption service. The model is used to look at how these services affect throughput by inserting waiting times in a series of queues in a small-sized network.

The results of algorithms for calculating cryptographic delay are then inserted in an OPNET model and examined against a control model for validation. These models assume a linear relationship between the cryptographic service time and the throughput. Further relationships between service time and throughput are suggested for use in other security systems.

This thesis concludes that the modeling framework presented is viable for creating higher fidelity performance simulations of network security services. Further, this thesis suggests model validation directions for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The purpose of this thesis is to propose a simulation framework for encryption-enabled communications services using the OPNET™ Modeler simulation system with particular attention to the field of ATM encryption and PKI authentication. The field of modeling data communication networks is relatively new and the modeling and simulation community has not given substantial attention to modeling cryptographic performance effects. This must change. The proliferation of virtual private network encryptors, link encryption and certification services are beginning in the commercial world. These services can have tremendous impact on the performance characteristics of the network across the enterprise.

This thesis looks at two cryptographic systems and creates novel performance models for each. The first is Kerberos, which has been a third party authentication system for several years. The second model presented is the CellCase™ ATM encryption system. Both models are queuing models. This thesis has the premise that a cryptographic negotiation is nothing more than a series of queuing delays across the enterprise that require service times to compute and negotiate the appropriate cryptographic protocols.

In both models, there are two critical factors. The first are the service times that a particular encryption schemes needs across the negotiation process. The second is the number of authentications allowed per negotiation. The models in this thesis deal with

the first in the Kerberos model and CellCase model and assumes that an every negation requires an authorization.

This thesis provides a summary of current ATM security issues and standards and Public Key Infrastructure and the OPNET™ models for each. ATM is being fielded throughout the DoD and so is third party Public Key Infrastructures (PKI). This thesis is a good starting point to examine third party PKI issues with the Kerberos Model and to examine "key agile" call setup processes.

These models are in a very immature form but they are novel. There is a solid starting point to build higher fidelity cryptographic performance models from the lessons learned from this research. The performance of future DoD networking efforts will very probably need to factor in delay and throughput performance factors in encrypted data networks, particularly as network managers scale PKI resources to larger and larger levels.

The next phase of this research would be twofold. First, we need to validate the Kerberos and CellCase™ OPNET™ models presented in this thesis. The Kerberos model can be adapted to fit DISA's ongoing PKI initiative. Taking these models and adapting them to a small PKI network like Defense Travel Office PKI testbed would allow real world validation of the algorithms in this thesis and the OPNET™ model. Second, the models here are shown as very simple point-to-point networks. Research into adapting this model to a multiple server/client/authenticator network in the Kerberos

OPNET™ model would yield valuable insights into the performance problems that a viable PKI authentication service, such as the one DISA is now planning, would have.

I. INTRODUCTION

A. PURPOSE

The wide area telecommunications infrastructure is changing rapidly from an unencrypted one to an encrypted world. Secure telecommunications were, until very recently, only reserved for the military and the largest of corporations.

Concurrent with this change, we are seeing a complete overhaul of the public switched telephone network. The overwhelming majority of the current telecommunications system is based on circuit switching, time division multiplexing, copper cabling and static routing. We are shifting to a network based on more adaptive routing, fiber optic cabling and statistical multiplexing. This shift in media, routing, and transmission has brought forth new standards such as the Synchronous Optical NETwork (SONET) for transmission services and Wideband Division Multiplexing (WDM) for multiplexing services. It is far from certain, but it appears that the replacement for the switching component of the PSTN will be based on a statistical multiplexing scheme based on Asynchronous Transfer Mode (ATM) standards. The following quote is from the DoD Contingency C4ISR Handbook: "The far term strategy for DISN support is intended to fully populate the entire DISN, both fixed and

1

deployed, with identical technical architectures. This will eliminate any discernable interface or transition points. It should be noted that, while the current deployed segment vision is based on ATM technology, it is not an irrevocable decision." [CONT 98]

All of these standards are in a state of flux. The security services for this emerging network are in a state of infancy and the modeling of these services is also very sparse. However, one trend looks relevant. The only true model of a wide area secure system available to industry and academia is the one provided by the military, which has been securing networks for years using encryption at the trunk, group and loop level.

The purpose of this thesis is to discuss and analyze the performance of a certificate-based security framework. This will be done by taking a known queuing analysis of Kerberos, converting it to an OPNET model, and then repeating the process with the CellCase ATM encryption service.

B.   MOTIVATION

Significant challenges exist in securing a global wide-area network.   The core of this challenge is to organize cryptologic systems at multiple levels in the protocol stack without causing a significant performance problem throughout the network.   The engine that has made secure communications

happen within the military has been an array of symmetric key devices that are placed at various locations within the network. These inline network encryptors (INE) are used to link encrypt both the Internet Protocol (IP) and voice networks in the Department of Defense (DoD).

DoD networks currently operate almost exclusively using IP for data transfer across the various interconnected networks. However, early use of ATM networks has been deemed very successful. The generally accepted time frame for this integration of an ATM switching architecture is 3-7 years with significant deployments occurring right now. [COHE 96]

The upcoming ascendancy of ATM networking requires an analysis of the performance impact of any and all security services unique to the ATM networking environment. DoD needs to understand how much security will cost in terms of traditional performance parameters such as call setup time compared with encrypted call setup time. It is our hope that the novel simulation presented in this thesis will help create a framework for future modeling of the services.

C. ORGANIZATION

The remaining chapters in this thesis will be presented as follows:

1. Asynchronous Transfer Mode

Chapter II will discuss the benefits of ATM and describe how ATM will become the backbone of military command and control systems.

2. Public Key Infrastructure

Chapter III will describe the public key infrastructure (PKI) and the performance considerations involved in deploying a public-key-based security system.

3. Basic Description of Secure ATM Channels and the Kerberos Protocol

Chapter IV will describe several schemes for securing ATM channels. The focus of this chapter is the description of algorithms such as key-agile and cell-based encryption methods and the current technology developed to institute those methods. This chapter will also describe the Kerberos protocol, which is the prototype for the analysis of ATM channels.

4. OPNET Simulation of the Kerberos Authentication Service and Secure ATM Channels

Chapter V will present a node level analysis of the problem at hand. The chapter will use an OPNET node level analysis to model the encryption devices and the certification servers. This chapter also discusses the metrics that we will be using to do a meaningful analysis of secure ATM traffic and the Kerberos authentication system.

5.   Conclusions

Chapter VI recommends areas of additional research and concludes with a summary analysis of the ATM security scheme and the simulation of secure ATM channels. This chapter will also describe the results of the simulation and concludes with a summary analysis of the validity concerns for the models developed.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ASYNCHRONOUS TRANSFER MODE

A. ATM OVERVIEW

Asynchronous Transfer Mode is a telecommunications standard where information packets are transferred asynchronously. The first research on ATM and its related technologies was published in 1983 by two research centers, CNET and AT&T Bell Labs. In 1984, the research center of Alcatel Bell in Antwerp started to develop the ATM concept.

ATM has the same basic characteristics of packet switching, but also the delay characteristics of circuit switching technology. ATM is widely viewed as the communications technology of the 21st century, and its impact is expected to be much like the ubiquitous PCM (pulse code modulation), which is used widely throughout current telecommunications networks[KUMA 95].

There are four advantages that ATM brings compared to standard packet switching or pure circuit switching. The first advantage is that ATM, because of its channeled nature enables network planners to avoid fractal, self-similar traffic patterns. Self-similar traffic patterns, such as Ethernet and standard packet switched networks, are exceptionally difficult on which to measure quality of service (QoS) parameters. The channelization of ATM traffic allows telecommunications managers to use statistical

distributions and techniques to do the same type of traffic analysis that existed in time division multiplexed (TDM) networks.

The second advantage of ATM is that you can seamlessly multiplex data streams with differing delay characteristics to a single point. Voice, video and data traffic on a network channel have very different transmission needs. The transmission needs of these services are so different that you really can't have uncontrolled access to the channel and have decent QoS. An email with a fifty slide Power Point presentation could potentially block out a voice call if that were the case.

The third advantage of ATM is the reduction of transport costs. The statistical multiplexing technique that ATM uses enables the available bandwidth to be used much more efficiently than traditional TDM schemes. This mechanism will be described in the next section.

The last advantage of ATM is that it provides for dynamic bandwidth allocation. This advantage goes hand in hand with the reduction of transport costs. The flexible bandwidth allocation that ATM provides allows the network to give bandwidth when the user application requires it. This is very different than time division multiplexing, which gives bandwidth even if it is not needed by the application.

B. ATM FRAMEWORK

This section addresses the basic principle behind ATM: divide and conquer. [KUMA 95] At the most basic level, ATM divides all data into 48 byte cells with a 5-byte header. The process of dividing the data stream into cells is called segmentation. The process of reassembling the cells into coherent traffic is called reassembly. The multiplexing scheme that ATM is replacing for voice traffic is TDM. A notional example of TDM is shown below in Figure 1.
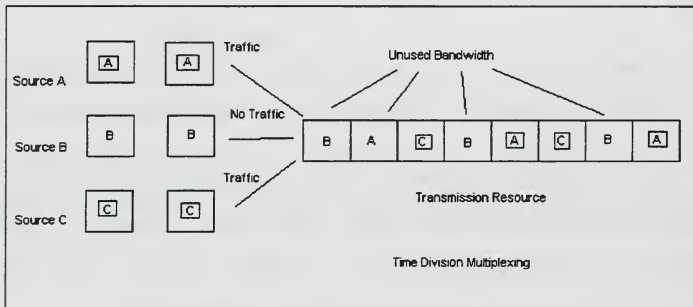
Figure 1 - TDM

In both networks, several data sources are multiplexed on a single link. In TDM networks the effective bandwidth is simply the sum of individual sources' bandwidths. If traffic source A has x bits per second (bps) and traffic source B has y bps then a TDM network would use x+y bps of bandwidth. ATM, using statistical multiplexing, has an

9

effective bandwidth of z bps, where z < (x+y) because the
multiplexer takes advantage of the fact that all individual
sources are not continuously transmitting. Figure 2 shows an
idealized example of statistical multiplexing.
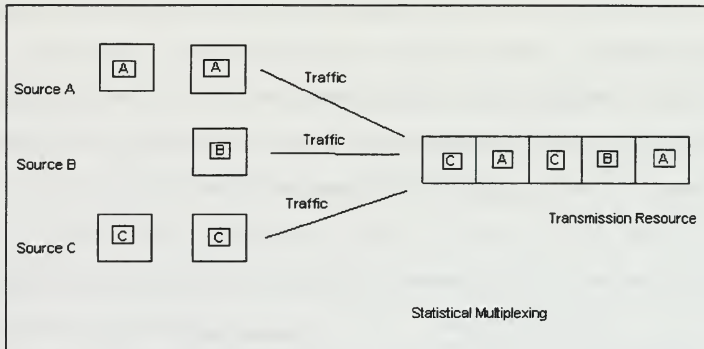
Figure 2 - Statistical Multiplexing

The 5-byte headers in ATM have little functionality so
the network can process them very fast. Figure 3 shows how
traffic of different speeds, 64 KBPS, 2Mbps, and 34Mbps are
divided into equal 48 byte packets. [KUMA 95]

C.   **ATM PROTOCOL**

The fixed 53-byte size of ATM cells along with the very
austere error correction and retransmission characteristics
of the ATM protocol give ATM a tremendous advantage for
applications   such   as   high-speed   networks.   Other
technologies such as X.25 and Frame Relay perform frame

10

delimitation and error checking. X.25 has a packet
retransmission function in addition to the features above.
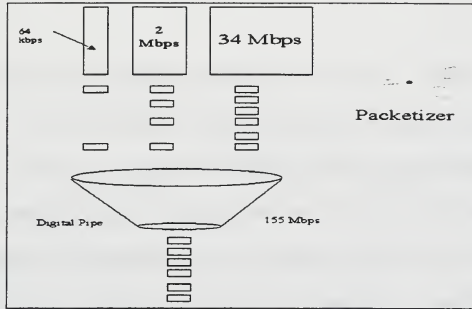
Figure 3 - ATM

The elimination of these functions is an advantage of
the ATM network because it simplifies the switching and
processing functionality of the network, leading to enhanced
performance.[KUMA 95] Figure 4 shows the protocol stack for
ATM.
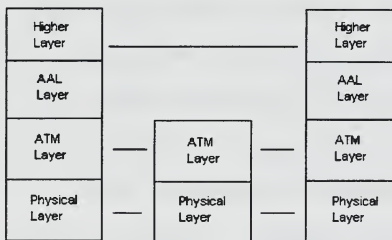
Figure 4 - ATM Protocol Stack

Higher layers of the protocol take care of the functions that are not performed at the ATM layer. The ATM protocol stack is roughly analogous to the seven layer OSI reference model.

