In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.

While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.

Security Key Management for IP-based VPN

By Yongqing Han

**A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of**

**Master of Science
in
Information and System Science**

Department of Systems and Computer Engineering

**Carleton University
Ottawa, Ontario
April 25, 2003**

**© copyright
2003, Yongqing Han**

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-88770-7

Our file Notre référence

ISBN: 0-612-88770-7

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Canada

**The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis**

Security Key Management for IP-based VPN

**Submitted by Yongqing Han
in partial fulfillment of the requirements for
the degree of Master of Science**

Thesis Supervisor

Chair, Department of Systems and Computer Engineering

Carleton University

April 25, 2003

Abstract

Virtual Private Networks (VPNs) provide an inexpensive and scalable security solution to transfer sensitive data through an unsecured network by creating a “tunnel” from sender to receiver. One of the most popular protocols for creating VPNs is the IPSec protocol suite, where secure key negotiation and exchange must be finished first before any encryption of data can take place.

This research examined the latest VPN technologies focusing on one of the factors which have an effect on VPN scalability, namely the security key management. A new aggregation key exchange approach compatible to current technologies is proposed for improving the key exchange performance in large VPN systems. The new proposed approach represents a trade-off between performance and security.

A simulation model based on the Network Simulator (*NS*) was developed for the new proposed approach. Simulation experiments for various scenarios were conducted and their results were compared to the traditional key exchange scheme.

Acknowledgement

First of all, I thank God for giving me the wisdom and strength to accomplish this research.

Second, I would like to take this opportunity to give my deep appreciation and respect to my thesis supervisor, Professor Dorina Petriu. Thanks for her guidance and encouragement throughout my M.S. program, especially during the research for this thesis. Without her effort and support, this research would not have been accomplished.

Finally, I would like to thank my parents, Uncle Joseph, my sister Yongli and Zhe Zhou who always stood behind me and help me to move forward step by step firmly.

Acronyms

AH	Authentication Header
CBR	Constant Bit Rate
CE	Customer Edge
DES	Data Encryption Standard
ESP	Encapsulating Security Payload
GRE	Generic Routing Encapsulation
GT-ITM	Georgia Tech Inter-network Topology Model
ICV	Integrity Check Value
IKE	Internet Key Exchange
IETF	Internet Engineering Task Force
IPSec	Internet Protocol Security
ISAKMP	Internet Security Association and Key Management Protocol
ISP	Internet Service Provider
L2F	Layer 2 Forwarding
L2TP	Layer 2 Tunneling Protocol
MACs	Message Authentication Codes
NIC	Network Interface Card
NS	Network Simulator
PE	Provider Edge
PPTP	Point-to-point Tunneling Protocol
RTP	Real Time Protocol
SA	Security Association
SGB	Stanford Graph Base
SPD	Security Policy Database
SPI	Security Parameter Index
SRM	Scalable Reliable Multicast
VPN	Virtual Private Network

Table of Contents

ABSTRACT	I
ACKNOWLEDGEMENT	II
ACRONYMS	III
TABLE OF CONTENTS.....	IV
LIST OF FIGURES	VIII
CHAPTER 1 INTRODUCTION	1
1.1 BACKGROUND.....	1
1.2 THESIS OBJECTIVE	3
1.3 THESIS CONTRIBUTIONS	4
1.4 THESIS OUTLINE	5
CHAPTER 2 BACKGROUND: VPN AND IPSEC	7
2.1 VIRTUAL PRIVATE NETWORK – VPN	7
2.1.1 Motivation of VPN	7
2.1.2 VPN Technologies	8
2.1.3 VPN Applications	12

2.2	OVERVIEW OF IPSEC PROTOCOL.....	14
2.2.1	Security Problems over Internet.....	15
2.2.2	IPSec Traffic Security Solutions – AH and ESP Protocols	19
2.2.2.1	Authentication Header (AH)	21
2.2.2.2	Encapsulating Security Payload (ESP)	23
2.2.2.3	Protection Mode	25
2.2.3	Internet Key Exchange – IKE.....	28
2.2.3.1	Security Association.....	28
2.2.3.2	Key Exchange	30
2.3	VPN SECURITY.....	37
2.3.1	VPN Security models.....	37
2.3.2	Scalability and Security Issues.....	41
2.3.3	Traditional Key Distribution and Public Key	45
CHAPTER 3 NETWORK SIMULATOR		47
3.1	NETWORK SIMULATOR – NS	47
3.1.1	NS Background.....	47

3.1.2	NS Architecture	48
3.1.3	Network Simulation Basics in <i>NS</i>	52
3.2	CREATION OF NEW PROTOCOL FOR <i>NS</i>	58
3.3	GT-IMT TOPOLOGY GENERATOR	61
CHAPTER 4 AGGREGATION APPROACH FOR KEY EXCHANGE.....		64
4.1	AGGREGATION APPROACH FOR KEY EXCHANGE	64
4.2	THE SIMULATION MODEL	69
4.2.1	Development of IKE Agent	70
4.2.2	Internetworking Topology Modeling.....	71
4.2.3	Simulation Model Design	76
4.2.4	Traffic Design	78
4.3	SIMULATION EXPERIMENTS	80
4.3.1	Key Parameters for the Simulation Experiments.....	80
4.3.2	Simulation Cases.....	81
CHAPTER 5 SIMULATION ANALYSIS.....		83
5.1	USING CONFIDENCE INTERVALS FOR DATA ANALYSIS	83
5.2	ANALYSIS OF SIMULATION RESULTS	89

