PERFORMANCE IMPACT OF ENCRYPTION ALGORITHMS ON KERBEROS NETWORK AUTHENTICATION PROTOCOL

By

ANUSH KRISHNAMURTHY

Bachelor of Engineering

University of Madras

Chennai, India

2000

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May 2006

PERFORMANCE IMPACT OF ENCRYPTION ALGORITHMS ON KERBEROS NETWORK AUTHENTICATION PROTOCOL

Thesis Approved:

Thesis Adviser
Dr. Johnson Thomas

Dr. G E Hedrick

Dr. John Chandler

A. Gordon Emslie
Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Dr. Johnson Thomas for his guidance and assistance at Oklahoma State University. I would also like to thank my committee members, Dr. G.E. Hedrick and Dr. John Chandler, for their helpful contributions and advice.

Heart-felt thanks goes to my parents and my girlfriend for their unending encouragement and emotional support throughout the years.

Finally I would like to thank all my friends who stood beside me with their unfailing and indispensable support.

Preface

Performance, in terms of user response time and the utilization of processing and communication resources, is an important factor to be considered when designing security authentication protocols. The growth of Internet subscription and network interconnectivity has created an environment in which information systems have a greater exposure to unauthorized access. To meet the challenge of protecting information and computing systems, researchers and developers have focused attention on authentication protocols. An authentication protocol helps assure that only properly authorized users can access to information systems by positively identifying the user. Unfortunately, a protocol that provides a high level of assurance often consumes significant processing and communication resources. As a result, improved security may exact a price in the servers and networks which are overburdened already and results lengthy delays for users. This thesis presents a performance analysis of one of the authentication protocols by varying the encryption algorithms and there by understand the performance implications of the ciphers on the protocol.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

Chapter 1

Introduction

1.1 Computer Networks

Just as there has been an unstoppable trend towards having additional computing power at one's fingertips, the world of networked computing has similarly advanced at an amazing pace, approximately doubling in connectivity and reach, every year. This implies that the number of computer users connected to the network next year is likely to exceed the total number of network-connected people in each previous year added together. This rate of growth has caused revolutionary changes in network technology development and has created social, business, and legal advances for integrating the technology into everyday life. The rapid growth of Internet subscription and network interconnectivity has created an environment in which information systems have a greater exposure to unauthorized access. Security becomes a critical issue in reliable network computing these days. Security of the information is at stake and is easy to be destroyed by unauthorized intrusions. Network-scanning events are becoming common events of life these days.

1.2 Network Authentication

Authentication is a fundamental process in securing information resources since it establishes the identity of the system or user wishing to gain access to the resource. Authentication is of major importance to the security of open networks. In open networks, the identities of communicating parties cannot be assumed, but must be authenticated (i.e., proven). Traditionally strong forms of authentication are not available in most operating system networking software. Instead, reserved ports or passwords are used, each of which might be easily compromised. Tools to sniff passwords off of the network are in common use by malicious hackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable. Worse yet, client/server applications rely on the client program to be "honest" about the identity of the user who is using it. Other applications rely on the client to restrict its activities to those, which is allowed to do, with no other enforcement by the server. One of the most basic authentication protocols is a password challenge – a user is asked to demonstrate knowledge of a pre-established password to prove his or her identity. Simple password security is a weak mechanism subject to several types of attacks. To meet the challenge of protecting information and computing systems, researchers and developers have focused attention on authentication protocols. Many authentication protocols have been developed to try to improve on basic password security. Kaufman et al. [3] describe both the weaknesses of password authentication and more sophisticated and secure protocols such as Kerberos.

1.3 Cryptographic Algorithms

Mature authentication protocols are based on cryptographic algorithms. Encoding the contents of a message in such a way that hides its content from outsiders is called Encryption. Then encrypted message is called ciphertext. The process of retrieving the plain text from the ciphertext is called Decryption. Encryption and Decryption usually make use of a key, and the coding method is such that decryption can be performed only by knowing the proper key. There are two classes of key-based algorithms, symmetric (or secret key) and asymmetric (or public key) algorithms. The difference is that symmetric algorithms use the same key for both the operations (or the decryption key is easily derived from the encryption key), whereas asymmetric algorithms use different keys for encryption and decryption, and decryption key cannot be derived from the encryption key.

Symmetric algorithms can be divided into stream ciphers and block ciphers. Stream ciphers can encrypt a single bit of plaintext at a time, while block ciphers can take a number of bits and encrypt them as a single unit. The mostly used symmetric ciphers are Data Encryption Standard and Triple-DES. Advanced Encryption Standard is the newest standard and is expected to replace the older versions to become the mostly used encryption algorithms.

Asymmetric algorithms also called as Public Key Cryptography permit the encryption key to be public, allowing anyone to encrypt with the key. Only the proper recipient who knows the decryption key can read the message. The encryption key is also called the public key and the decryption key is the private or secret key. RSA is probably the best known asymmetric encryption algorithm used. Elliptic Curve Cryptography is also another advanced asymmetric encryption algorithm which is being considered as an alternative option to RSA in many fields.

Kerberos is a mature network authentication protocol based on secret key cryptography. Matured protocols today have a wide range of proposals to extend them in ways that were not originally envisioned by their authors. Kerberos is a good candidate for an analysis with the encryption algorithms discussed above.

The protocol designer faces many decisions about the use of encryption that affect the level of assurance and performance. Advanced Encryption Standard (AES) is now the latest approved standard for symmetric encryption algorithm. While benchmarks of the algorithm has been made, the performance in our authentication environment needs to be analyzed. We analyze in more general terms the performance characteristics of the authentication protocol with varied encryption algorithms to better understand the assurance and performance we get with the variant.

The rest of this thesis is organized as follows. The next chapter describes briefly the Kerberos Authentication protocol. Chapter 3 talks about the encryption algorithms used in Kerberos currently and the variants available based on encryption algorithms. After its major standardization, we discuss the typical issues in current Kerberos based on cryptography in chapter 4. Chapter 5 discusses the proposed variants for Kerberos and

Chapter 6 talks about the performance evaluation for the approaches. Chapter 7 concludes on the approaches and provides pointers to future work.

Chapter 2

Encryption & Kerberos Authentication Protocol

Kerberos acts as a dependable authentication service with the use of conventional cryptography. The basic version of Kerberos used the stream encryption ciphers, which can be simulated by commonly available block encryption ciphers, such as Data Encryption Standard, in conjunction with block chaining and checksum methods [1]. We start with the basics of cryptography and get into the variations of Kerberos based on some of the standards.

2.1 Basics of Cryptography

People mean different things when they talk about cryptography. A method of encryption and decryption is called cipher. All algorithms use a key to control encryption and decryption; a message can be decrypted only if the key matches the encryption key. There are two classes of key-based encryption algorithms, symmetric (or secret-key) and asymmetric (or public-key) algorithms. The main difference between the two is that the symmetric algorithm uses the same key for both encryption and decryption or sometimes the decryption key is derived from the encryption key, whereas asymmetric algorithms use a different key for encryption and decryption, and one cannot be derived from the other.

2.2 Symmetric Key Cryptography

Symmetric algorithms can be divided into two as stream ciphers and block ciphers. A stream cipher does the encryption for a single bit of a plaintext at a time, and the block cipher can take a number of bits and encrypt them as a single unit. Symmetric ciphers are the most straightforward approach to data encryption. They are mathematically less complicated than their counterparts and have been used for many centuries. There are two common symmetric ciphers widely used in today's information systems.

2.2.1 Data Encryption Standard (DES)

The data encryption standard (DES) is an algorithm developed in mid 1970's. It was turned into a standard by the US National Institute of Standards and Technology (NIST), and was adopted by several other governments worldwide. It was and still is widely used, especially in the financial industry.

DES is a block cipher with a 64-bit block size. It used 56-bit keys. 8-bits are used for parity. This makes this algorithm susceptible to exhaustive key search with modern computers and special-purpose hardware. DES is still strong enough to keep most random hackers and individuals out, but is easily breakable with special hardware by government, criminal organizations, or major corporations. DES is getting too weak, and is not used in new applications these days.

A variant of DES, TRIPLE-DES (also 3DES) is based on applying 3 stages of DES with a separate key for each stage. So the key length in 3DES is 168 bits. The 3DES is much stronger than its predecessor, however, it is rather slow compared to some new block ciphers. Even though DES and its variants seem to be have little interest for applications today, there are many reasons for considering it still important. DES was the first block cipher, which was widely deployed in public sector, and hence it played an important role making strong cryptography available to the public.

The design was exceptionally good for a cipher that was meant to be used only a few years. DES proved to be a very strong cipher and it took over a decade for any interesting crypt analytical attacks against it to develop. Development of differential cryptanalysis and linear cryptanalysis opened ways to better understand the design of block ciphers. Even today, where DES is no longer considered for any future practical solution, it is often used to describe new crypt analytical techniques. It is remarkable that even today there is no technique that would completely break DES in a structural way; indeed, the only real weakness known is the short key size and perhaps the small block size.

2.2.2. Advanced Encryption Standard (AES)

In response to the growing feasibility of attacks against DES, NIST launched a call for proposals for an official successor that meets 21^{st} century security needs. This successor is called Advanced Encryption Standard (AES). After five algorithms making to the second round, Rijndael was selected to be the final standard. Let's take a look at the cryptographic peculiarity of AES.

AES has a 128-bit block size, and it supports 128, 192, 256 bit keys. The rather large key sizes are probably required to give means for construction of efficient hash functions. AES follows the tradition of square ciphers. NIST gave as its reasons for selecting Rijndael that it performs very well in hardware and software across a wide range of environments in all possible modes. It has excellent key setup time and has low memory requirements, in addition its operations are easy to defend against power and timing attacks.