The higher layers of the reference model convert the user information into standard ATM cells. The ATM layer adds 5 bytes of header information. This header carries information to route the cells in the ATM network. The ATM cell header consists of six different fields of varying sizes: Generic Flow Control (GFC), Virtual Channel Identifier (VCI), Virtual Path Identifier (VPI), Payload Type (PT), Cell Loss Priority (CLP), and Header Error Control (HEC).

1. Generic flow control (GFC)

The GFC provides contention resolution and simple flow control for shared medium-access arrangements at the customer premises equipment (CPE).

2. Virtual channel identifier (VCI)

The VCI is used to establish connections using translation tables at switching nodes that map incoming and outgoing VCIs.

3. Virtual path identifier (VPI)

The VPI maps groups of VCIs. The VPI is used in setting up the end-to-end virtual path connections of multiple virtual path segments.

4.   Payload type (PT)

The PT is a three-bit field that is used to differentiate management and data cells traversing the same virtual circuit.

5.   Cell loss priority (CLP)

The CLP is a one-bit flag used to determine cells of lower priority. Depending on network congestion, the cells with a set CLP may be discarded.

6.   Header error control (HEC)

Header error control (HEC) performs a CRC calculation on the first 4 bytes of the header field for error detection and correction. [KUMA 95]

Within the ATM cell header there are two formats: the user to network interface (UNI format), which is the header format for the cells between the user and the network. There is also the network to network interface (NNI format), which are between ATM backbone links. The UNI and NNI cell formats are shown in Figure 5 below.
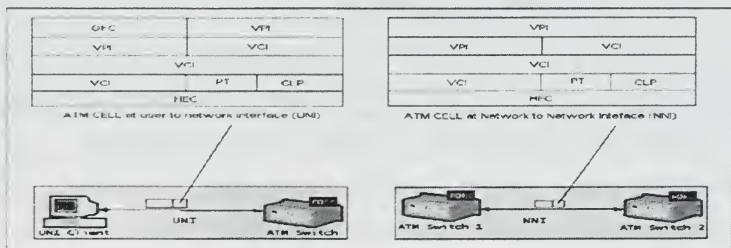
Figure 5 - Cell Formats

13

D.   **ATM SERVICE CATEGORIES**

   1.   **Switched Virtual Circuits**

   A user can establish ATM connections in one of two ways. The user can set up a switched virtual circuit (SVC) or a permanent virtual circuit (PVC). A switched virtual circuit is analogous to the voice telephone network where you do not have a pre-determined path set up for you. A switched virtual circuit is set up in a manner similar to a direct-dialed telephone call.

   2.   **Permanent Virtual Circuits**

   PVC service is similar to dedicated lines. The installation of PVC provides several advantages. First, the provisioning time is very short, with close to real-time provisioning of the circuit. Second, there are no call establishment procedures to go through. Third, PVCs provide excellent flexibility for extensions. The downside of PVC service is the cost and time required installing the service.

   3.   **Call Setup Procedure**

   This will be discussed in some detail, as it is a very important metric in the performance of secure ATM channels. A normal call exchange is made of six messages. The six messages are: setup, call proceeding, connect, connect acknowledge, release, release complete. There are two other messages that are common to ATM signaling: status inquiry and status. Figure 6 shows a typical ATM call exchange.
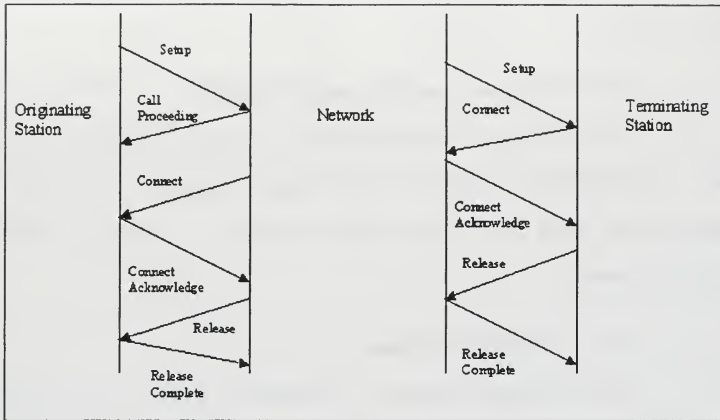
Figure 6 - ATM Call Exchange

a)   *Setup Signal.*

The setup phase is sent by the source to the ATM switch to initiate a connection.

b)   *Call Proceeding Signal*

The call proceeding signal is sent from the originating switch to the source acknowledging the start of the process to establish a call. The call proceeding signal may also be sent from the destination to the terminating switch.

c)   *Connect Signal*

The connect signal is sent from the destination to the terminating switch, passed through the network, and then

15

from the originating switch to the source for call acceptance.

d) Connect Acknowledge Signal

The connect acknowledge is sent by the source to the originating switch and then passed on from the terminating switch to the destination to complete the call setup process.

e) Release Signal

The release signal is sent by either the source or destination to terminate the call. The destination can also use the release signal with error codes to reject call setup.

f) Release complete signal

This signal is sent by the source, destination or other switches to complete the call teardown process and acknowledge release of network resources.

g) Other Signals

There are two other common messages: the status inquiry signal and the status signal. The status inquiry is sent by the source, destination, or network switches to request status messages. The status signal is a response to an inquiry.

E. SUMMARY

This chapter has given a very abridged overview of what ATM is and, in particular, what is involved in the call setup procedure. This information will link to the description of the CellCase encryption system in future chapters.

THIS PAGE INTENTIONALLY LEFT BLANK

III. PUBLIC KEY INFRASTRUCTURE

A. CRYPTOGRAPHY

1. Cryptographic Engines

Cryptography is the science of coding data so that the cost of improperly acquiring the data is greater than any benefit gained by having the data in the first place. A cryptographic engine is a mathematical technique to code this data. A cryptographic engine requires three elements: plaintext data, ciphertext data and a cryptographic algorithm.

There are three types of cryptographic algorithms. The first type is called a message digest or hashing algorithm. The second type is called a secret key algorithm. The last, and most recent, algorithm family is called public key cryptography.

Each of these algorithms requires a mathematical value called a key. A key has two values associated with it. The first, the key length, is the length in bits of the key. The second, the key space, is a collection of all mathematical values that have the same length of the key. A key of length n generates a key space of 2^n for a binary cryptographic engine. [FEGH 98]

2. Cryptographic Attacks

There are three major forms of cryptographic attacks that are relevant to the scope of this thesis. The first is called an ciphertext-only attack where a cryptanalyst intercepts some ciphertext and wishes to obtain the plaintext or recover the key. This can be done in several ways. The first method is called the brute-force attack where the analyst does an exhaustive search of the entire key space to find a matching key. The second brute-force method is called a dictionary attack where you try to match the encrypted text with common known text encrypted by the same cryptoengine.

The second form of attack is the known-plaintext attack. This is where the cryptanalyst has access to a ciphertext message and a corresponding plaintext message. It is extremely easy for the analyst to acquire both of these messages. E-mail applications, for example, place standard headers at the beginning of each message, which provides the analyst with a known plaintext.

The third form of attack is the chosen-plaintext attack. This is a subtle variant of the known-plaintext attack in which the cryptanalyst can select the plaintext that the cryptosystem sends. The classic example of this attack was the breaking of the Japanese cryptosystem before the Battle of Midway. The U.S. Navy sent a bogus message that Midway Island's water condenser was down. This

effectively inserted these words in the Japanese secure communications system and gave the U.S. Navy words that they could key on and use to break the code.

B.   SECRET KEY CRYPTOGRAPHY

    1.   Secret Key Algorithms

    Secret key cryptography uses a secret key to encrypt a message into ciphertext.   Secret key cryptography is also known as symmetric cryptography.   Figure 7 shows two parties, Alice and Bob, who use secret key cryptography to protect their message from Eve, who wishes to intercept it. Alice and Bob must agree on a shared secret key before they can share traffic.   Alice can run an advertisement in The New York Times to inform Bob of the key.   She can tell him the key over the telephone or slip the key in his mailbox.
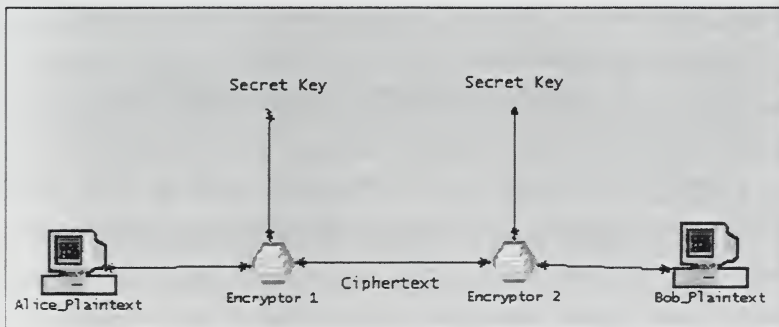
Figure 7 - Secret Key Encryption

The problem with this is that Eve could intercept the key if Alice used these methods. Key distribution centers have been proposed and used to solve the secret key exchange and have been used in the military for decades. These centers suffer from scalability and administration problems that make them difficult to implement.

There are a variety of secret key algorithms. The mathematics of each varies, as does the security. Table 1 shows current secret-key algorithms.

Algorithm Name	Mode	Key Length (bits)
Blowfish	Block cipher	Variable up to 448
DES	Block cipher	56
IDEA	Block cipher	128
RC2	Block cipher	Variable(1 to 2048)
RC4	Stream cipher	Variable(1 to 2048)
RC5	Block cipher	Variable(1 to 2048)
Triple DES	Block cipher	56 (112 effective)

Table 1 - Secret Key Algorithms

Secret key ciphers are generally used as bulk traffic encryption ciphers. The secret key algorithm performance is much faster then other forms of encryption such as public key. This speed advantage allows secret key ciphers to be very useful in encrypting at the group or supergroup level.

2. Block Ciphers

Block ciphers take a fixed-size block of plaintext, usually 64 bits, and produce a fixed-size block of ciphertext, usually at the same size as the input block. The length of the key may be 56 such as with DES or 128 for the IDEA algorithm.

The mapping from an input block to an output block must be one-to-one to make the algorithm reversible. If more than one input block is mapped to one output block, the algorithm can encrypt, but it cannot decrypt. The block size is a critical issue in the field of ATM security because the ATM cell format does not conform to any conventional blocksize used by any current or proposed secret-key algorithms.[FEGH 98]

3. Stream Ciphers

Stream ciphers operate over one bit (sometimes one 8-bit byte or one 32-bit word) of input data at a time. A specific and widely used stream cipher is the one-time-pad. The one-time-pads are stream ciphers that perform an XOR of one bit of an input stream with one randomly generated bit to get one bit out of the output stream. The sequence of random bits constitutes the session key, or the pad, which is the same size as the input and output streams.

C. PUBLIC KEY CRYPTOGRAPHY

1. Public Key Algorithms

Public key cryptography was officially invented by Whitfield Diffie and Martin Hellman in 1975. Recently, there has been widespread speculation that the concept was invented by the British Intelligence Agency GCHQ many years earlier. The scheme involves the use of two distinct keys, a public key and a private key. The public key is freely distributed to all entities that you want to communicate to while the private key is kept under very close hold. Figure 8 shows how our couple, Alice and Bob, can communicate using a public key cryptosystem.
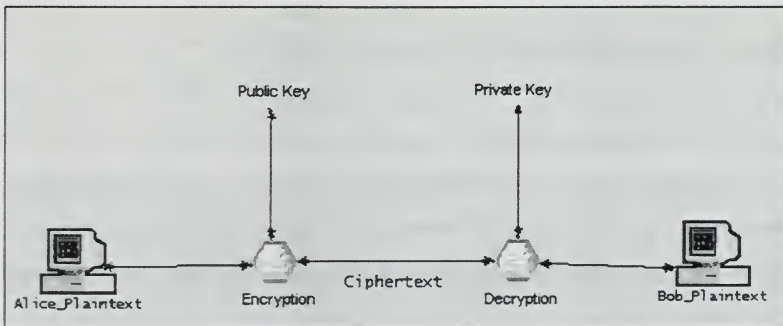
Figure 8 - A public key cryptosystem

2. Mathematical Foundations of Public Key Cryptography

Public key cryptography is based on the idea that you have an easy computation and a hard computation. Most public key encryption is based either on integer

24

factorization or discrete logarithms that provide both easy and hard computations. Public key systems based on integer factorization rely on the fact that multiplying large prime numbers is easy while factoring large numbers into primes is hard. The problem of finding a simple way to factor large numbers into primes has been looked at by some of the greatest minds in human history and has yet to be solved. A system based on integer factorization, such as RSA, will not allow a deriving of the public key from the private key within a computationally feasible timespan.

The second mathematical technique is the Discrete Logarithm. In modulo n (n>0), two integers are equivalent if they have the same remainder when divided by n. The remainder of an integer m divided by n is the smallest nonnegative integer that differs from m by a multiple of n. The numbers 6, -4, and 16 all have the same remainder 6, when divided by 10, and therefore, are equivalent in the arithmetic modulo 10. The easy computation (analogous to multiplication in integer factorization) is called modular exponentiation. Modular exponentiation a^x modulo n is the exponentiation in arithmetic modulo n. For example, 3^4 modulo 10 is 81 modulo 10 which is the same as 1 modulo 10. The hard computation is finding the discrete logarithm or the inverse of modular exponentiation. This is the problem of finding the x where $a^x = b$ modulo n. If $11^x = 1$ modulo 10, then x equals 2. [STIN 97]

25

The Diffie-Hellman key exchange algorithm uses the technique of discrete logarithms to exchange keys. This is of particular interest because the key exchange mechanism used by the CellCase ATM encryption system uses Diffie-Hellman. The Diffie-Hellman Algorithm uses the exponent d to hide a key and sends $e = g^d$ modulo p along an unsecured channel to another party. Eve can know e, g, and p but cannot solve for d [STIN 97].

3. Public Key Cryptography Standards

Public Key Cryptography Standards (PKCS) are a set of standards that the RSA Laboratories developed in collaboration with Apple, Digital, Lotus, Microsoft, MIT, Northern Telecom, and Sun Microsystems. The PKCS is analogous to the Request For Comment (RFC) framework that is responsible for standardizing the Internet. There are currently 12 papers that make up the PKCS, labeled PKCS#1 to PKCS#12.

4. Public key versus Secret Key Length

The relative strengths of public key and secret key cryptography are not one to one. For example to obtain the security of a 40 bit secret key cryptosystem you would have to have a 274 bit key length for a public key cryptosystem. Table 2 shows the relative strength of secret keys versus public keys [FEGH 98].

Secret Key (bits)	Standard Public Key (bits)
40	274
56	384
64	512
80	768
96	1024
112	1792
120	2048
128	2304

Table 2 - Strength of Public and Secret Keys

D.   **HASH ALGORITHMS**

Hash algorithms or message digest algorithms take a variable sized message and return a fixed length message as output. Hashes are critical in the generation of digital signatures. A hash must have the following properties to be secure. First, the hash must be able to work on messages of any size. Second, it must produce a hash of a fixed size. Third, it must be relatively easy to compute, so that it is a practical means of authentication. Fourth, the hash must depend on the whole message. Last, it must be prohibitively expensive to invert the hashing process.

Two hashing structures are relevant today, Message Digest 5 (MD5) and the Secure Hashing Algorithm (SHA). The SHA was developed by the National Institute of Standards and Technology (NIST) and was deemed the standard for the Federal Government in 1993. Table 3 provides a comparison between these two algorithms. [FEGH 98]

Algorithm	MD5	SHA
Digest Length	128 bits	160 bits
Basic Processing Unit	512 bits	512 bits
Number of Steps	64(four rounds of 16)	80
Maximum Message Size	Infinite	2^{64} Bits
Primitive Logical Functions	4	3
Additive constants used	64	4

Table 3 - A comparison of MD5 and SHA

E. DIGITAL SIGNATURES

1. Introduction

A digital signature is a data item that confirms the origin and integrity of a message or entity. This is very important even if the message is encrypted because it is an extra check on the entity's identity. There are two major Digital Signature Standards: RSA and the Digital Signature Standard (DSS). Figure 9 shows how digital signatures are generated and verified.

Alice first computes the digest of her message and uses her private key to sign the digest. Alice then transmits the message and the signed digest to Bob. Bob then decrypts the message and the hash and compares the two. If they are equal then the message by Alice is authentic. If they are not equal then the message by Alice is not authentic.
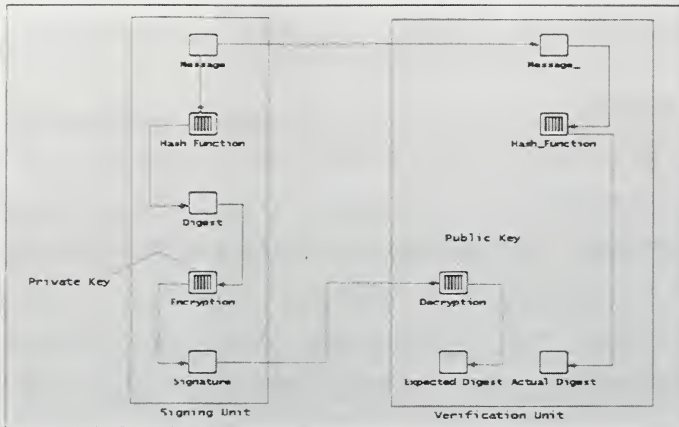
Figure 9 - Digital Signature Scheme

2. RSA Digital Signatures

The RSA digital signature was the defacto standard until the Federal Government proposed the DSS in 1991. The RSA signature scheme uses MD5 for the hashing algorithm and the RSA PK algorithm for encryption.

3. DSS Digital Signatures

The DSS digital signature became the federal standard for signatures in 1991. The DSS standard uses the El Gamal PK algorithm for encryption and the SHA-1 standard for

hashing. El Gamal is a PK algorithm that uses discrete logs as its mathematical engine. RSA uses integer factorization.

F.   SUMMARY

The purpose of this chapter was to introduce the reader to the components of public key cryptography and secret key cryptography. The simulation of the ATM Security solutions now proposed involves secret key cryptography for encryption and public key cryptography for key exchange and authentication. The concepts presented in this chapter and Chapter II will be tied together in the next chapter.

IV. DESCRIPTION OF SECURE ATM CHANNELS AND THE KERBEROS AUTHENTICATION PROTOCOL

A. ATM SECURITY FRAMEWORK

The ATM Forum Security Working Group proposes a Security Framework that addresses the basic requirements for ATM security. This framework identifies the main security objectives for ATM security: confidentiality, data integrity, accountability, and availability. The principal functions, which the ATM security system should provide under the ATM Forum framework, are listed in Table 4.

The ATM Forum Security Specification includes a very general interpretation of these functional areas.[SPEC97] The framework is abstract enough to provide a guideline to different ATM instances. It is interesting to note that only the first eight out of ten requirements provide security services. The last two are required to support the maintenance of security services.

1. ATM Security Scope

There are three planes relevant to ATM security. The three planes are the user plane, the control plane and the management plane. Each plane includes entities as shown in Figure 10.

31

Function	Description
Verification of Identities	System must be able to verify any claimed identity
Controlled Authorization and Access	Entities on network must not be able to gain access to information or resources unless authorized.
Protection of Confidentiality	All stored and communicated data must be confidential
Protection of Data Integrity	System must guarantee the integrity of the stored and communicated data
Strong Accountability	Entities can not repudiate actions or effects that they cause on the network
Activities Logging	The security system must support the capability to retrieve information
Alarm Reporting	The security system must be able to generate alarm information
Audit	System must keep detailed logs
Security Recovery	System must be able to recover from breaches of security
Security Management	The security system must be able to manage the security services derived from the above requirements

Table 4 - ATM Forum Security Framework Functions
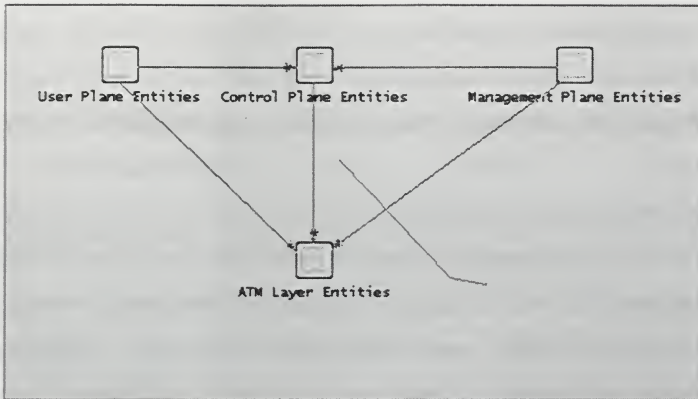
Figure 10 - ATM Entities Interaction

The entities in the user plane are used to transfer user data. The entities in the control plane deal with connection establishment, release and other connection functions. The management plane entities perform management and coordination functions that relate to both the user plane and the control plane. All three planes and the ATM layer must be included in the ATM security scope.[SPEC97]

2. Placement of ATM Security Services

The user plane is responsible for all interactions between the user and the network. The user plane provides services such as access control, authentication, data confidentiality, key exchange, certification infrastructure and integrity in an ideal ATM security framework.[SPEC97]

The control plane will configure the network to provide a communication channel for a user. Figure 10 shows that the control plane interacts with all other entities in the ATM entity model. The critical services to secure the control plane are authentication and confidentiality.[SPEC97]

The management plane security is also important. Management plane security considerations should consider the following items: bootstrapping security, authenticated neighbor discovery, the Interim Local Management Interface security and PVC security.[CHUA 96]

The last, and most important, layer of concern is the ATM layer. This is the layer where most ATM security solutions that have been proposed or under development reside. ATM security at the ATM layer must be implemented at the end-to-end, edge-to-edge, and ATM endpoint switches. Security at the ATM layer is the focus of the Secant/CellCase product line that evolved out of the MCNC effort [MCNC 99], the Cylink ATM encryptor, and the GTE TACLANE and FASTLANE encryptor [COHE 96].

3. ATM Security Challenges

The first challenge in ATM network security is to design a cryptographic mechanism that will match the high communication speed of the switch. Most traditional cryptographic mechanisms operate within 10 Mbps when

implemented in software and 100 Mbps when implemented in hardware. This creates a significant speed mismatch between the speed of the switch which will operate at hundreds of Mbps up to Gbps and the cryptographic engine that operates significantly slower.

The second challenge in ATM network security goes hand in hand with the first. The ATM cell payload is 48 bytes, which means that any block-ciphering scheme with a block size greater than 384 bits cannot be used without block chaining. If the block size is smaller than 384 bits the cryptographic algorithm needs to be block chained, which significantly increases the time required to encrypt the cell.[LIAN 96]

B. ENIGMA2 PROJECT

1. Project Contributors

The Enigma2 project is a collaboration between the Microcomputing Network Center (MCNC), Portland State University and the Mayo Clinic to provide a secure key-agile ATM security scheme under a DARPA contract. The Enigma2 project has spunoff a commercial effort by the Celotek Corporation (formerly Secant Corporation) with the development of CellCase ATM cryptographic systems products.

2. Project Overview

The Enigma2 project mission was to investigate issues associated with secure communications over public ATM networks. Enigma2 involves the design, fabrication and evaluation of a proof of concept key-agile ATM cryptographic system using the DES and RSA algorithms. Key agility refers to the ability of a cryptographic unit to dynamically change keys.[SHB95] The design goal was for the system to support full duplex secure communications at 622 Mbps. Up to 65,000 simultaneous connections can be active in the system with a unique cryptographic key for each connection. [MCNC99]

Enigma2 includes efforts in architecture development, software and hardware design, integration and analysis. The result was three Beta CellCase units that supported DES encryption and decryption of ATM cell payloads at the OC-12c rate. There were many other tasks involved in this effort, ranging from covert channel analysis to key management software.