Many commonly used ciphers are block ciphers. Block ciphers transform a fixed-size block of data into another fixed-size block using a function selected by the key. If the key, input block and output block all have n bits, a block cipher basically defines a one-to-one mapping from n-bit integers to permutations of n-bit integers. If the same block is encrypted twice with the same key, the resulting cipher text blocks are also the same. This mode of encryption is called electronic codebook or ECB. This information could be useful for an attacker. To cause identical plaintext blocks being encrypted to different cipher text blocks, two standard modes are commonly used:

AES is an iterated block cipher with a variable block length and a variable key length. They block length and the key length can be chosen independently as 128, 192 or 256 bits. The block where the operations are applied is called a state. A state is organized as an array of 8 bits with 4 rows. The number of columns is equal to the block length divided by 32. The cipher key is also organized in the same form. Shown below is the state with 192 bits.

A0,0	A0,1	A0,2	A0,3	A0,4	A0,5	A0,6	A0,7
A1,0	A1,1	A1,2	A1,3	A1,4	A1,5	A1,6	A1,7
A2,0	A2,1	A2,2	A2,3	A2,4	A2,5	A2,6	A2,7
A3,0	A3,1	A3,2	A3,3	A3,4	A3,5	A3,6	A3,7

Like DES, AES consists a number of equivalent rounds. The number depends on the lengths of the block and key and is provided in the following table.

	128	**192**	**256**
128	10	12	14
192	12	12	14
256	14	14	14

The transformation in each round is composed of four different transformations in the following order

Byte Substitution (ByteSub) – This is a non-linear byte substitution realized by an 8-8 S-box. This is executed on each byte in the state.

Row Shifting (ShiftRow) – The four rows of the state are cyclically shifted over different offsets.

Column Mixing (MixColumn) - Every column of a state is transformed using a matrix multiplication in this step.

Round key addition (AddRoundKey) – A round key is applied to the state by a bitwise XOR.

However in the final round the step of column mixing is not performed. The round keys are derived from the key by a key schedule. For each round, we need a round key of the same size as the size of the state. This is achieved by recursive expansion of the key to the size of (number of rounds) * (size of state). From this expanded key the round keys are taken sequentially.

The Encryption Algorithm

The encryption algorithm consists of

- Key expansion
- An initial round key addition
- Several rounds of ByteSub, ShiftRow, MixColumn and AddRoundKey
- And a final round of ByteSub, ShiftRow and AddRoundKey

The Decryption Algorithm

The inverse of a round is given by AddRoundKey, Inverted MixColumn, inverted ShiftRow and inverted ByteSub. The inverse of a final round is got by AddRoundKey, inverted ShiftRow and inverted ByteSub. After which a final AddRoundKey is done.

Due to algebraic properties of the steps of a round, the algorithm can be rearranged like the following.

- Inverse key expansion
- An initial round of key addition
- Several rounds of inverted ByteSub, ShiftRow, MixColumn and AddRoundKey.
- A final round of inverted ByteSub, ShiftRow and AddRoundKey.

2.3 Asymmetric Key Cryptography

Asymmetric Key Cryptography is also called Public-key cryptography. It uses a secret key that must be kept from unauthorized users and a public key that can be made public to everyone. Both the public key and the private key are mathematically linked; data encrypted with the public key can be decrypted only by the private key and the data signed with the private key can only be verified with the public key.

The public key can be published to anyone. Both the keys are unique to the communication session. Public-key cryptographic algorithms use a fixed buffer size as opposed to variable length buffer size used by the private-key variants. Public-key algorithms cannot be used to chain data together into streams like their counterparts can.

The theory behind asymmetric key algorithms was first published by Whitfield Diffie and Martin Hellman in 1975 [5], and since then, several implementations have been created. One widely-used asymmetric key algorithm is RSA. It uses exponentiation modulo a product of two large primes to encrypt and decrypt. The public key exponent differs from the private key exponent, and determining one exponent from the other is believed to be fundamentally hard without knowing the primes. Public-key algorithms can be used, depending on the protocol, for either confidentiality or sender authentication. For instance, a user can encrypt a message with their private key and send it. That it can be decrypted using the corresponding public key provides assurance that that user sent it and none else.

Examples of well regarded asymmetric key algorithms include:

- Diffe-Hellman
- RSA
- ElGamal
- Elliptic Curve cryptography

And Examples of protocols using these algorithms includes:

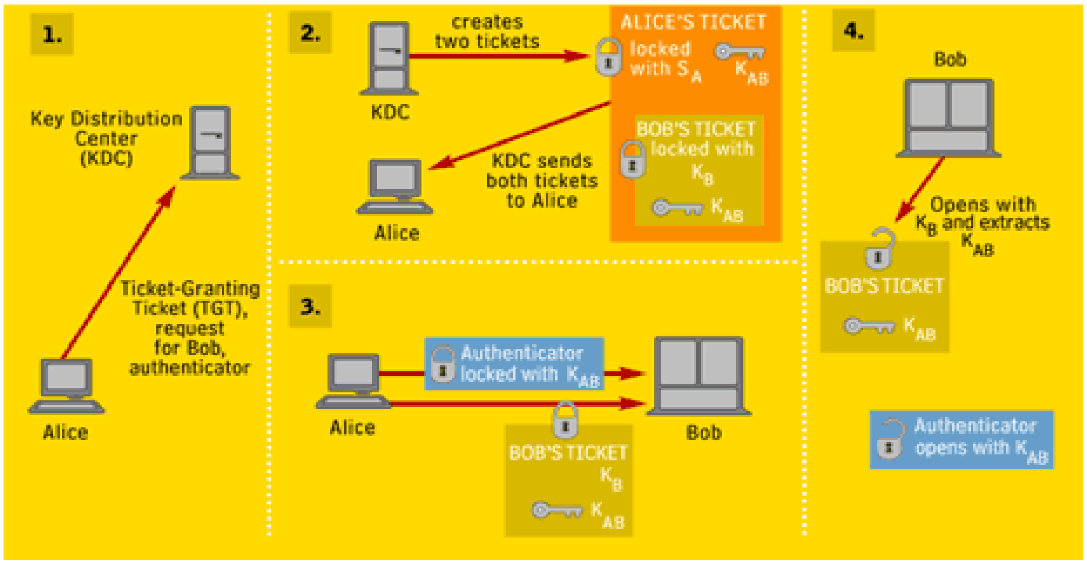
- SSH
- SSL now implemented as an IETF standard – TLS
- Digital Signature Standard (DSS)

2.4 Kerberos – A Primer

Kerberos provides a means of verifying the identities of principals, (e.g., a workstation user or a network server) on an open (unprotected) network. Its purpose is to allow users and services to *authenticate* themselves to each other. That is, it allows them to demonstrate their identity to each other, unequivocally. This is accomplished without relying on authentication by the host operating system, without basing trust on host addresses, without requiring physical security of all the hosts on the network, and under the assumption that packets traveling along the network can be read, modified, and inserted at will. Kerberos relies heavily on authentication technique involving conventional cryptography, i.e., shared secret key. The basic concept is simple: if a secret is known by only two people, then either person can verify the identity of the other by confirming that the other person knows the secret.

For example, let's suppose that Alice often sends messages to Bob and that Bob needs to be sure that a message from Alice is really has come from Alice before he acts on its information. They decide to solve their problem by selecting a password, and they agree not to share this secret with anyone else. If Alice's messages can somehow demonstrate that the sender knows the password, Bob will know that the sender is Alice.

The only question left for Alice and Bob is to resolve how Alice will show that she knows the password. The password could be simply included somewhere in the message, perhaps in a signature block at the end – Alice, Secret. This could be simple and efficient and will work only if Alice and Bob can be sure that no one else is reading their information. Unfortunately, that's not the case. The messages pass over a network used by other people of whom, some have a hobby of scanning traffic in hope that one day they might spot a password. So it is out of question for Alice to prove that she knows the secret by simply saying it. To keep the password secret, she must show that she knows it without revealing it. Kerberos solves this problem with secret key cryptography. Rather than sharing a password, communication partners share a cryptographic key, and they use knowledge of this key to verify one another's identity. One party proves the knowledge of the key by encrypting a piece of information and other by decrypting it.


1.
Key Distribution Center (KDC)
Ticket-Granting Ticket (TGT), request for Bob, authenticator
Alice
2.
KDC
creates two tickets
ALICE'S TICKET locked with S_A K_{AB}
BOB'S TICKET locked with K_B K_{AB}
KDC sends both tickets to Alice
Alice
3.
Alice
Authenticator locked with K_{AB}
BOB'S TICKET K_B K_{AB}
Bob
4.
Bob
Opens with K_B and extracts K_{AB}
BOB'S TICKET K_{AB}
Authenticator opens with K_{AB}

The Authenticator. When Alice wants to authenticate to Bob, she must first contact the Key Distribution Center (KDC). 1) Alice sends her Ticket-Granting Ticket (TGT) and a request to authenticate to Bob to the KDC. 2) The KDC creates two tickets and a new session key (K_{AB}). The KDC nests Bob's ticket inside Alice's ticket. Alice's ticket is encrypted with Alice's secret key (S_A), and Bob's ticket with his secret key (K_B). Both tickets include the new key (K_{AB}). The KDC returns both tickets to Alice. 3) Alice decrypts her own ticket, revealing the new session key (K_{AB}). Alice then encrypts an authenticator with K_{AB} and sends Bob's ticket and the encrypted authenticator to Bob. 4) Bob decrypts his ticket with K_B, revealing the session key (K_{AB}), which he can use to reveal the authenticator from Alice.

Figure: 1 Kerberos Pictorial Representation

The actual authentication process in the protocol proceeds as follows: A client sends a request to the authentication server (AS) requesting "credentials" for a given server. The AS responds with these credentials, encrypted in the client's key. The credentials consist of 1) a "ticket" for the server and 2) a temporary encryption key (often called a "session key"). The client transmits the ticket (which contains the client's identity and a copy of the session key, all encrypted in the server's key) to the server. The session key (now shared by the client and server) is used to authenticate the client, and may optionally be used to authenticate the server. It may also be used to encrypt further communication between the two parties or to exchange a separate sub-session key to be used to encrypt further communication.