The Enigma2 project claimed some significant accomplishments that are summarized as follows: creation of cryptographic systems technologies, transfer of these systems to industry, and understanding of ATM security issues. The Enigma2 project: (1) designed the first ATM cryptographic system that offers DES encryption at the OC-12 rate, (2) supports key agility by encrypting each VC with a unique key, (3) supports transparent operation of

36

cryptographic systems with existing ATM systems and with ATM Forum standard signaling, and (4) supports dynamic key distribution using a public key protocol.[MCNC98]

C. ENCRYPTION SYSTEMS

1. Link Encryption System

A link encryption system provides protection at the physical protocol layer. Figure 11 provides an example of this type of mechanism. This type of system can be compatible with and transparent to larger low layer protocols such as SONET, but it encrypts all data.[SHB95] This approach has the benefit of preventing traffic analysis by an eavesdropper. This mechanism is the overwhelming type of cryptosystem used in current military telecommunications networks.
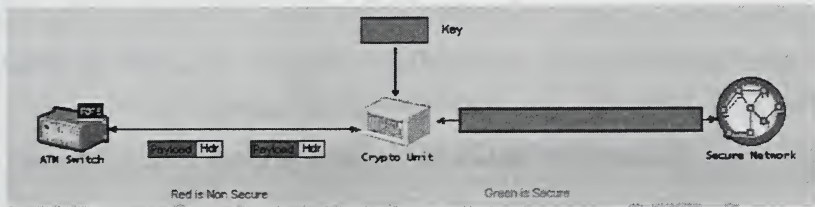
Figure 11 - Link Encryption System

2. ATM cell encryption system

ATM cell encryption codes the user data while leaving the cell headers in the clear. This approach permits privacy of communications with the full flexibility of ATM

networking while securing the data traffic at the switch nodes and the interswitch links. This is a much less secure method then link encryption because of the ability of an eavesdropper to gather network control information from the unsecured cell header. Figure 12 shows a notional cell encryption scheme.
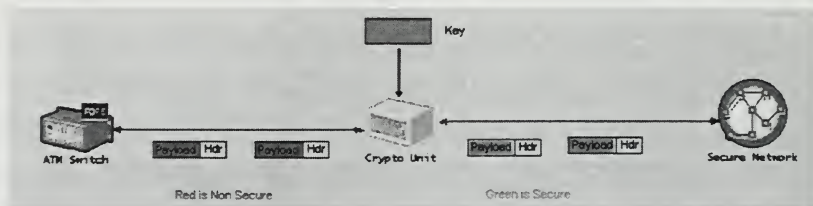
Figure 12 - Cell Encryption

3.   Key-agile cell encryption system

A third scheme for securing ATM networks is to use key agility. ATM uses key agility by having a unique key for each active VC in the system. Figure 13 shows a simple example of a key-agile ATM encryption system.

4.   Algorithm-Agile encryption system

A novel proposal by researchers at Sandia National Laboratory calls for using different cryptographic algorithms for each VC. [THBW98] The QoS ramifications of this cryptosystem, particularly with heterogeneous traffic, are very significant and are beyond the scope of this thesis.
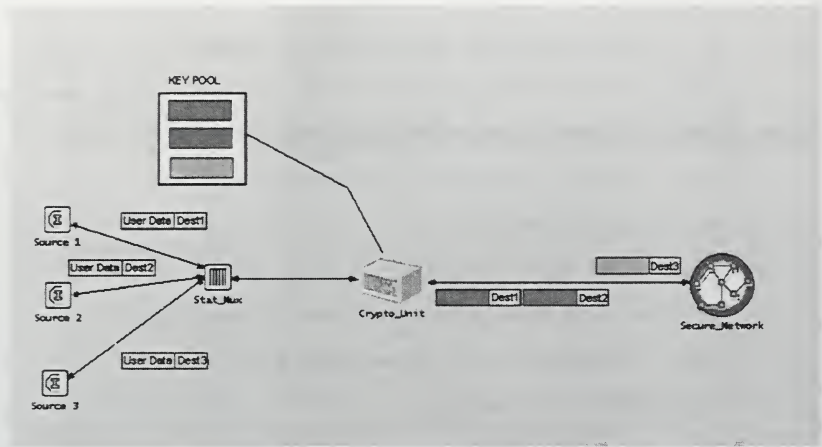
Figure 13 - Key-agile encryption system

D.   SECURE CALL ESTABLISHMENT

1.   Overview

The Enigma2 encryption effort creates significant complexity to the call setup efforts. This complexity is a result of negotiating three types of security related messages in addition to the regular ATM call setup process. The first negotiation that has to happen is an exchange of authentication in the form of a certificate. The second negotiation is a key exchange between the encryption devices. The third message is a handshake to implement the traffic encryption. There are two different types of protocols used in this system: point-to-point connection and point-to-multipoint connection protocols.

2. Point-to-Point Connection Protocol

This protocol design deals with the call setup process. The notation used to describe the protocol is:

Ca = A's public key certificate

Na = A's challenge

Ea(M) = Encrypt M with A's public key

Sa(M) = Sign Hash of M with A's private key

Kab = Session key for flow from A to B

SI = SETUP information

ID = Identifier for data connection

Figure 14 shows the point-to-point call set up. Calling host is the first ATM switch. Called host is the second ATM switch. Crypto A and crypto B are the two encryption devices.[MCNC97]

3. Point-to-Multipoint Connection Protocol

The point-to-multipoint connection protocol is somewhat more complex than the point-to-point connection protocol. This thesis defers simulation of this protocol to future research.

40
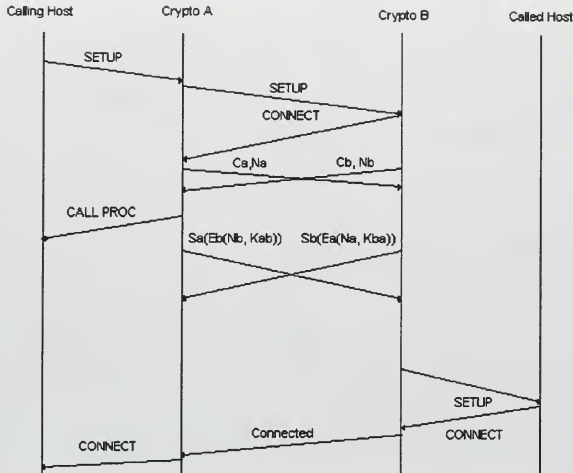
Figure 14 - Point to Point Call Setup

E.   INTRODUCTION TO KERBEROS

This section provides a basic description of the Kerberos authentication system and protocol. Information processing systems have grown ever more heterogeneous and decentralized and security solutions such as Kerberos have evolved as part of this process.

This thesis examines the Kerberos protocol to develop a model based on queuing analysis. This model will then be

bootstrapped to create a framework for modeling the CellCase service.

In a decentralized environment the internal security is essentially one of arbitrating who gains access to network services. This problem is characterized in three steps: identification, authentication and authorization [NEUM 93].

F. KERBEROS AUTHENTICATION SYSTEM

1. Concept

The Kerberos system has the mission of providing authentication and authorization services. The Kerberos system is centered on the concept of the ticket. The Kerberos ticket is a cryptographically generated token that grants access for a function or a device. Kerberos was developed as a part of Project Athena, at the Massachusetts Institute of Technology (MIT).

2. The Sequence of Steps in the implementation of the Kerberos Protocol

The sequence of steps executed in the implementation of the Basic Kerberos protocol is shown in Figure 15. The first step is simply a request from the client to the ticket-granting server (TGS) for a ticket to use a particular resource.
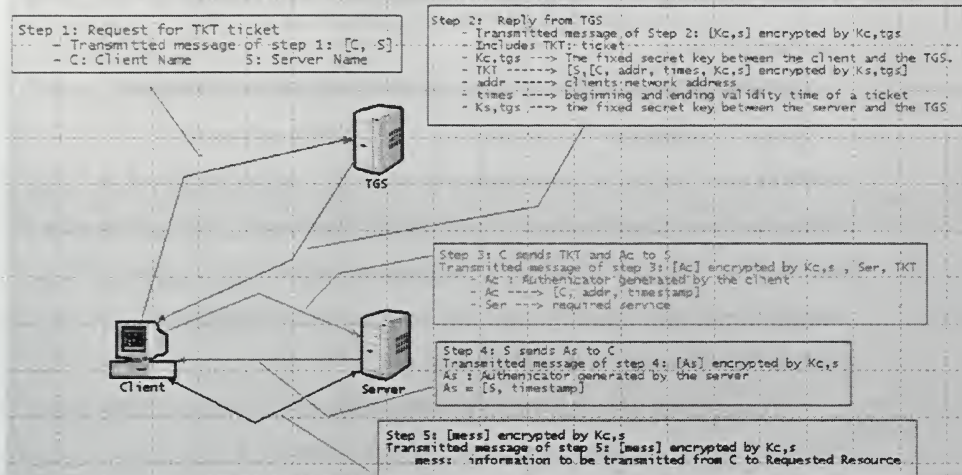
Figure 15 - Basic Kerberos Protocol

The second step is the reply from the TGS to the client. The TGS will respond with the session key $K_{c,s}$ that will be encrypted by the client key encryption key $K_{c, TGS}$. The response of the TGS in step two also includes a ticket, TKT, which consists of the key for communicating between the client and server $K_{c,s}$, its validity interval, the client name C and the address of C. The message with TKT is encrypted by another key $K_{s,TGS}$, which is the fixed symmetric key between the server and the TGS.

The third step is where the client C sends the ticket TKT and an authenticator Ac to the server S. This message is encrypted by $K_{c,s}$. Step 4 transmits A_s encrypted by $K_{c,s}$ from

43

the server back to the client. Step 5 concludes the process with the transmission of a secure message from C to S encrypted by $K_{c,s}$.

3. SUMMARY

This chapter provided a basic description of the contemporary ATM security framework and security solutions. The call setup sequence described here is particularly noteworthy, as that will be one of the three simulation models presented in this thesis.

This chapter also provided a very brief description of the Kerberos authentication protocol. The Kerberos protocol will be modeled in succeeding chapters as a core example of an OPNET cryptographic performance model based on queues.

V. PERFORMANCE ANALYSIS OF SECURE ATM CHANNELS AND KERBEROS AUTHENICATION SERVICE

A. KERBEROS AUTHENICATION PROTOCOL PERFORMANCE ANALYSIS

1. Queuing Analysis of Kerberos Protocol

A very flexible performance model of the Kerberos protocol has been given by El-Hadidi, Hengazi, and Aslan [ELHA 97]. This thesis will restate this model and expand on the published results. This study will also reference [ELHA 97] in the algorithm developed in the following chapter to model the CellCase ATM encryption unit.

Figure 16 shows a time sequence diagram from [ELHA 97] that shows the time sequence in serving a client request in the Kerberos environment.
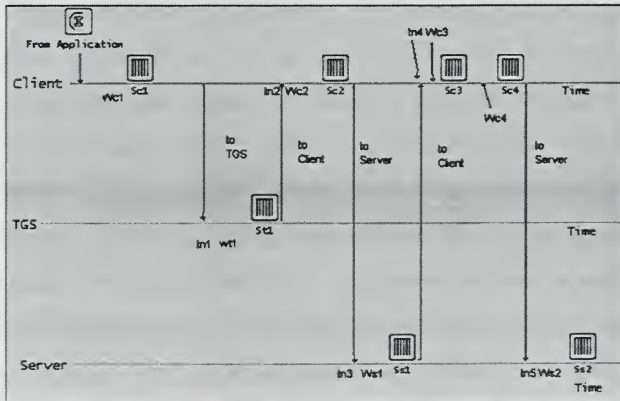
Figure 16 - Time Sequence Diagram for Kerberos Network

45

The time sequence begins with an application issuing a request for accessing a specific server. This request is forced to wait a time W_{c1} before the client can service the request in the time interval S_{c1} producing an unencrypted message. At the end of the interval S_{c1} the unencrypted request is sent to the Ticket Granting Server (TGS) and arrives at time t_{n1}, waits for W_{t1} to be served by the TGS and stays for the interval S_{t1} to be served by the TGS processor. The result of this is an encrypted response message. At the end of the interval S_{t1} the request is then sent back to the client and arrives at time t_{n2}, waits for W_{c2} to be served, and undergoes a service time of S_{c2} to service the encrypted message. The output from the client is an encrypted ticket and an authentication message that goes back to the server at time t_{n3}. This message arrives at the desired server and waits for a time W_{s1} after which it is processed during a time S_{s1} while the server produces and encrypts and timestamp. The client then waits for the timestamp to come from the server that arrives at time t_{n4}. The message waits at the client for a time W_{c3} before it is served for a duration of S_{c3} for the client to decrypt the timestamp. The client then encrypts the message M in the interval W_{c4} and sends the message M to the server where it arrives at t_{n5}, waits for a time W_{s2}, and gets processed by the server in the interval S_{s2}.

.

46

For the Kerberos protocol the following message lengths are assumed:

    Key = 8 bytes

    Ticket = 32 bytes

    Authenticator = 24 bytes

    Timestamp = 8 bytes

    Mean message length = 125 bytes

    Based on the above the service times are obtained:

    St1 = (8+32)*8/TDES = 320/TDES

    Sc2 = (8+24)*8/TDES = 256/TDES

    Ss1 = (32+24+8)*8/TDES = 512/TDES

    Sc3 = 8*8/TDES = 64/TDES

    Sc4 = 1000/TDES

    Ss2 = 1000/TDES

TDES is the rate of symmetric key encryption (DES) in bits/second. An exponential distribution is assumed for the exchanged message [ELHA 97]. These service times are the critical element in simulating the Kerberos system in OPNET.

2. OPNET Model of a Single TGS Network.

The OPNET simulation is implemented as two separate networks. The first is a simple ideal generator, a point-to-point simplex link and a sink that is used as a control simulation. The second is the same generator and sink with the queuing processes given the above service times. The ideal generator is sending traffic following a Poisson process. The queues that represent the negotiation process between the client, server and TGS are all active, concentrating, bit-oriented (acb) FIFO queues.

The structure of this model is very simple. We simply insert queues at the appropriate times and insert the service times that the computation of the cryptographic algorithm requires.

The service times are dependent on the rate of symmetric key encryption, TDES. A simple Microsoft Excel spreadsheet gives the results for an encryption rate of 150, 2,500 and 1Mbps bits per second by calculating the equations presented above. These resultant service times can be either placed in the queues as queuing delay or in the connecting links as link delay. The network and node level view of the traffic generators of the model is shown in Figure 17 below.
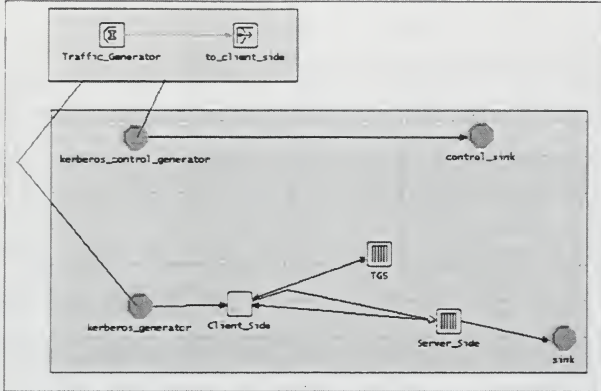
Figure 17 - Network View of OPNET Kerberos Simulation

The simulation process works as follows. The client object accepts the generated packet flow from the source and delays it for time S_{c1} before sending it to the TGS object. The TGS object delays the packet flow for a time S_{t1} and then sends it back to the client object. Upon receiving the packet flow from the TGS object the client object delays the packet stream for the time S_{c2} and forwards it to the server object where the packet stream is delayed for a time S_{s1} seconds. The server object then forwards the packet stream back to the client object where it is delayed by two queues, S_{c3} and S_{c4}, for S_{c3} and S_{c4} seconds, respectively. The client object then forwards the packet flow to the server object where it undergoes the final S_{s2} second delay.

49

The packet stream is then forwarded to the sink along a point-to-point link.

The client module is shown in the left box of Figure 18. The client module is a comprised of two receiver transmitter sets, one point-to-point transmitter, one point-to-point receiver and four acb queues. The TGS shown in the top right box of Figure 18 is composed of a logically associated receiver/transmitter and an acb queue. The bottom right of Figure 18 shows the server_side node model is composed of 1 logically associated receiver/transmitter, two acb queues, a point-to-point receiver and a point-to-point transmitter. The links are all generic point-to-point links with an unspecified bit rate and packet format.
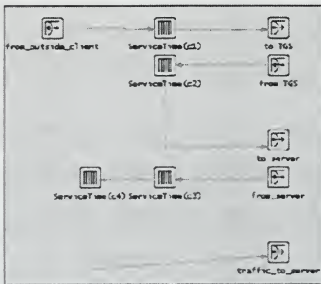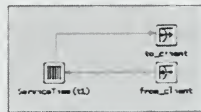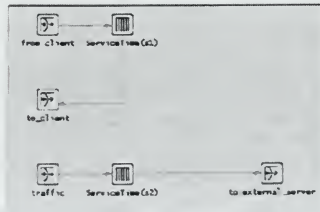
Figure 18 - Node Models of Client, Server and TGS

3. Results of Simulation

This simulation was designed to look at the mean throughput of a 'Kerberized' link compared with a control

50

link for different encryption times. Three different
encryption times were used: 150bps, 2.5kbps and 1Mbps.
Table 5 is a summary of the service times used.

	150 bps	.5 kbps	1 Mbps
Service Time S_{t1}	2.1333	0.1280	0.0003
Service Time S_{s1}	3.4133	0.2048	0.0005
Service Time S_{s2}	6.6667	0.4000	0.0010
Service Time S_{c2}	1.7067	0.1024	0.0003
Service Time S_{c3}	0.4267	0.0256	0.0001
Service Time S_{c4}	6.6667	0.4000	0.0010

Table 5 - Service Times per Encryption Rate

This paper focused on analyzing the control link
between the source and the sink and the 'Kerberized' link
between the to_external_server transmitter in the
server_side object and the sink. By comparing a control
traffic source with the delayed 'Kerberized' link, we hope
to achieve some insight into how this scheme affects traffic
patterns. To use the acb queues implemented in the model
the service times above are converted to service rates in
bps using the following formula:

$$ServiceRate = \frac{Traffic(inBits)}{ServiceTime},$$

Which yields the results shown in Table 6 for a traffic rate
of 1024 kbps.

	150 bps	2.5 kbps	1 Mbps
Service Rate SR_{t1}	480 bps	8000 bps	3.2 Mbps

Service Rate SR_{s1}	600 bps	10000 bps	4 Mbps
Service Rate SR_{s2}	300 bps	5000 bps	2 Mbps
Service Rate SR_{c2}	2400 bps	40000 bps	160 Mbps
Service Rate SR_{c3}	53.6 bps	2560 bps	1024 Kbps
Service Rate SR_{c4}	53.6 bps	2560 bps	1024 Kbps

Table 6 - Service Rates per Encryption Rate

Inserting these rates into the appropriate queues and running the simulation yields the following throughput results from OPNET modeler shown in Figures 19 to 22.
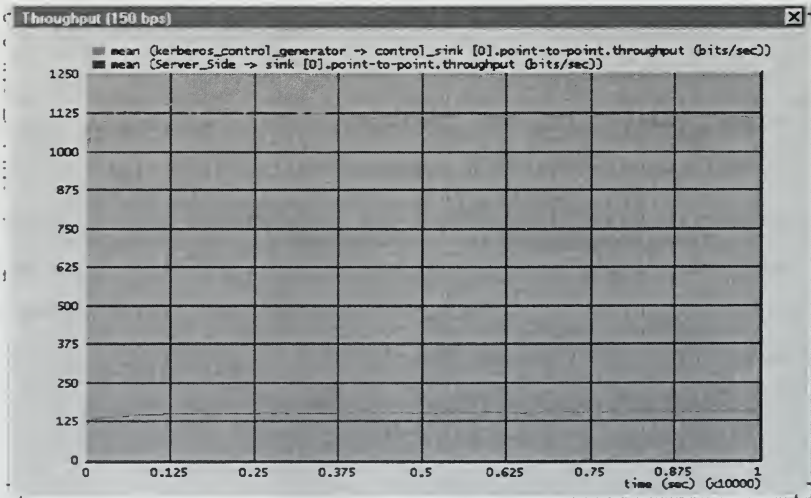
Figure 19 - Throughput at TDES = 150bps
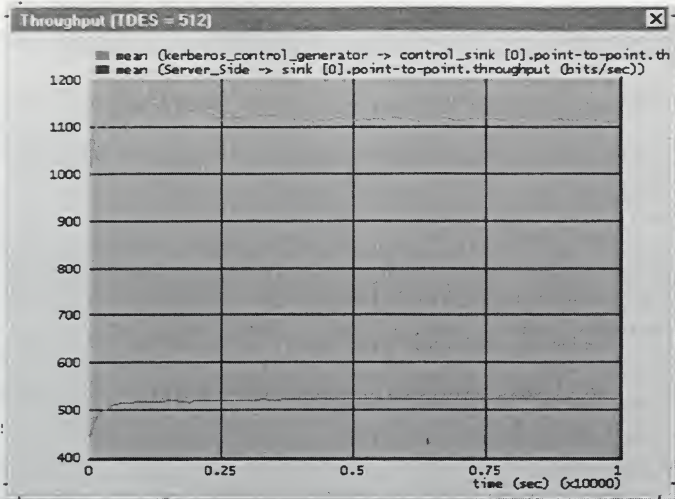
Figure 20 - Throughput at TDES = 512bps
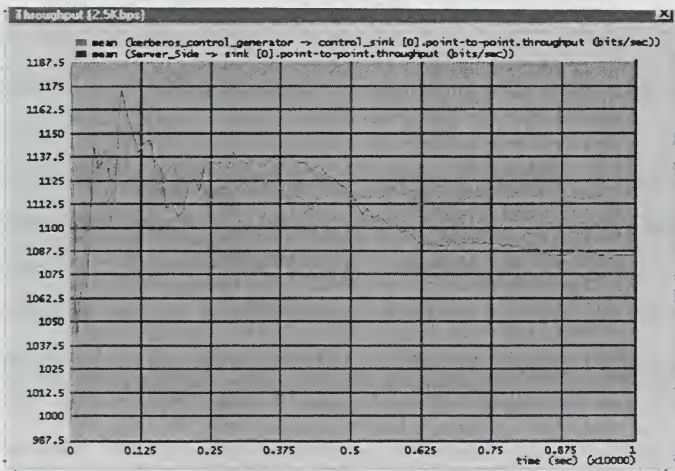
Figure 21 - Throughput at TDES = 2.5Kbps
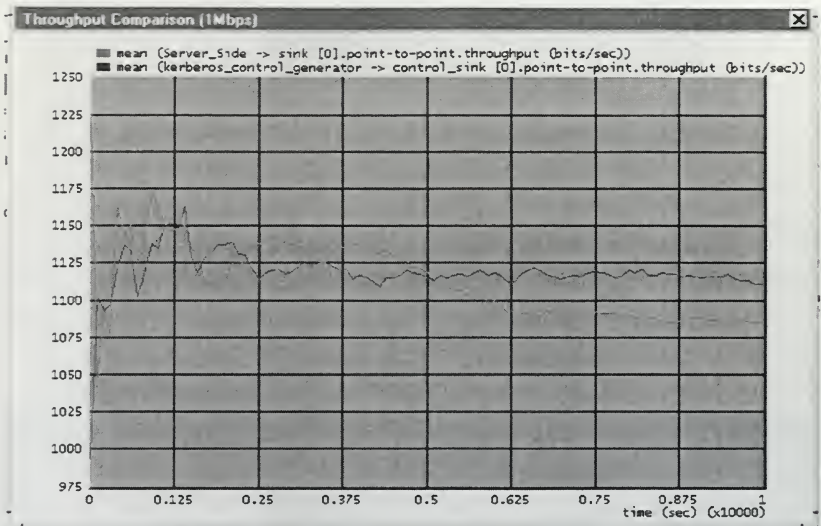
Figure 22 - Throughput at TDES = 1Mbps

In conclusion, these results indicate that the Kerberos model behaves in a manner consistent with intuition. The throughput is very low at the low encryption rates compared to the control link. The throughput matches the control link at higher encryption rates. The user of this model can add ideal generation sources and get a rough idea of what sort of performance his cryptographic engines should have in order to optimize throughput in a secure environment. It should be noted that the transition is nonlinear and the largest gains occur between 512 bps and 2500 bps. Above 2500 bps basically operates at line speed.