The implementation consists of one or more authentication servers running on physically secure hosts. The authentication servers maintain a database of principals (i.e., users and servers) and their secret keys. Code libraries provide encryption and implement the Kerberos protocol. In order to add authentication to its transactions, a typical network application adds one or two calls to the Kerberos library, which results in the transmission of the necessary messages to achieve authentication.

The Kerberos protocol consists of several sub-protocols (or exchanges). There are two methods by which a client can ask a Kerberos server for credentials. In the first approach, the client sends a cleartext request for a ticket for the desired server to the AS. The reply is sent encrypted in the client's secret key. Usually this request is for a ticket-granting ticket (TGT) which can later be used with the ticket-granting server (TGS). In the second method, the client sends a request to the TGS. The client sends the TGT to the TGS in the same manner as if it were contacting any other application server which requires Kerberos credentials. The reply is encrypted in the session key from the TGT.

Once obtained, credentials may be used to verify the identity of the principals in a transaction, to ensure the integrity of messages exchanged between them, or to preserve privacy of the messages. The application is free to choose whatever protection may be necessary. To verify the identities of the principals in a transaction, the client transmits the ticket to the server. Since the ticket is sent "in the clear" (parts of it are encrypted, but this encryption doesn't thwart replay) and might be intercepted and reused by an attacker, additional information is sent to prove that the message was originated by the principal to whom the ticket was issued. This information (called the authenticator) is encrypted in the session key, and includes a timestamp. The timestamp proves that the message was recently generated and is not a replay. Encrypting the authenticator in the session key proves that it was generated by a party possessing the session key. Since no one except the requesting principal and the server know the session key (it is never sent over the network in the clear) this guarantees the identity of the client.

The integrity of the messages exchanged between principals can also be guaranteed using the session key (passed in the ticket and contained in the credentials). This approach provides detection of both replay attacks and message stream modification attacks. It is accomplished by generating and transmitting a collision-proof checksum (elsewhere called a hash or digest function) of the client's message, keyed with the session key. Privacy and integrity of the messages exchanged between principals can be secured by encrypting the data to be passed using the session key passed in the ticket, and contained in the credentials. The authentication exchanges mentioned above require read-only access to the Kerberos database. Sometimes, however, the entries in the database must be modified, such as when adding new principals or changing a principal's key. This is done using a protocol between a client and a third Kerberos server, the Kerberos Administration Server (KADM). The administration protocol is not described in this document. There is also a protocol for maintaining multiple copies of the Kerberos database, but this can be considered an implementation detail and may vary to support different database technologies. Our interest is in exploring the performance of the protocol when the cryptic algorithm is changed to Advanced Encryption Standards.

Chapter 3

Proposed Protocol Variation and Methodology

3.1 Overview

The importance of performance of security protocols and encryption algorithms has been recognized for long by developers. Researchers have applied either complexity analysis to evaluate algorithm performance or measurement techniques to analyze system performance in most cases. [13-16] Protocol performance has become an increasingly important topic as security algorithms are more commonly used in high-workload computing and networking environments.

Early research involved in determining if encryption calculations would put a damper on rapidly increasing data rates. Zorkadis [17] has identified the communication performance impact of five basic security services: authentication, access control, confidentiality, integrity and non-repudiation. Zorkadis started exploration of impacts by constructing a three-node tandem queuing model for secure communications. In his experiment, he considered the network to be the bottleneck server and modeled it as an M/D/1 [10] queue. Zorkadis recommended several optimization strategies and analyzed the impact of pre-processing feedback blocks in the DES cipher text stream.

Since Kerberos was the standard network authentication protocol in the Open Software Foundation's Distributed Computing Environment (DCE) [18, 22], and also in the latest version of Microsoft's Network family of Operating Systems [Windows 2000 and Windows XP], this protocol has been analyzed in these contexts. DCE [19] and Windows services have been benchmarked and analyzed. In addition performance characteristics have been loosely measured in some of its applications [20], with good results. In a more recent analytical study El-Hadidi et al. [21] used a multiple, independent queue model to compare Kerberos performance to Diffie-Hellman and their proposed hybrid protocol, “HAH”. Their hybrid used secret key cryptography to distribute the Diffie-Hellman components. Their analysis concluded that Kerberos provided the best performance, Diffie-Hellman the least and their hybrid HAH in between the two.

The performance of public and secret key encryption algorithms has been studied and documented widely. Hardware implementations of secret key operations can be up to 1000 times faster than public key operations and their software counterparts can be 100 times faster [4].

3.2 Proposed Variation - AES Encryption in Kerberos

Very recently, the Kerberos Work Group has been working on including AES as the standard encryption in the protocol. This would support 128-bit block encryption, and key sizes of 128 and 256 bits. The NSA has approved the 128-bit AES for use up to SECRET level and 192-bit AES for use up to TOP SECRET level.

With large number of people involved in using Kerberos in day-to-day encryption and decryption needs, the algorithm used in this authentication protocol has been different so far. The following is a simple list of ‘modern’ Kerberos implementations and their usage of algorithms.

MIT & Heimdal

- 3DES with HMAC/SHA1 digest
- RC4 with HMAC

Microsoft

- RC4 with HMAC

Cybersafe

- 3DES with MD5 digest

With the large number of vendors using 3DES cipher, its clear that 3DES is now the most used algorithm in Kerberos. People have moved away from DES after its vulnerabilities have been exposed. Today, with RC4 support many of the implementations can work well with Microsoft Software. However the long term desire is for all implementations to use AES as the default preference instead of RC4 or even 3DES.

The Kerberos protocols described in this document are designed to use stream encryption ciphers, which can be simulated using commonly available block encryption ciphers, such as the Data Encryption Standard [6], in conjunction with block chaining and checksum methods [7]. Encryption is used to prove the identities of the network entities participating in message exchanges. All principals registered in that realm to store a secret key in confidence trust the Key Distribution Center for each realm. Proof of knowledge of this secret key is used to verify the authenticity of a principal.

The KDC uses the principal's secret key (in the AS exchange) or a shared session key (in the TGS exchange) to encrypt responses to ticket requests; the ability to obtain the secret key or session key implies the knowledge of the appropriate keys and the identity of the KDC. The ability of a principal to decrypt the KDC response and present a Ticket and a properly formed Authenticator (generated with the session key from the KDC response) to a service verifies the identity of the principal; likewise the ability of the service to extract the session key from the Ticket and prove its knowledge thereof in a response verifies the identity of the service.

The Kerberos protocols generally assume that the encryption used is secure from cryptanalysis; however, in some cases, the order of fields in the encrypted portions of messages are arranged to minimize the effects of poorly chosen keys. It is still important to choose good keys. If keys are derived from user-typed passwords, those passwords need to be well chosen to make brute force attacks more difficult. Poorly chosen keys still make easy targets for intruders. Kerberos uses Triple-DES in its recent implementations as we have seen above. Variations of Kerberos now include Public Key Infrastructure & RSA standards with Message Digest 5 checksum methods.

In this dissertation, we implement a skeleton version of Kerberos with AES cipher and compare the performance of the version with the DES and the 3DES versions of the protocol. We analyze the performance of user authentication within and across networks based on these ciphers.

The resource consumption pattern for authentication protocols cycles from the client to a network to a server and back again. In Kerberos, the cycle is from the client to a KDC and back to a client and then to a server and back to the client. This pattern is a candidate for analytical modeling with closed queuing networks. Examples of applying close queuing networks to the performance analysis of communications network to model the performance analysis of communication protocols are available. For instance, Bjorkman and Gunningberg [11] used a closed queuing network to model the performance of TCP multiprocessing architectures. Menasce and Almeida [12] provide practical guidance in development of closed and open queuing network models of Internet-based processing and communications systems – similar in architecture to the authentication protocol analyzed in this dissertation. We can discuss the methodology by which we can analyze the performance of authentication protocol in the following paragraphs.

3.3 Closed Queuing Networks

In a simple closed queuing network, (shown in the figure below), customers branch from one server to the next and consume resources at each stop. Each station in this network has queuing of customers and service demand as a result. The number of customers in the system determines the workload level. The calculation of queuing network state probabilities can be used to produce performance metrics for each queuing station and the system as a whole such as the average number of customers at a queue, the average delay time and the customer throughput.