B.   SECURE ATM SYSTEM PERFORMANCE SIMULATION

    1.   CellCase Introduction

    The CellCase Security System consists of two encryption entities and one certificate server (CA) entity.   The cryptographic unit functions are intended to be transparent to normal operations of end-user and network equipment.   The CellCase45 supports T3 and E3 data rates and interfaces. The CellCase155 supports OC-3c interfaces [STEV 95].   The CellCase product line goal is to support data rates at the OC-12c (622 Mbps) level.

    The CellCase Security System creates a secure point-to-point connection in the following manner. First, a host follows the standard SETUP-CONNECT procedure.   The calling cryptographic device (crypto A) checks the bandwidth reservation for the connection; if the connection has sufficient bandwidth, the call setup procedure continues. This paper assumes that sufficient bandwidth is available.

    When a connection has been established each cryptographic device will send a certified exchange message, containing its own public key certificate and a nonce (Cx, Nx).   At this point each cryptographic device concurrently verifies the certificate given by their peer through a certificate authority (CA).   After receiving its peer's public key certificate message, each cryptographic device

proceeds concurrently to verify the certificate, check access permissions, and generate a key distribution message (KDM) consisting of its own sending session key and the nonce received by the peer, encrypted with the peer's public key and signed using its own private key. The KDM is then sent to the peer cryptographic device on the established connection. When the KDM exchange is finished, each cryptographic device verifies the signature and decrypts the message using its own private key. After the KDM is decrypted, cryptographic device B now installs the sending and receiving session keys and forwards the original SETUP message from the called host. It sends a CONNECTED message to cryptographic device A. Upon receipt of the CONNECTED message, cryptographic device A installs the session keys and then sends a CONNECT message back to the calling host, which completes the secure connection. Figure 23 shows the call setup procedure.

2. Secure ATM System Performance Metrics

a) *Encryption Speed*

A critical aspect to the performance analysis of a secure ATM system is the encryption speed of the component
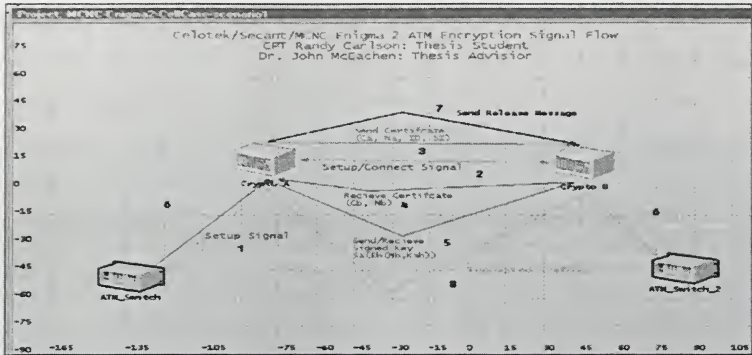
Figure 23 - Call Setup Procedure of the CellCase System

during both call setup and traffic phases of the calling
process. Stevenson, Hillary and Byrd[STEV 95] indicate that
the ability to generate and switch keys on a cell by cell
basis is the main performance concern in achieving cell
transparency in a key-agile system. This problem is
mitigated in the CellCase approach by using a key table and
by limiting the possible number of active VCs. The ATM
encryption system proposed by Wigelt and Rahman [WEIG 95]
uses encryption speed as a key concern in their simulation
of secure ATM networks.

In each case encryption speed determines the setup
time in the call setup phase and cell transfer delay in the
traffic phase. Results from simulations show that the added

delay for the encrypted ATM cells consisted solely of the delay due to encryption and decryption, when the encryption speed was greater than the segmentation delay at the AAL.

b)   Cell Transfer Delay (CTD)

The cell transfer time is a critical metric that is the major throughput consideration.   This is the time delay experienced by cells using the connection as measured on an end-to-end basis.

c)   Authentications per Key

The CellCase system has no hard requirement for a one-to-one relationship between two sites and a particular key. The equipment can be configured to allow multiple authentications based on one successful key exchange.   The setup time required will decrease as the ratio of the number of authentications to keys goes up.

3.   CellCase ATM Encryptor OPNET simulation

The OPNET model presented here is one of the Call Setup sequence of delays involved in the CellCase encryption system. The structure of this model is very simple.   The model assumes that cryptographic negotiation can be best represented by service delays in the call setup process. Figure 24 shows the atm8_cellcase_mod_2 node model that is

the heart of this simulation effort. Figure 24 follows the call setup process for the CellCase process.
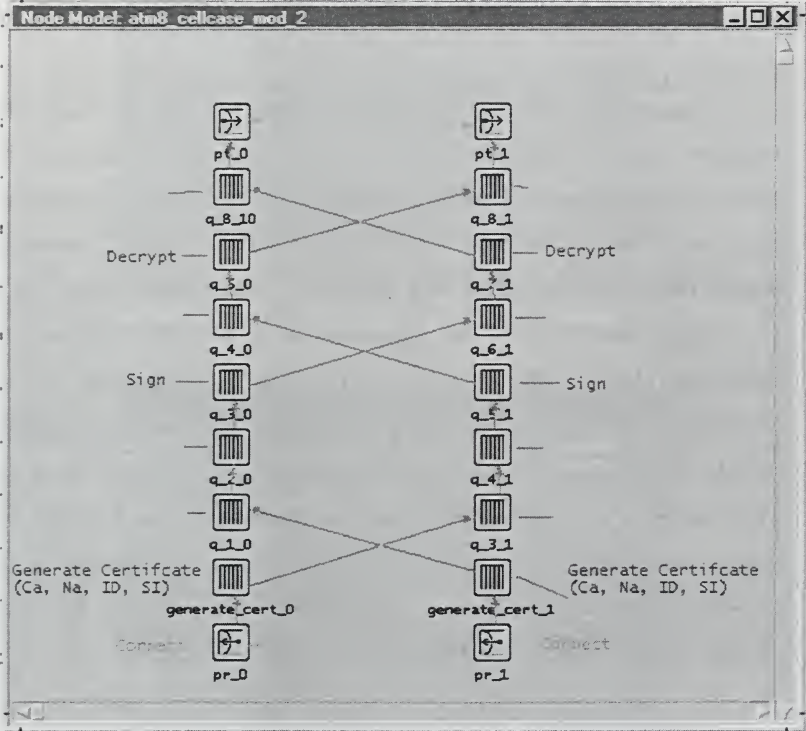
Figure 24 - CellCase Call Setup Scheme

The queue objects are acb_fifo queues that can delay traffic by delaying the service rate parameter in the queue.

The service rate that the queue model accepts is related to two factors, the time of encryption/decryption and the number of authorizations allowed per key. It is beyond the scope of this paper to perform an analysis of this relation, so a linear one is assumed to perform a very austere check of model validity.

Figure 25 shows the network level view of the OPNET simulation. Figures 26 and 27 respectively show the node level views of the generator and sink. The ideal generator has an interarrival rate of 0.01, which generates approximately a 64 kbps bit stream. The packet size is set at a constant rate of 53 bytes with a Poisson interarrival distribution. The service rate of the queues in the CellCase setup model, with the exception of the generate certificate, sign, and decrypt queues are all set to a very high rate of 155,000,000 bps which means that they will simply pass through packets without delay. Figure 30 also shows a control model that will be used to compare encrypted versus unencrypted results.
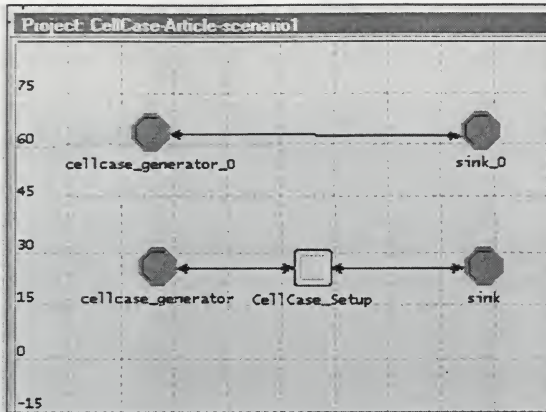
Figure 25 - Network Level view of Simulation
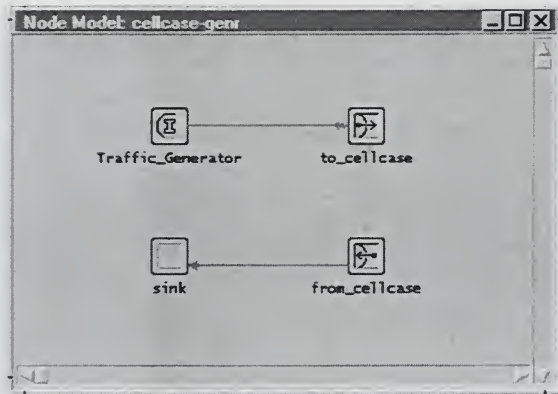
Figure 26 - Node Level view of CellCase Generator Object
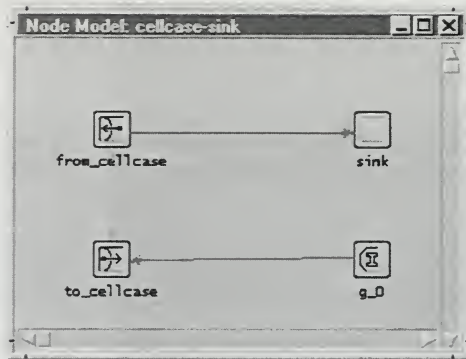
Figure 27 - CellCase Sink Model Object

The generate_certificate, sign and decrypt queues are subject to the following assumptions: 1) the service rate is limited to the rate of PK encryption, 2) the rate of PK encryption is 32 kbps and 3) the service rate has a linear relation to authorizations per key exchange. Table 7 below shows the input values into the queues.

Traffic (Bits per Second)	Rate of Pk Encryption	Service Rate	Number of Authorizations per key
64K	32 kb/s	32 kb/s	1
64K	32 kb/s	64 kb/s	2
64K	16 kb/s	16 kb/s	1
64K	16 kb/s	32 kb/s	2
64K	16 kb/s	64 kb/s	4

Table 7 - Input values into CellCase Model Queues

62

C.   RESULTS OF SIMULATION

The simulation was run with the above service rates in the generate certificate, sign and decrypt queues for a time of 100 seconds with the hope of observing a linear relationship between the service rate and the control link. The output is the mean throughput analysis of the link from the CellCase node model to the sink and from the control generator to the sink.   The results of the last three, with service rates of 16kbps, 32kbps, and 64kbps are shown here. The first two runs resulted in the same output as run number four and run number five. Figure 28 shows the results of a service rate of 16 kbps and one authorization per key.
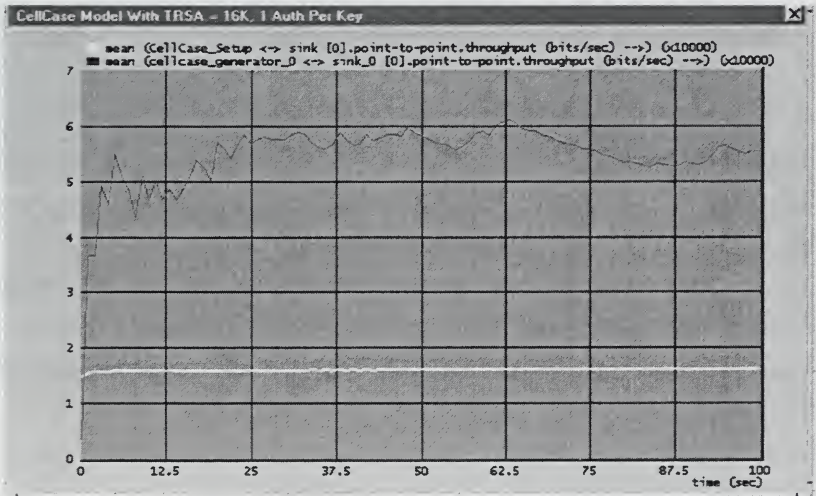
Figure 28 - CellCase Results with 16K Service Time and 1
Authorization per key

The control link hovers around 50,000 to 60,000 bps while the encrypted link hovers around 16 kbps. Figure 29 shows the notional results with 16 kbps and two authorizations per key.
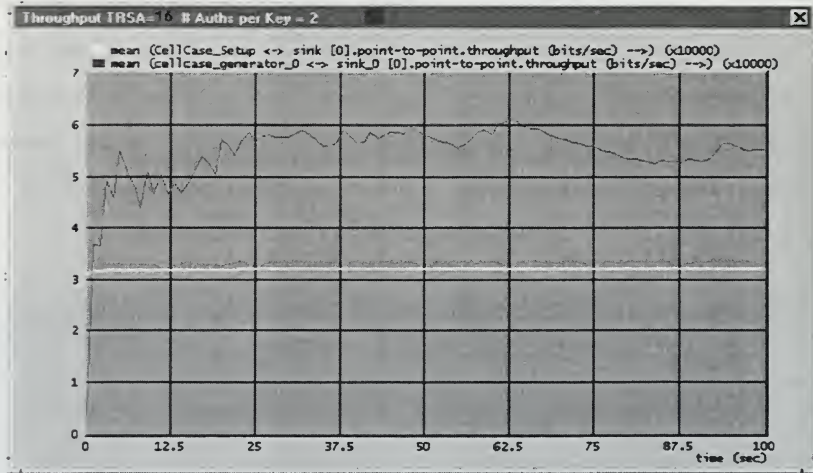
Figure 29 - CellCase Results with 16K Service Time and 2 Authorizations per key

The control link hovers around 50,000 to 60,000 bps while the encrypted link hovers around 32 kbps. Figure 30 shows the notional results with 16 kbps and four authorizations per key.