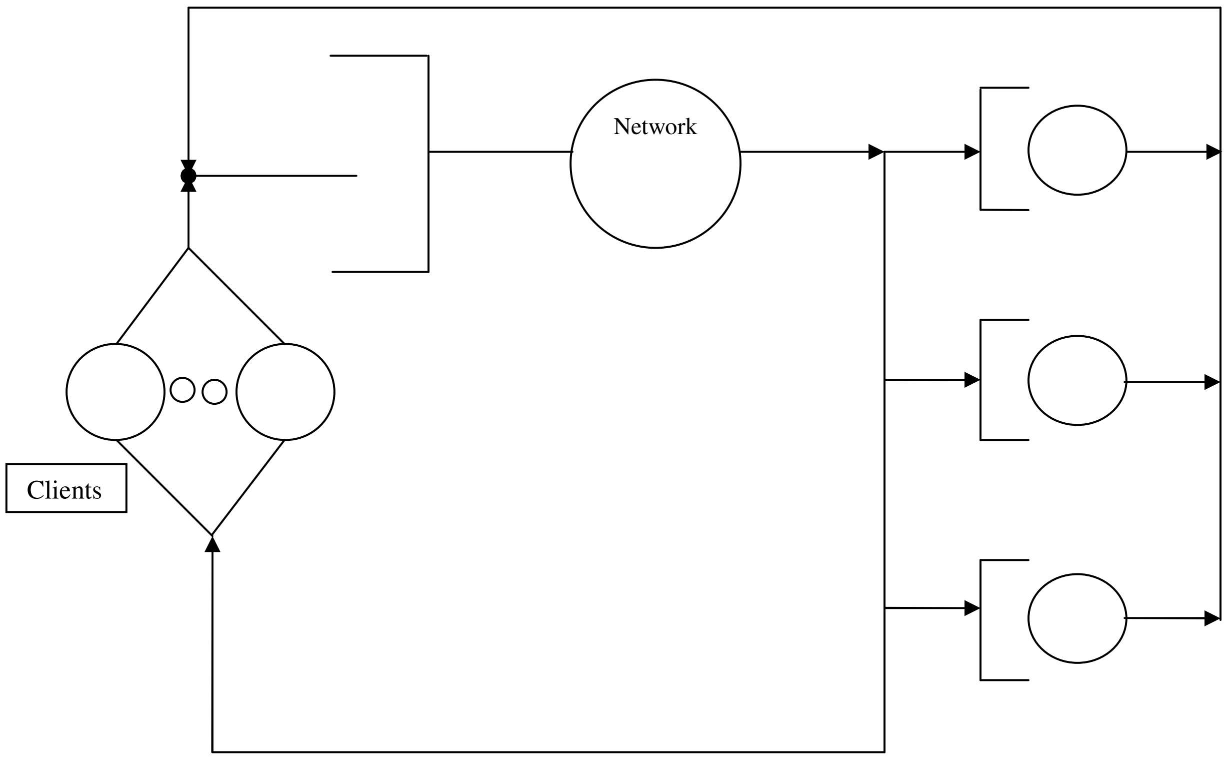


Figure 2: Closed Queuing Network Model

Mathematical derivation of the closed queuing network state probabilities can be obtained easily by enumerating all states and solving the set of equations representing balanced probabilistic flow into and out of each state. This is called global balance solution [10]. When the state space is extremely large, this global balance equation is impractical to solve.

If the closed queuing network meets a certain criteria, it has the "product form" property [14]. In a product form network, the probability that the system is in a given state N=(n1, n2, …, nk) representing the number of customers at all K servers is proportional to the product of the marginal probabilities Pi(ni) that server I has ni customers. The product form network has an efficient solution, even for large numbers of servers and customers [35,36]. The criteria required to ensure product form have been widely studied. The definitive required assumptions are called the BCMP conditions after the authors Baskett, Chandy, Muntz and Palacios [14]. BCMP requires that:

- The scheduling disciplines at each service station are first-come-first-serve (FCFS), processor sharing (PS), last-come-first-served-preemptive-resume (LCFS-PR), or infinite server (IS).
- The service times at a FCFS server must be exponentially distributed and same for all customers. This restriction does not apply to the other queuing disciplines.
- The service time at a FCFS server is either independent of queue length or depends on the length of the queue only. This restriction is relaxed for the other queuing disciplines, but there are still restrictions.

A typical queuing analyst may desire to model situations in which different customers waiting at a queuing station have a different mean service times, or different branching characteristics upon departure from a station. This situation to be modeled requires the introduction of classes of customers and the ability of customers to switch from one class to another. Bruell and Balbo [12] have generalized the Mean Value Analysis (MVA) algorithm to allow for these features.

3.4 Methodology

The methodology supports the analysis of the performance characteristics of authentication protocols over a wide range of operational conditions. At a high level, the methodology follows standard practices for conducting a performance modeling analysis: construct the model, validate the model, vary modeling parameters, and analyze the results. However, it is important to accommodate the unique characteristics of security protocols in implementing the details of each step.

The methodology follows three high-level steps:

1. Construct a closed queuing network model that reflects authentication protocol performance.
2. Validate the model.
3. Conduct “what if analysis”

In order to give the methodology the flexibility to model a wide range of operational parameters in the protocol, the queuing network solution method was chosen and it should be able to accommodate large number of users and queuing servers, support the mix of service times characterized by the multi-class formulation and there by provide a computationally efficient projection of performance metrics. The MVA method can meet the criteria for our purpose. For modeling clients or networks, a fixed delay queuing server was thought to be appropriate. We assumed that the client is not a shared resource and will not queue multiple customer requests. We coded the

AMVA algorithm for multiple class/ class-switching using the algorithm used by Harbitter and Menasce in their performance analysis.

Our proposed variant of using AES for the encryption and decryption not only will give us a stronger encryption, but also one that cannot be compromised and is faster in encryption and decryption. Variable key length and Variable block length option provided to us by AES can be used to change the encryption and decryption on the fly based on the networks authenticated and hence get stronger and faster encryptions.

3.5 Mean Value Analysis

The Mean Value Analysis (MVA) algorithm [16] provides a stable, exact, iterative solution for closed queuing networks. It is based on three relations that hold for product form networks: the arrival theorem, the forced flow law and Little's formula. The arrival theorem states that a new customer arriving at a queuing station will join a line that has the same average length as the total average length of the same queuing station in a network with one less customer. The forced flow relates the system throughput to the throughput for individual queuing stations. Little's formula states that the average number of customers in line at a queuing station is equal to the arrival rate to the station time the average wait for service. These theorems can be combined to derive the four equations behind the iterative MVA method:

Wi(n) = si + siLi(n-1) (3.1)

Y0(n) = n/(Sigma viWi(n)) (3.2)

$$Yi(n) = Yo(n)Vi \quad (3.3)$$

$$Li(n) = Yi(n)Wi(n) \quad (3.4)$$

Where:

Wi(n) = the mean delay at server i when there are n customers in the system

Si = the mean service time at server i

Li(n) = the mean number of customers at server I when there are n customers in the system.

Yo(n) = the average system throughput when there are n customers in the system

n = the number of customers in the system

K = the number of devices in the system

Vi = the average number of visits made to server I in one cycle through the system

Yi(n) = the average throughput at server I when there are n customers in the system

Equation 3.1 states the arrival theorem. Equation 3.2 is Little's formula applied to the entire system. Equation 3.3 is the forced flow law. Finally, equation 3.4 is Little's formula for each queuing station. Note that equation 3.1 requires information from a state with one less customer than the current state – the MVA algorithm is iterative. The performance measures for the full system are computed by iterating equations 3.1

through 3.4 for all K devices in the systems, starting from n = 0 up to the total number of customers.

For large numbers of customers, completing all of the MVA iterations may be compute intensive, especially when there are different classes of customers. Bard and Schweitzer [17] have developed an approximate MVA algorithm (AVMA) that eliminates the requirements to iterate on the number of customers. AMVA replaces equations 3.1 (i.e., the equation that requires the result from the previous iteration on the number in system) with an approximation:

$$Wi = si + \{(n-1)/n\} siLi \qquad (3.5)$$

In equation 3.5, Wi, si and Li are independent of the iteration on the number in system – they represent their final MVA values (i.e., after the MVA iterations are complete). Effectively, AVMA approximates the number at a server when there are n-1 customers in the system {(n-1)/n]Li. This estimates the requirement to iterate on customer count. Instead, the AMVA equations are solved successively until the difference in consecutive estimates of Li is less that some threshold value, e. Agarawal [9] has proven that AMVA, for a single class network, always converges.

3.5 Program Description Language (PDL) – AMVA Algorithm

```
AMVA() // AMVA Algorithm
Begin:

//INPUTS
Input starting number of customers in each class
Input the visit ratios v(I,r) for servers I and classes r
Input server queuing types ServerType(i) – Delay for PS
Input service time for each server and class s(I,r)
Input assignment of classes to ECs

//INITIALIZATIONS
//Calculate visit ratios v*(I,q) for the ECs
Do for all servers I, ECs q, and classes r in EC q;
        V*(I,q) = (SUMover r in EC q(v(I, r)))/(SUMover r in EC q (v(0,r)));
EndDo;

// Calculate alpha(i,r)'s
Do for all servers i, Ecs q and classes r in EC q;
        Alpha(I,r) = v(I,r)/SUMover s(v(I,s));
EndDo;

//Calculate the service time s*(I,q) for the ECs
Do for all servers I, ECs q and classes r in EC q;
        S*(I,q) = SUMover r in EC q(s(I,r));
EndDo;

//initialize n*estimate (I,q)'s
n*estimate(i,q) = the sum of the customer count for each server i and class member of EC q;
```

```
//Main Calculation
epsilonCheck = FALSE;
Do while (NOT(epsilonCheck));
          SOLVE(n*, n*estimate, w*, s*, x*);
If(the difference between all n*(i) and n*estimate(i) is less than epsilon)
          epsilonCheck = TRUE;
EndDo;

//Calculate Class Level statistics from EC Statistics
Do for all servers I, ECs q, and classes r;

 if(serverType(i) = "Delay" then w(I,r) = s(I,r);

else w(i,r) = s(I,r) * (1 + SUMover q(n*(i,q)) + (n*(i,q') -1));

x(i,r) = alpha(i,r) *x*(1,q')*v*(i,q); //where class r is a member of EC q'

EndDo;

EndMain;


Procedure SOLVE(n*, n*estimate, w*, s*, x*)
          //update queue length estimates to latest value

          Do for all servers I and ECs q;
          n*(i,q) = n*estimate(i,q);
          EndDo;

          //Calculate wait time
          Do for all servers i and Ecs q;
                    If ServerType(i) = "delay" Then w*(i,q)  = s(i,q);
                    Else w*(i,q) = s*(i,q) * (1+(n*(i,q)-1)/n*(i,q));
                   X*(0,q) = Nq(q)/SUMover i(w*(i,q)*v*(i,q));
         EndDo;

          Do for all devices I and ECs q;
                    n*estimate(i,q) = x*(1,q)*w*(i,q)*v*(i,q);
          EndDo;
EndProc;
```

Chapter 4

Setup and Simulation

To investigate the performance aforementioned we created classes (C++) for the various encryption algorithms in question and formed a skeleton implementation of the protocol. We then used a closed network queuing model to simulate the authentication process by changing the various parameters to get close to accurate results.

4.1 The Queuing Model

The KDCs, application server, communication networks and client workstations are finite resources that process workload as Kerberos authentication transactions execute. We built a closed queuing network model to represent each resource used by the protocols. The setup anticipates that the local KDC may be connected to the client by a local area network (LAN), and the remote KDC and application server may be connected via a wide area network (WAN). The validated model will use a LAN to connect all KDCs and servers, matching the test bed configuration.

Kerberos though available freely to configure ourselves on our servers was not used in entirety for this performance modeling. We built a 'skeleton' implementation of the protocol. We developed the skeleton in C++ and used the Karns' DES/3DES library

[8] for symmetric key encryption. We also used the Brian Gladman's [24] C++ version of AES library for our evaluation.

We implemented the KDCs one on Microsoft Windows 2000 and one on Gentoo Linux server. The application server was also implemented on Microsoft Windows 2000 and on Mac OS X. We had clients on Windows XP and Solaris for X86. We did not use any operating-system specific extensions in our programs. The client process steps through the standard Kerberos authentication message sequence to request service from an application server in a local or remote realm. The primary purpose of the skeleton software on the client is to issue requests periodically, quickly confirm the validity of the response and timestamp the transaction to report response time statistics. The client side processing has been simplified to focus on the encryption process and decryption process on the KDCs and the application servers. The client process will loop through many transactions for the purpose of reporting average response time statistics.

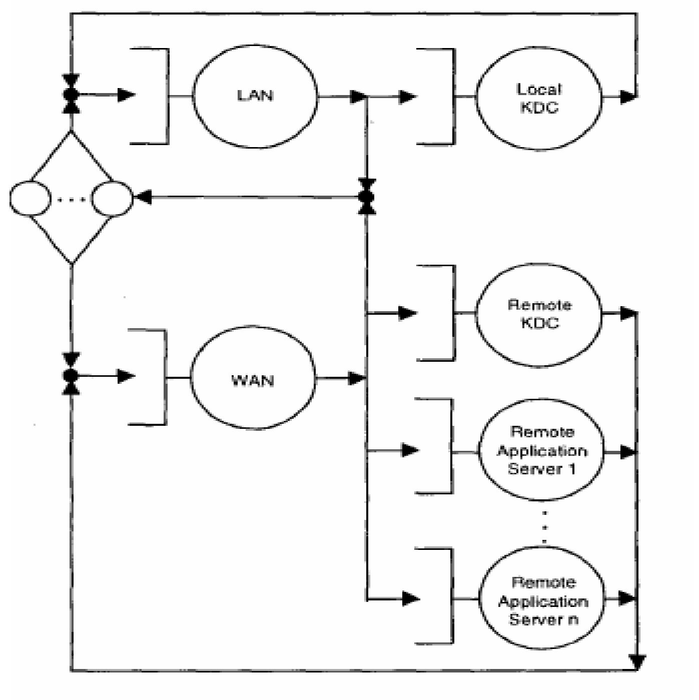


Figure 3: Topology of a two –realm Kerberos closed queuing model

The above figure shows as a sample of how the two-realm close queuing model was setup in our simulation.

4.2 Setup

We configured the client, KDCs and application server implementations to perform all operations with 192 Bit keys in 3DES and also with 192 bit key and block size. The total number of authentications was recorded on each side for a transaction. The transaction phase of the protocol was recorded in steps as shown below

Client requests TGT

Local KDC responds

Client requests remote TGT

Local KDC transacts with remote KDC

Client requests service ticket from remote KDC

Remote KDC responds

Client authenticates to remote application server.

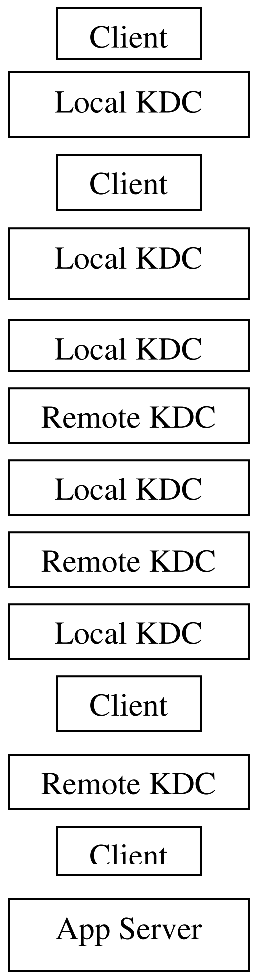


Figure 4: Transaction Phase for a cross realm authentication

The above figure has the steps in which the Kerberos authentication across two realms would happen. The following figure is the setup in our test bed illustrating this cross realm simulation that we did.

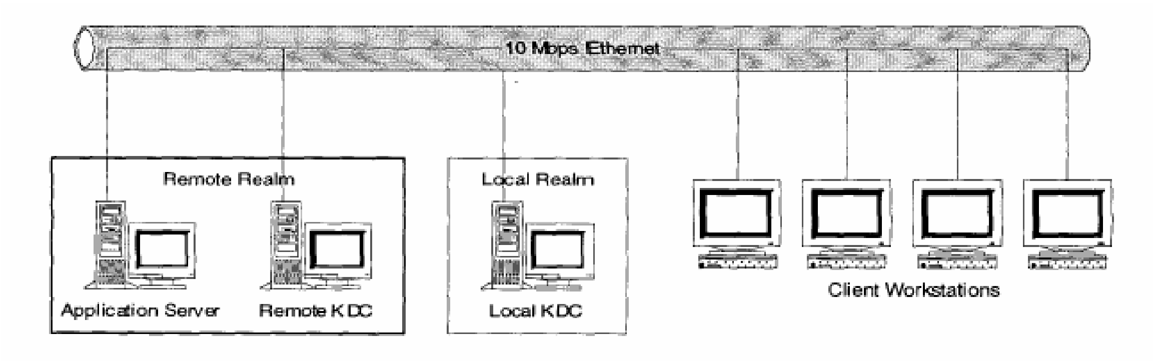


Figure 5: Test Bed setup for the cross realm authentication

The clients in this setup where Microsoft Windows XP machines and the local KDC was a windows 2000 server and the remote KDC was a Linux server running KDC on it and the application server was also a Windows 2000 Server machine.

4.3 Simulation

A single process was set to run on KDC to accept client requests in UDP datagram and use one of the encrypt algorithms on each run to authenticate the requests locally and also cross-authenticate requests with remote KDC. Two processes run on the remote KDC: one waiting for standard Kerberos requests arriving as UDP datagram, and the other opens up a TCP listening socket and waits for the remote requests from KDCs. All KDC and application server processes are multi-threaded; when they receive a message, they dispatch a thread to process and respond to the request. In the step of transaction where the client authenticates to the application server, it uses a ticket received from the local or remote KDC appropriately.

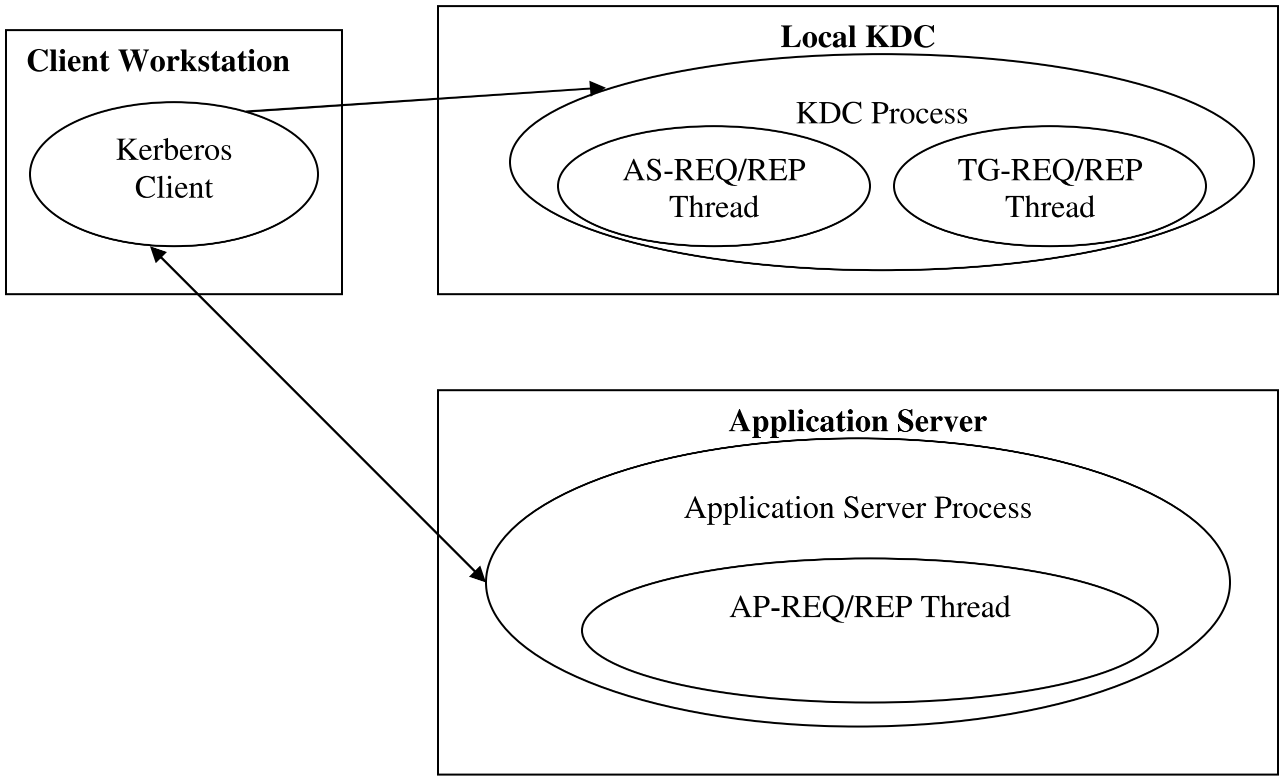
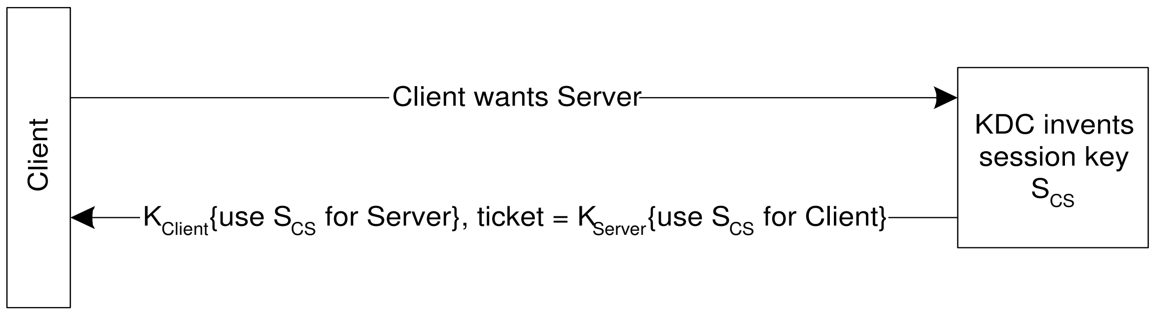