64
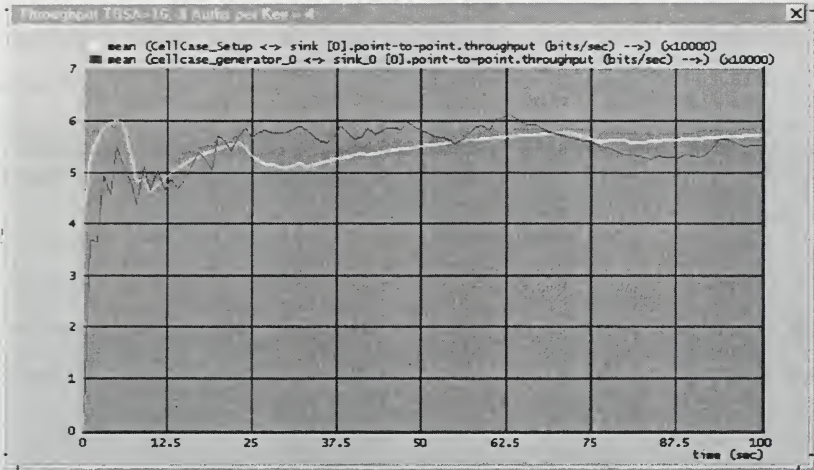
Figure 30 - CellCase Results with 16K Service Time and 4
Authorizations per key

With 16 kbps service time and four authorizations per
key the control link hovers around 50,000 to 60,000 bps
while the encrypted link hovers around the same rate: 64
kbps.

D.   SUMMARY

Figures   28,   29   and   30   show   the   same   general
relationship   as   the   three   models   shown   in   the   Kerberos
model.   The   mean   throughput   is   very   stable   on   the   low
throughput   runs   but   very   irregular   as   the   data   rate

65

increases. The last run shows a spike where the encrypted data actually outpaces the unencrypted data at the 60K data rate level. Further runs of these models may be needed to determine if this irregularity is due to the OPNET simulation engine or some other problem. This OPNET model extends the El-Hadidi, et. al [ELHA 97] queuing analysis into the CellCase product. If validated, this process can be used for many authentication services throughout DoD.

VI.   CONCLUSIONS

A.   ISSUES FOR ENCRYPTION MODELLING RESEARCH

1.   Call Setup time versus Authentication Policy

To make this a higher fidelity model further research should be done on the relationship between call setup time and the authentication key policy.   While researching this thesis, the author used the queuing analysis used of El-Hadidi et al. [ELHA 97] to look at the number of authorizations/key exchange versus the setup time.   The result for Kerberos is shown in Figure 31 and the result for CellCase is shown in Figure 32.
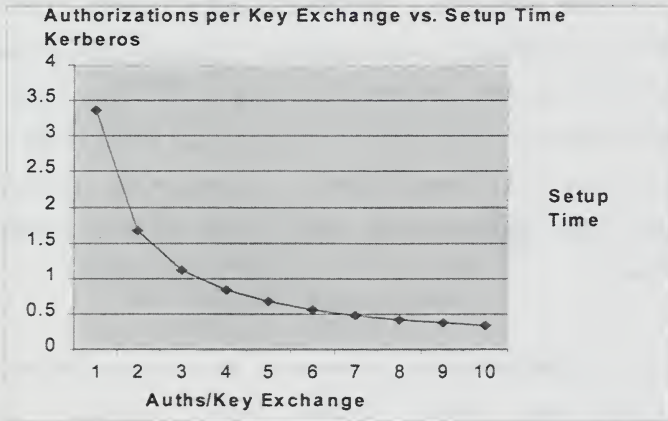
Figure 31 - Authorizations/Key Exchange versus Setup Time

**Authorizations per Key Exchange vs. Setup Time
CellCase**
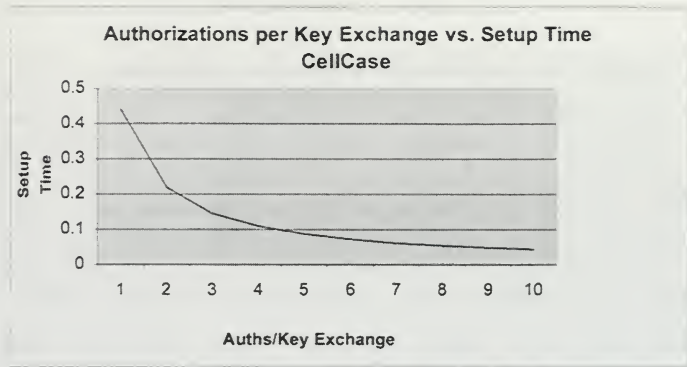
Figure 32 - Authorizations/Key Exchange versus Setup Time
CellCase

In the simulation of the CellCase model, this thesis assumed a linear relationship between the number of authorizations/key exchange and the call setup time. Using a M/G/1 queuing model to calculate setup time shows a faster drop-off of call setup time. It would be interesting to isolate this relationship and use it in the OPNET models provided.

2. Cryptographic Processing Rate versus Setup Time

The same model yielded some insights into the relationship between the time of encryption and the setup time. This relationship is shown, respectively, in Figure 33 and Figure 34 for the Kerberos and CellCase system.

68
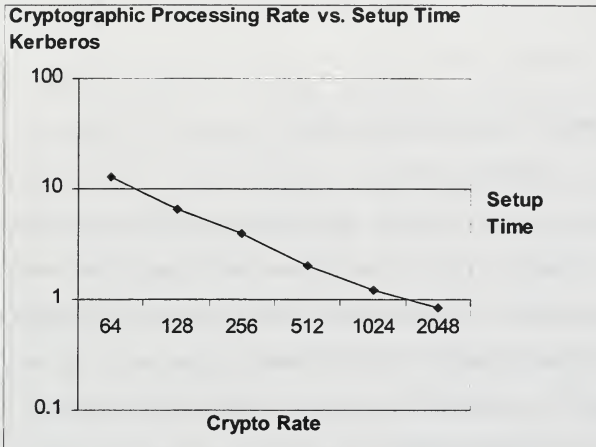
Figure 33 - Cryptographic Processing Rate vs. Setup Time
Kerberos
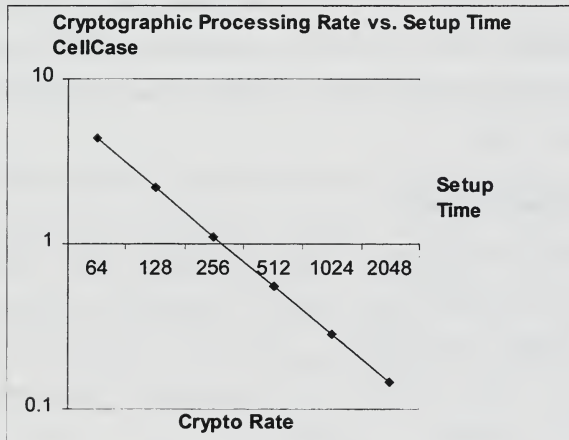
Figure 34 - Cryptographic Processing Rate vs. Setup Time
CellCase

This is a generally linear relationship. This relationship was the one used to calculate the service times in the CellCase model.

B. COMMENTS ON MODEL VALIDITY

1. Kerberos Model

The Kerberos Model presented here validates previous analytic studies of the Kerberos service. The next stage of this process is to conduct model validity studies with the actual Kerberos system and compare results. This cannot be considered a valid model until this is accomplished.

2. CellCase Model

The CellCase Model provides a framework to conduct analytic studies of the CellCase service. The next stage of this process is to test the CellCase model with the actual CellCase system and then integrate that model into the ATM model set (ams).

C. FUTURE RESEARCH

These models are in a very immature form but they are novel as the author has found no operational security models in the OPNET environment. There is a solid starting point to build higher fidelity cryptographic performance models from the lessons learned from this research. The performance of future DoD networking efforts will very probably need to factor in delay and throughput factors in

70

encrypted data networks, particularly as network managers scale PKI resources to larger and larger levels.

The next phase of this research would be twofold. First, we need to validate the Kerberos and CellCase OPNET models presented in this thesis. The Kerberos model can be adapted to fit DISA's ongoing PKI initiative. Taking these models and adapting them to a small PKI network like the Defense Travel Office PKI testbed would allow real world validation of the algorithms in this thesis and the OPNET model. Second, the models here are shown as very simple point-to-point networks. Research into adapting this model to a multiple server/client/authenticator network in the Kerberos OPNET model would yield valuable insights into the performance problems that a viable PKI authentication service such as the one DISA is now planning would have.

The performance implications that the current DISA PKI medium assurance initiative implies across the DISN are significant. The Kerberos model presented here can be adapted to model the small systems being fielded now such as the Defense Security Service, Navy Region San Diego and the Office of the Army Chief of Staff. A continuation of this research will be done, in one form or the other, simply because the performance ramifications of overlaying a PKI security structure will need to be thoroughly understood if

the user is to get quality of service and information assurance.

APPENDIX A. OPNET CODE FOR ACB_FIFO QUEUING MODEL

A. OPNET SOURCE CODE

```
/* Process model C form file: acb_fifo.pr.c */
/* Portions of this file copyright 1992, 1996, 1998 by MIL 3, Inc. */


/* This variable carries the header into the object file */
static const char acb_fifo_pr_c [] = "MIL_3_Tfile_Hdr_ 51D 30A opnet 7 375CB7A6 375CB7A6 1 zn8wp
Administrator 0 0 none none 0 0 none 0 0 0 0 0 0";


/* OPNET system definitions */
#include <opnet.h>
FSM_EXT_DECS


/* Header Block */

#define QUEUE_EMPTY     (op_q_empty ())
#define SVC_COMPLETION  op_intrpt_type () == OPC_INTRPT_SELF
#define ARRIVAL         op_intrpt_type () == OPC_INTRPT_STRM

/* End of Header Block */


#if !defined (VOSD_NO_FIN)
#undef   BIN
#undef   BOUT
#define  BIN            _fstack_local_info.last_line_passed = __LINE__ - _block_origin;
#define  BOUT    BIN
#define  BINIT   _fstack_local_info.last_line_passed = 0; _block_origin = __LINE__;
#else
#define  BINIT
#endif /* #if !defined (VOSD_NO_FIN) */


/* State variable definitions */
typedef struct
        {
        /* Internal state tracking for FSM */
        FSM_SYS_STATE
        /* State Variables */
        int                     server_busy;
        double                  service_rate;
        Objid                   own_id;
        } acb_fifo_state;
```

```
#define pr_state_ptr                ((acb_fifo_state*) SimI_Mod_State_Ptr)
#define server_busy                 pr_state_ptr->server_busy
#define service_rate                pr_state_ptr->service_rate
#define own_id                      pr_state_ptr->own_id

/* This macro definition will define a local variable called*/
/* "op_sv_ptr" in each function containing a FIN statement*/
/* This variable points to the state variable data structure, */
/* and can be used from a C debugger to display their values.        */
#undef FIN_PREAMBLE
#define FIN_PREAMBLE    acb_fifo_state *op_sv_ptr = pr_state_ptr;


/* No Function Block */

enum { _block_origin = __LINE__ };




/* Process model interrupt handling procedure */


void
acb_fifo (void)
        {
        int _block_origin = 0;
        FIN (acb_fifo ());
        if (1)
                {
                Packet*         pkptr;
                int        pk_len;
                double       pk_svc_time;
                int        insert_ok;

                FSM_ENTER (acb_fifo)

                FSM_BLOCK_SWITCH
                        {
                        /*-------------------------------------------------------------*/
                        /** state (init) enter executives **/
                        FSM_STATE_ENTER_FORCED (0, state0_enter_exec, "init", "acb_fifo () [init enter
                        execs]")
                                {
                                /* initially the server is idle   */
                                server_busy = 0;

                                /* get queue module's own object id    */
                                own_id = op_id_self ();

                                /* get assigned value of server           */
                                /* processing rate                                        */
                                op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
```

74

```
                    }


/** state (init) exit executives **/
FSM_STATE_EXIT_FORCED (0, state0_exit_exec, "init", "acb_fifo () [init exit execs]")
        {
        }


/** state (init) transition processing **/
FSM_INIT_COND (ARRIVAL)
FSM_DFLT_COND
FSM_TEST_LOGIC ("init")

FSM_TRANSIT_SWITCH
        {
        FSM_CASE_TRANSIT (0, 1, state1_enter_exec, ;)
        FSM_CASE_TRANSIT (1, 2, state2_enter_exec, ;)
        }
        /*---------------------------------------------------------*/



/** state (arrival) enter executives **/
FSM_STATE_ENTER_FORCED (1, state1_enter_exec, "arrival", "acb_fifo () [arrival
enter execs]")
        {
        /* acquire the arriving packet                     */
        /* multiple arriving streams are supported. */
        pkptr = op_pk_get (op_intrpt_strm ());


        /* attempt to enqueue the packet at tail*/
        /* of subqueue 0.                                  */
        if (op_subq_pk_insert (0, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
            {
            /* the inserton failed (due to to a    */
                    /* full queue) deallocate the packet.    */
            op_pk_destroy (pkptr);

                    /* set flag indicating insertion fail     */
                    /* this flag is used to determine          */
                    /* transition out of this state           */
            insert_ok = 0;
            }
        else{
                    /* insertion was successful                */
            insert_ok = 1;
            }


        }


/** state (arrival) exit executives **/
```

75

```
FSM_STATE_EXIT_FORCED (1, state1_exit_exec, "arrival", "acb_fifo () [arrival exit
execs]")
        {
        }


/** state (arrival) transition processing **/
FSM_INIT_COND (!server_busy && insert_ok)
FSM_DFLT_COND
FSM_TEST_LOGIC ("arrival")

FSM_TRANSIT_SWITCH
        {
        FSM_CASE_TRANSIT (0, 3, state3_enter_exec, ;)
        FSM_CASE_TRANSIT (1, 2, state2_enter_exec, ;)
        }
        /*----------------------------------------------------------*/



/** state (idle) enter executives **/
FSM_STATE_ENTER_UNFORCED (2, state2_enter_exec, "idle", "acb_fifo () [idle
enter execs]")
        {
        }


/** blocking after enter executives of unforced state. **/
FSM_EXIT (5,acb_fifo)


/** state (idle) exit executives **/
FSM_STATE_EXIT_UNFORCED (2, state2_exit_exec, "idle", "acb_fifo () [idle exit
execs]")
        {
        }


/** state (idle) transition processing **/
FSM_INIT_COND (ARRIVAL)
FSM_TEST_COND (SVC_COMPLETION)
FSM_TEST_LOGIC ("idle")

FSM_TRANSIT_SWITCH
        {
        FSM_CASE_TRANSIT (0, 1, state1_enter_exec, ;)
        FSM_CASE_TRANSIT (1, 4, state4_enter_exec, ;)
        }
        /*----------------------------------------------------------*/



/** state (svc_start) enter executives **/
FSM_STATE_ENTER_FORCED (3, state3_enter_exec, "svc_start", "acb_fifo ()
svc_start enter execs]")
```

76

```
            {
            /* get a handle on packet at head of subqueue 0 */
            /* (this does not remove the packet)                */
            pkptr = op_subq_pk_access (0, OPC_QPOS_HEAD);

            /* determine the packets length (in bits)           */
            pk_len = op_pk_total_size_get (pkptr);

            /* determine the time required to complete          */
            /* service of the packet                                 */
            pk_svc_time = pk_len / service_rate;

            /* schedule an interrupt for this process           */
            /* at the time where service ends.                      */
            op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

            /* the server is now busy.                              */
            server_busy = 1;


            }


/** state (svc_start) exit executives **/
FSM_STATE_EXIT_FORCED (3, state3_exit_exec, "svc_start", "acb_fifo () [svc_start
exit execs]")
            {
            }


/** state (svc_start) transition processing **/
FSM_TRANSIT_FORCE (2, state2_enter_exec, ;)
            /*---------------------------------------------------------*/


/** state (svc_compl) enter executives **/
FSM_STATE_ENTER_FORCED (4, state4_enter_exec, "svc_compl", "acb_fifo ()
[svc_compl enter execs]")
            {
            /* extract packet at head of queue; this */
            /* is the packet just finishing service   */
            pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);

            /* forward the packet on stream 0, causing       */
            /* an immediate interrupt at destination./
            op_pk_send_forced (pkptr, 0);

            /* server is idle again.                              */
            server_busy = 0;
            }


/** state (svc_compl) exit executives **/
```

```
                        FSM_STATE_EXIT_FORCED (4, state4_exit_exec, "svc_compl", "acb_fifo ()
                        [svc_compl exit execs]")
                                {
                                }


                        /** state (svc_compl) transition processing **/
                        FSM_INIT_COND (!QUEUE_EMPTY)
                        FSM_DFLT_COND
                        FSM_TEST_LOGIC ("svc_compl")

                        FSM_TRANSIT_SWITCH
                                {
                                FSM_CASE_TRANSIT (0, 3, state3_enter_exec, ;)
                                FSM_CASE_TRANSIT (1, 2, state2_enter_exec, ;)
                                }
                                /*--------------------------------------------------------*/



                        }


                FSM_EXIT (0,acb_fifo)
                    }
        }


Compcode
acb_fifo_init (void ** gen_state_pptr)
        {
        int _block_origin = 0;
        static VosT_Cm_Obtype        obtype = OPC_NIL;

        extern void                                Vos_Vnop (void);

        FIN (acb_fifo_init (gen_state_pptr))

        if (obtype == OPC_NIL)
                {
                /* Initialize memory management */
                if (Vos_Catmem_Register ("proc state vars (acb_fifo)",
                        sizeof (acb_fifo_state), Vos_Vnop, &obtype) == VOSC_FAILURE)
                        FRET (OPC_COMPCODE_FAILURE)
                }

        *gen_state_pptr = Vos_Catmem_Alloc (obtype, 1);
        if (*gen_state_pptr == OPC_NIL)
                FRET (OPC_COMPCODE_FAILURE)
        else
                {
                /* Initialize FSM handling */
                ((acb_fifo_state *)(*gen_state_pptr))->current_block = 0;

                FRET (OPC_COMPCODE_SUCCESS)
```

78

```
                }
        }


void
acb_fifo_diag (void)
        {
        int _block_origin = __LINE__;

        FIN (acb_fifo_diag ())


        FOUT;
        }



void
acb_fifo_terminate (void)
        {
        int _block_origin = __LINE__;

        FIN (acb_fifo_terminate (void))

        if (1)

                {
                Packet*         pkptr;
                int             pk_len;
                double          pk_svc_time;
                int             insert_ok;

                /* No Termination Block */

                }
        Vos_Catmem_Dealloc (pr_state_ptr);

        FOUT;
        }


/* Undefine shortcuts to state variables to avoid */
/* syntax error in direct access to fields of */
/* local variable prs_ptr in acb_fifo_svar function. */
#undef server_busy
#undef service_rate
#undef own_id



void
acb_fifo_svar (void * gen_ptr, const char * var_name, char ** var_p_ptr)
        {
        acb_fifo_state              *prs_ptr;
```

```
        FIN (acb_fifo_svar (gen_ptr, var_name, var_p_ptr))

        *var_p_ptr = (char *)VOS_NIL;
        prs_ptr = (acb_fifo_state *)gen_ptr;

        if (Vos_String_Equal ("server_busy" , var_name))
                {
                *var_p_ptr = (char *) (&prs_ptr->server_busy);
                goto func_return;
                }
        if (Vos_String_Equal ("service_rate" , var_name))
                {
                *var_p_ptr = (char *) (&prs_ptr->service_rate);
                goto func_return;
                }
        if (Vos_String_Equal ("own_id" , var_name))
                {
                *var_p_ptr = (char *) (&prs_ptr->own_id);
                goto func_return;
                }
func_return:


        FOUT;
        }
```

LIST OF REFERENCES

[CHEN 96] Shaw-Cheng Chuang: "Securing ATM Networks", 3rd ACM Conference on Communications and Computer Security, pp 19-21, 1996

[COHE 96] Cohen G., Kamenel B., and Kubic C., "Security for Integrated IP-ATM/Tactical-Strategic Networks", *IEEE Conference Proceedings, Military Communications Conference*, 1996. MILCOM '96, Volume: 2 , Page(s): 456 -460 vol.2

[CONT 98] Office of Secretary of Defense, "Contingency C4ISR Handbook for Integrated Planning", 1998

[ELHA 97] El-Hadidi M., Hegazi N., and Aslan H., "Performance Analysis of the Kerberos Protocol in a Distributed Environment", 2nd IEEE Symposium on Computers and Communications, Alexandria, Egypt, pp. 235-239, June 1997

[FEGH 99] Feghhi, J. and others, *Digital Certificates: applied Internet security*, pp. 27-91, Addison-Wesley, 1999

[KUMA 95] Kumar, B., *Broadband Communications: A professionals guide to ATM, Frame Relay, SMDS, SONET and BISDN*, pp. 141-158, McGraw-Hill, 1995

[LIAN 96] Liang, Donglin, "A Survey on ATM Security", http://www.cis.ohio-state.edu/~jain/cis788-97/atm security/index.htm, 1996

[MCNC 98] MCNC Information Technologies Advanced Networking Research and Secant Network Technologies and Portland State University, Report Number A802, *Final Report Key Agile Cryptographic Systems and Covert Channels in Broadband Public Systems*, by F. Gong, pp. 7-49, July 1998

[NEUM 94] Neuman B. and Ts'o T., "Kerberos: An Authentication Service for Computer Networks", *IEEE Communications*, pp. 33-38, September 1994

[OPPL 96] Oppliger, R., *Authentication Systems for Secure Networks*, pp. 29-56, Artech House, 1996

[SHWB 95] Stevenson, D. Hillery, N. Winkelstein, D. and Byrd, G. "Design of a Key Agile Cryptographic System for OC-12c Rate ATM," paper presented at the Symposium on Network and Distributed System Security, San Diego, CA, Feb 1995

[SPEC 97] The ATM Forum, ATM Security System Specification, Version 1.0 AF-SEC-0100.000, pp. 15-37, 1997

[STEV 95] Stevenson D., Hillery N. and Byrd G. "Secure Communications in ATM Networks", *Communications of the ACM*, pp. 45-52, February 1995

[STIN 95] Stinson, D. R., *Cryptography: Theory and Practice*, pp. 70-157, CRC Press, 1995

[THPW 98] Tarman T. D., Hutchinson R. L., Pierson L. G., and Witzke E. L., "Algorithm Agile Encryption in ATM Networks", *Proceedings of the IEEE Computer Society*, pp. 57-64, September 1998

[WEIG 95] Weigelt J.H. and M.H. Rahman, "Securing Asynchronous Transfer Mode Based Networks Through the Use of Encryption" IEEE Canadian Conference on Electrical and Computer Engineering, 1995. Canadian Conference on Volume: 1 , Page(s): 428 -431 vol.1

INITIAL DISTRIBUTION LIST