Figure 6: Authentication Steps in the Simulation

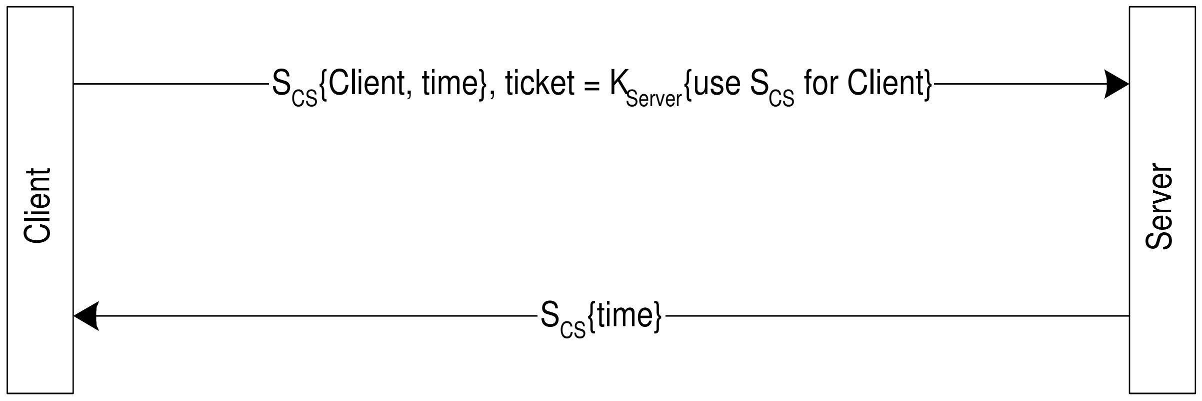
The above figure illustrates the authentication steps setup in the simulation for the test. The client first requests the local KDC with AS-REQ/REP and then the KDC authenticates and responds with the ticket grant TG-REQ. The client takes the ticket to the application server with an AP-REQ and the server decrypts and provides service to the client. Following are the sequences of authentication in the setup.



Client
Client wants Server
KDC invents session key S_{CS}
K_{Client}\{use S_{CS} for Server\}, ticket = K_{Server}\{use S_{CS} for Client\}

Figure 7: Sequence showing Client request to KDC

The above sequence diagram shows the step at which the initial request is made by the client to the KDC server. The first encryption happens at this point. The KDC however at this point only provides the ticket granting service.



Client
Server
S_{CS}\{Client, time\}, ticket = K_{Server}\{use S_{CS} for Client\}
S_{CS}\{time\}

Figure 8 Client using the Session Ticket to request service from Server

The above figure illustrates the next step at which the client then deciphers the ticket it got from the KDC and then uses the session ticket provided by the KDC to request for service from the application server. Now there is one more deciphering at the Servers end to authenticate the ticket provided by the client. And if the mutual authentication of Kerberos is desired by the implementers, then there can be another authentication from the client side to authenticate the server so that he can know he is talking to the server he thinks to be.

Clients were programmed to make several thousand requests to the local application servers and several thousand requests to the remote application servers. The response time for each authentication was noted down and also the total throughput time for each request on the whole was noted down. The number of clients was increased for each run and also the depth of the remote access to the KDC was also changed for each runs. Data was collected from the client programs, KDC and the application server programs. The analysis of the simulation and the results are discussed in the following chapter.

Chapter 5

Analysis and Results

We used the validated model to investigate the performance of the protocol based on the encryption algorithms. We increased the number of application servers and we found that as we increased the number, the number of 'visits' made to the corresponding servers in each transaction also increases. This means the number of encryption/decryption process also increases with each increased step. On the other hand when we tested the access to local servers, the number of encryptions is always constant due to a single step authentication process. So the response time of the protocol authentication when number of clients was increased was calculated with this protocol variant.

5.1 Charts and Analysis

All the collected data from the simulation was transported the Microsoft Excel Charts. We analyze the performance of the proposed variant of Kerberos versus the 3DES version. The following chart provides the comparative performance with respect to the total throughput time of the authentication itself within a local realm.

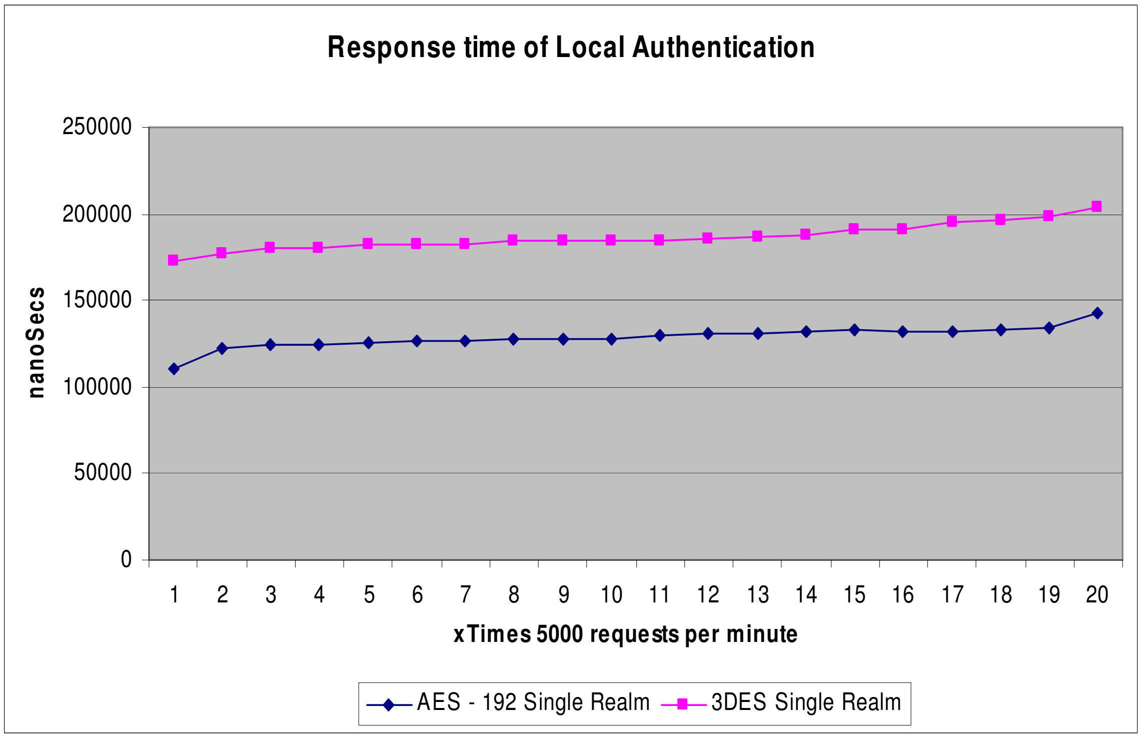


Figure 9: Comparison of Response time of Local Server transaction

From the above figure it is clearly evident that the AES takes 50% less time to authenticate users when compared to Triple DES on a 192 bit key length AES. The following chart on the other hand shows the total response time for cross authentication of servers in Kerberos based on our implementation. Again the comparison is between Triple DES and a 192 bit key length AES.

Response time is the elapsed time between the start of the request made by the client to the KDC and the time the application server authenticates the client's ticket to grant service in our model. This would include the entire authentication process including cross realm authentication with other KDC and application servers when needed.

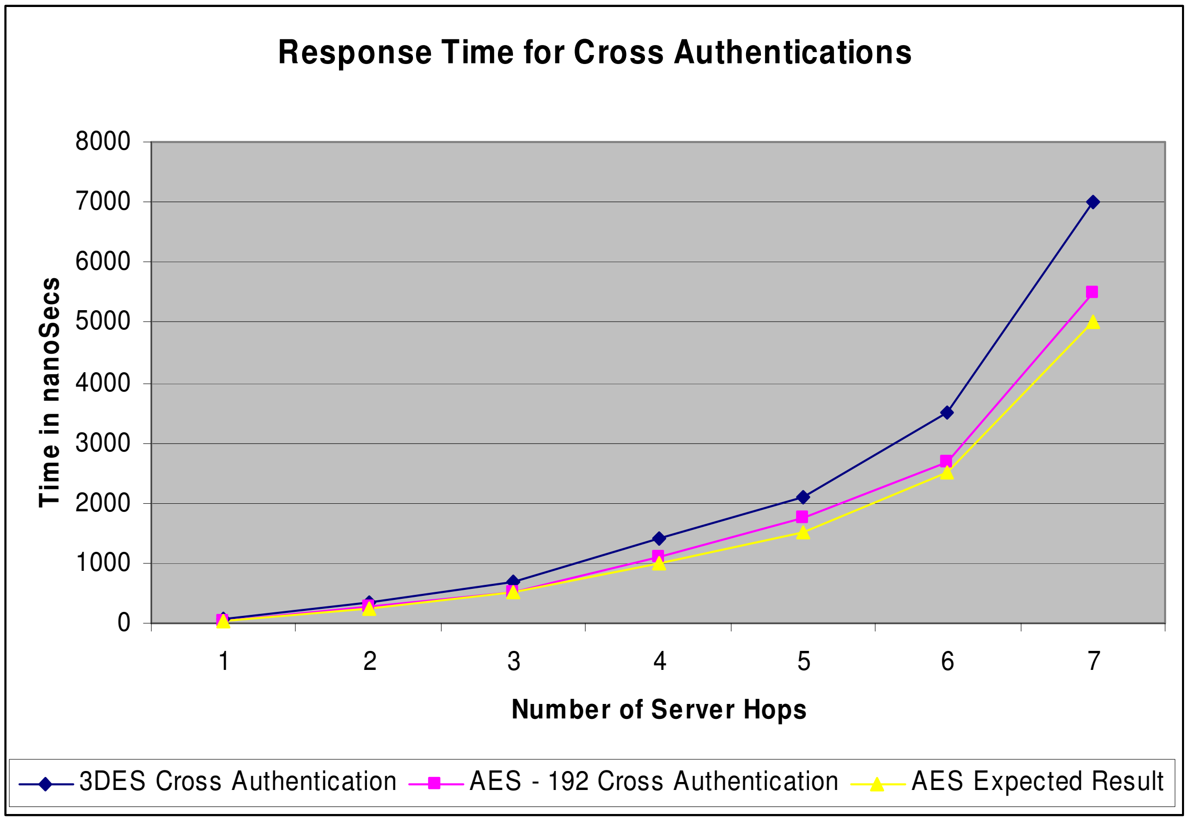


Figure 10: Comparison of Response Time for Cross Server Authentication

Once again the above figure shows that AES performed close to expectations. However we believe that on a complete implementation of the Kerberos with more tuning based on the hardware used, the results could be even better. We based our expectations on the Approximate Mean Value Analysis we used in our queuing theory and the Crypto ++ benchmarks from Dai.W, [2] for the encryption algorithms that we use.

To explore the relationship between the protocol and the encryption algorithm used, we reran the models with collecting the encryption and decryption times. We did not change the configurations of our servers or KDC. Our model accounts only for the authentication process and we did not consider the network workload in our sampling. Since a significant portion of the authentication process consists of encryption and decryption, we analyzed the performance of the same. The data we got is shown in the chart below.

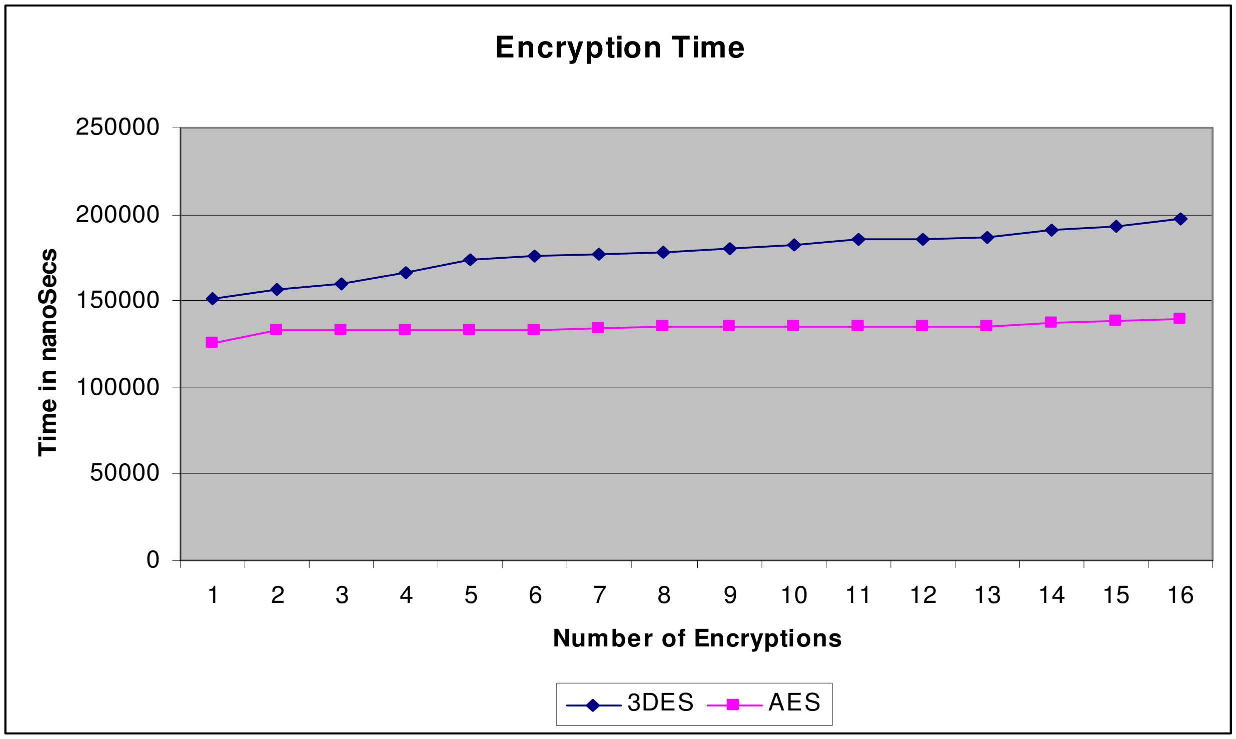


Figure 11: Comparison of Encryption time of two algorithms with Kerberos

As we can clearly see above, this chart also proves that AES proves to be faster in encryptions than its competitor 3DES. Since the designers of 3DES want the algorithm to be backward compatible with DES, they had to do a decryption in between two encryptions for 3DES encryption and reversed the process for decryption. In 3DES, so during decryption, the process is to decrypt, encrypt and then decrypt. This makes decryptions also to be slower on the algorithm when compared to AES. We tried to limit the bandwidth of our internal network setup to 10Mbps and still got similar results.

We have demonstrated, through the use of validated analytical queuing model, the quantitative performance differences between the two variants of Kerberos. Our analysis shows that

1. Given KDCs and application servers of approximately equal capacity, AES variant outperforms the 3DES variant for authenticating to application server in the same realm. We compared the response time that the Kerberos setup took to completely authenticate the client and provide application service to him.
2. Our observation on authentication response time for cross-realm servers also yielded that AES variant was faster than the 3DES variant. We maintained the capacity of application servers to be equal even when in cross-realm mode.
3. Comparing only the encryption time, with controlled environment of all the variables, we still found that AES variant was faster in authentication than 3DES variant. Decryption times were also compared and found to be faster in AES variant.

At the end of our analysis, we see that AES remains the preferred cryptic algorithm for Kerberos authentication protocol as it provides better performance combined with better cryptography.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Kerberos has proven to be a very good network authentication protocol and is being used by many vendors in their products in the recent years. With the introduction of AES, Kerberos should be even better than what it is now and provide stronger and secure network authentication. We have demonstrated, through the use of validated analytical models, the performance between the two variants of the protocol: 3DES and AES. Our analysis shows that:

- Given KDCs and application servers of approximately equal capacity, the AES outperforms the simpler protocol 3DES for authenticating to more than one application server in local or remote realm.
- Speeding up the application server relative to KDCs with improving the block and key length of AES, provides better throughput for high range authentication and also greatly improves the response times.
- We were also able to tune the KDCs to provide multiple authentications for remote realms with faster response time using the minimal key and block length in AES.

We believe that the flexibility provided by AES to the encryption options in the protocol enables us to tune the system based on the individual requirements of the network authentication and hence provide a strong and safe protocol.

6.2 Future Work

What we have done here is a low level analysis on the encryption options the Kerberos authentication protocol can have with the availability of AES. Our findings can be used to guide a high-level network authentication protocol that can combine the power and flexibility of AES to improve performance on the whole. Use of such a high-level protocol with AES would provide flexibility to the designers of security networks in small and large scale organizations to tune the protocol. As there is already a request to design the next version of Kerberos with AES included, this analysis would be of great help to compare the performance of the two variants.

References

[1] Apostolopoulos, G., V.Peris and D.Saha. *Transport Layer Security: How Much does it really cost?* In IEEE INFOCOM. 1999.

[2] Dai, W., *Crypto++ 3.1 Benchmarks. 1999*

[3] Kaufman, C., R.Perlman, and M Speciner, *Network Security, Private Communication in a Public World. 1995,* Englewood Cliffs, New Jersey; PTR Prentice Hall.

[4] Schneier, B., *Applied Cryptography.* Second Ed. 1996, New York: John Wiley & Sons, Inc. 758

[5] Diffie, W. and M.E.Hellman, *New Directions in Cryptography.* IEEE Transactions on Information Theory, 1976. IT-22(6): p, 644-654

[6] A.H. Harbitter and D.A. Menasce *Performance of Public-Key Enabled Kerberos Authentication in Large Networks 2001* IEEE. Symposium on Security and Privacy. Oakland, CA P.170-183

[7] Metricom Inc. *Ricochet Security Whitepaper,* April 2001

[8] Karn, P., *Index of /cryptocd/source/ciphers/des/c/karn, 2000:* http://www.cryptocd.org/cryptocd/source/cyphers/des/c/karn/.

[9] Agarawal, S.C., *Metamodeling: A Study of Approximations in Queuing Models.* MIT Series in Computer Systems, ed. H.Shwetman. 1985, Cambridge, MA: MIT Press. 262.

[10] Gross, D and C.M.Harris, *Fundamentals of Queuing Theory.* Third Ed. 1998,

New York; John Wiley & Sons, Inc. 439

[11] Baskett ,F., et al., *Open, Closed and Mixed Networks of Queues with Different classes for customers.* J.ACM, 1975. 22(2).

[12] Bruell, S.C, and G.Balbo, *Computational Algorithms for Closed Queuing Networks.* The Computer Science Library, ed. P.J.Denning. 1980, New York; Elsevier North Holland, Inc

[13] Booselaers, A., *Fast Implementations on the Pentium, in* http://www.esat.kuleuven.ac.be/~bosslae/fast.html. 1999

[14] Schneier, B., et al., *Performance Characteristics of the AES Submission.* 1999

[15] Baasham, L.E., *Efficiency Testing of ANSI C Implementations of Round 1 Candidate Algorithms for the Advanced Encryption Standard. 1999,* Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology.

[16] Blaze, M., *High Bandwidth Encryption with Low-Bandwidth Smartcards.* Lecuture notes in Computer Science, 1996. Number 1039

[17] Zorkadis, V. *Security versus Performance Requirements in Data Communications Systems.* 1994

[18] OpenGroup, DEC 1.1: *Authentication and Security services.* 1997: http://www.opengroup.org/publications/catalog/c311.htm.

[19] Martinka, J., et al. *A Performance Study of DCE 1.0.1 Cell Directory Service: Implications for Application and Tool Programmers.* In Lecture notes in Computer Science. 1993.

[20] Stallings, W., *Kerberos Keeps the Enterprise Secure, in Data Communications.* 1994. p.103-111.

[21] El-Hadidi, M.T., N.H.Hegazi, and H.K.Aslan. *Performance Evaluation of a New Hybrid Encryption Protocol for Authentication and Key Distribution*. In IEEE International Symposium on Computers and Communications. 1999. Red Sea, Egypt.

[22] OpenGroup, *Distributed Computing Environment.* 1998 http://www.camb.opengroup.org/dce/.

[23] Saha, S., M.Jamtgaard, and J.Villasenor, *Bringing the Wireless Internet to Mobile Devices,* in IEEE Computer. 2001. p.54-58

[24] Brian Gladman's AES library in C++ http://fp.gladman.plus.com/cryptography_technology/rijndael/

Bibliography

[B1] Apostolopoulos, G., et al., *Securing Electronic Commerce: Reducing the SSL Overhead.* IEEE Network, 2000: p.8-16

[B2] RSA., *Understanding Public Key Infrastructure (PKI).* 1999, RSA Security Inc.

[B3] Needham, R.M. and M.D. Schroeder, *Using encryption for authentication in large networks of computers.* Communications of ACM, 1978, 21 (December 1978): p. 993-999.

[B4] Ashley, P. and B.Broom, *A Survey of secure Multi-Domain Distributed Architectures.* 1997, Queensland University of Technology, Faculty of Information Technology.

[B5] Sirbu, M.A. and J.C.-I. Chuang. *Distributed Authentication in Kerberos Using Public Key Cryptography. in Symposium on Network and Distributed system security.* 1997. san Diego, California: IEEE Computer Society Press.

[B6] Personal Communications Industry Association, *Market Demand Forecast for Terrestrial Third Generation* (IMT – 2000) Service for the PCIA. 1998

[B7] Fox, A and S.D.Gribble, *Security on the Move: Indirect Authentication using Kerberos.* In MOBICOM 96. 1996. Rye, New York.

[B8] Zenel, B., *A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment.* Wireless Networks, 1999, 5: p. 391- 409

[B9] Swift, M., et al., *Initial and Pass Through Authentication Using Kerberos v5 and the GSS-API (IAKERB), 2001, IETF*

[B10] WAP, *Wireless Application Protocol Wireless Transport Layer Security Specification, 2000,* Wireless Application Forum, Ltd, 2000

[B11] Burrows, M., M.Abadi, and R.Needham, *A Logic of Authentication,* ACM Transactions on Computer Systems, 1990, 8(1):p, 18-36

[B12] Lambert, P., *Elliptic Curve Cryptography Delivers High Performance and Security of E-Commerce.* Computer Security Journal, 1998. XIV(4); p.23-29

[B13] Intel, *What is Moore's Law, 2000 Wysiwyg://6//http://www.intel/museum/25anniv/hof/moore.htm*

[B14] Brooks, D.M et al., *Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors*, in IEEE Micro, 2000 p.26-44.

[B15] Stevens, W.R., *TCP/IP Illustrated, Volume 1*. 1999, Reading, Massachusetts, Addison-Wesley. 576

[B16] Standard Performance Evaluation Corporation, CFP2000rate Benchmarks, 2000: http://www.spec.org

[B17] Johnson, M., North American Cryptography Archives. 2000: http://cryptography.org/cgi-bin/noexport.cgi

[B18] Cristian, F,. *Exception Handling and tolerance of Software Faults,* in Software Fault Tolerance, M.R> Lyu, Editor. 1995, Wiley: Chichester. P.81-107

[B19] Menasce, D.A and V.A.F Almeida, *Scaling for E-Business: Technologies, Models, Performance and Capacity Planning. 2000: Prentice Hall.*

[B20] Bjorkman, M. and P.Gunningberg, *Performance Models of Multiprocessor Implementation of protocols.* IEEE/ACM Transactions of Networking, 1998. 6: p.262-273

[B21] Orozco-Barbosa, L., L.Serrano, and E. Quiroz. *Performance Evaluation of the IS-41 Security Mechanisms in a PCS Supporting Intelligent Network Services.* In Intelligent Network Workshop. 1998.

[B22] Adcock, J.M., et al *Trading Off Strength and Performance in Network Authentication: Experience with the ACSA project* in DARPA Information Survivability Conference and Exposition. 2000. Hilton Head, South Carolina, IEEE.

Request For Comments

[R1] Kohl, J., *The Kerberos Network Authentication Service (V5),* C. Neuman, Editor. 1993: http://www.ietf.org/rfc/rfc1510.txt?number=1510.

[R2] Tung, B., et al., *Public Key Cryptography for Initial Authentication in Kerberos.* 2001: http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt

[R3] Tung, B., et al., *Public Key Cryptography for Cross-Realm Authentication in Kerberos. 1998:* http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-cross-03.txt

[R4] Medvinsky, A., et al., *Public Key Utilizing Tickets for Application Servers (PKTAPP).* 1997: http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-tapp-03.txt

[R5] Neuman Cliff *The Kerberos Network Authentication Service (V5)* 2003: http://www.ietf.org/internet-drafts/draft-ietf-krb-wg-kerberos-clarifications-02.txt

Appendix A

Glossary

Kerberos – The network authentication protocol developed by MIT as a part of Athena project.

PKI – Public Key Infrastructure is the combination of digital certificates, public-key cryptography and certificate authorities into total network security architecture.

KDC – Key Distribution Center, it acts as an admin to authorize and authenticate request from clients within the local realm and also authorize requests from remote realms.

Authenticators - A simple protocol that uses secret key authentication when someone outside a communications door wants to get in. To gain entry the person presents a piece of information encrypted in the secret key.

Long-term Key – The cryptographic key stored by the KDC for each principal and that is exchanged between the security principal and the KDC is known as Long-term key. In most implementations it's derived from the user's logon password.

Session key - When a client wants to access a server for information, it sends request to KDC and the KDC distributes a unique short-term session key for the two parties when they authenticate each other. The server's copy of the session key is encrypted in the server's long-term key. The client's copy of the session key is encrypted in the client's long-term key.

TGT - Ticket-Granting Ticket is a special session ticket that KDC returns to the client. The TGT is encrypted in the KDC's long-term key. And the TGT also contains the client's session key that it can use to communicate to the KDC

AES – Advanced Encryption standard is a block cipher that was designed by Joan Daemen and Vincent Rijmen as a candidate algorithm. The cipher is discussed in detail in this document.

Vita

Anush Krishnamurthy

Candidate for the Degree of

Master of Science

Thesis: PERFORMANCE IMPACT OF ENCRYPTION ALGORITHMS ON KERBEROS NETWORK AUTHENTICATION PROTOCOL

Major Field: Computer Science

Biographical:

Personal Data: Born in Madras (Chennai), Tamilnadu, India, On July 20, 1979, the son of Mr.J.Krishnamurthy and Mrs.P.V.Rajalakshmi.

Education: Graduated from C.T.T.E.Higher Secondary School, Chennai, Tamilnadu, India in April 1996: Received Bachelor of Engineering in Computer Science and Engineering from the University of Madras, Chennai, Tamilnadu, India in April 2000.
Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December 2004.

Experience: Employed as Software Developer with Bloomberg L.P, a financial media and software company in New York October 2002 – November 2005.
Employed as Software Engineer with Intel Corporation, a processor and platform making company in Chandler, Arizona since November 2005.

Name: Anush Krishnamurthy Date of Degree: May, 2006

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: PERFORMANCE IMPACT OF ENCRYPTION ALGORITHMS ON KERBEROS NETWORK AUTHENTICATION PROTOCOL

Pages in Study: 53 Candidate for the Degree of Masters of Science

Major Field: Computer Science

Scope and method of Study: Kerberos is a network authentication protocol that was developed by MIT and now widely used by various organizations. Microsoft Corporation uses Kerberos as their network authentication protocol in their latest network operating systems family [Windows 2000 and Windows XP]. Key performance issues of authentication protocols have always been the compromises by cryptanalysis over the ciphers used. While speed of the authentication is most important in authentication protocols, a strong cipher also is most desired by the users. This thesis analyses the performance of the authentication protocol in question with various encryption algorithms. Apart from the stronger encryption and decryption achieved in the proposed variant, the speed of the authentication also greatly improves with this new encryption standard.

Findings and Conclusions: When the key size for the encryption is increased and the authentication is within the known network, the performance is very much promising with more transactions throughput. Standard key sizes and increased block sizes in the proposed encryption algorithm improves the cross network authentication throughput largely.

ADVISOR'S APPROVAL:______________________________

Dr.Johnson Thomas