TABLE OF CONTENTS

5.2.1	Simulation Output.....	89
5.2.2	Relationship between Network Size and Key Exchange Time	90
5.2.3	Relationship between Node Connectivity and Key Exchange Time	92
5.2.4	Effect of Key Exchange Frequency	94
CHAPTER 6 CONCLUSIONS AND FUTURE WORK.....		98
6.1	SUMMARY OF RESEARCH WORK.....	98
6.2	CONCLUSION OF PRESENT RESEARCH.....	100
6.3	FUTURE WORK	101
REFERENCES		102

List of Figures

Figure 2.1	Intercepted and Opened Up by Unauthorized Parties	16
Figure 2.2	Identity spoofing.....	18
Figure 2.3	IPSec Network Layer Encryption.....	20
Figure 2.4	the Standard Format of AH Header.....	22
Figure 2.5	Standard Format of ESP Header.....	24
Figure 2.6	AH Transport Mode	26
Figure 2.7	ESP Transport Mode	26
Figure 2.8	AH Tunnel Mode.....	27
Figure 2.9	ESP Tunnel Mode.....	27
Figure 2.10	IKE Main Mode.....	32
Figure 2.11	IKE Aggressive Mode	33
Figure 2.12	IKE Quick Mode	34
Figure 2.13	Format of IKE Header	35
Figure 2.14	Format of General Header for All Types of Payload	35

Figure 2.15 IKE Payloads Chained to From a Message	36
Figure 2.16 Typical VPN Architecture	38
Figure 2.17 Traditional CE-to-CE VPN Architecture	39
Figure 2.18 Network Based Layer 3 VPN Model.....	41
Figure 2.19 Comparison of Scalability for CE-based VPN Model vs IP Network-based VPN Model	43
Figure 3.1 Dual Objects Built in NS Appear in both C++ and OTcl Classes.....	49
Figure 3.2 Structure of Unicast Node	53
Figure 3.3 Composite Construction of a Duplex Link.....	54
Figure 3.4 Structure of Unicast Routing between Two Nodes with Duplex Link.....	55
Figure 3.5 Bi-direction Traffic Flow between Two Agents Attached on Nodes.....	57
Figure 3.6 Partial C++ Class Hierarchy in NS.....	59
Figure 3.7 API of Transit-Stub Model Graph Created by GT_IMT	63
Figure 4.1 Aggregated Key Exchange by Establishing Public Tunnel in Backbone Network when Using Public Key	67
Figure 4.2 Transit-Stub Domain Random Graphic Structure	72
Figure 4.3 Topo_N50A with 52 Nodes and Node Connectivity Degree 2.679.....	73

Figure 4.4	Topo_N50B with 52 Nodes and Node Connectivity Degree 3.692	74
Figure 4.5	Topo_N100A with 114 Nodes and Node Connectivity Degree 2.682	74
Figure 4.6	Topo_N100B with 114 Nodes and Node Connectivity Degree 3.686	75
Figure 4.7	Topo_N150A with 150 Nodes and Node Connectivity Degree 2.684	75
Figure 4.8	Topo_N150B with 150 Nodes and Node Connectivity Degree 3.69	76
Figure 5.1	Relationship between Key Exchange Time and Network Size with Lower Connectivity Degree	90
Figure 5.2	Relationship between Key Exchange Time and Network Size with Higher Connectivity Degree	91
Figure 5.3	Key Exchange Time Versus Network Size for different key exchange approaches and connectivity degrees	92
Figure 5.4	Effect of Node Connectivity Degree on the Performance of the Traditional Key Exchange Scheme	93
Figure 5.5	Effect of Node Connectivity Degree on the Performance of the Aggregation Key Exchange Scheme	93
Figure 5.6	Key Exchange Time with Various Key Lifetimes for 52-node Topology	95
Figure 5.7	Key Exchange Time with Various Key Lifetimes for 108-node Topology ...	96
Figure 5.8	Key Exchange Time with Various Key Lifetimes for 150-node Topology ..	96

List of Tables

Table 3.1	Classes of <i>NS</i> Protocols.	48
Table 4.1	Summary of Simulation Topology Characteristics.....	73
Table 4.2	Summary of Simulation Cases.....	81
Table 5.1	Key Exchange Time for Different Network Size with Node Connectivity 2.68.....	89
Table 5.2	Key Exchange Time for Different Network Size with Node Connectivity 3.69.....	89
Table 5.3	Key Exchange Time with Different Lifetime Values.....	95

Chapter 1 Introduction

1.1 Background

The Internet is quickly becoming the primary vehicle for communication in the business community. The transfer of important and sensitive data over the Internet has become popular. Unfortunately, using the Internet to transfer important data is not as safe as many users assume. This is because anyone with access to the Internet and the required technology can intercept data and use the information to cause damage. Obviously, there is a need to provide a safe and reliable way to communicate sensitive data over the Internet.

In the past, customers who required a secure network were forced to lease public telecommunication lines or use frame relay circuits. These solutions may have been safe, but they were also expensive, inflexible and had difficulties in supporting remote users. An alternative solution that is quickly gaining acceptance is the Virtual Private Network – VPN.

The strength of a VPN is its ability to transmit information securely and reliably over unsecured public telecommunication infrastructure. A VPN is designed to transmit encrypted information through point-to-point tunnels over a packet switched unsecured network. At the receiving far-end, the information is decrypted, filtered and checked for

the identity. VPN provides network customers with an inexpensive, safe and scalable security solution.

A VPN may be implemented in several ways: LAN to LAN – extranet, remote users to LAN, and within an Intranet. There are four common protocols used for creating VPNs over the Internet:

- Point-to-point Tunneling Protocol (PPTP) [23]
- Layer 2 Forwarding (L2F) [45]
- Layer 2 Tunneling Protocol (L2TP) [44]
- IP Security Protocol (IPSec) [22, 29, 30, 31]

Among these, IPSecs has become the most popular protocol in use. By adding security to the IP protocol, IPSec combines safe communication with TCP/IP connectivity. In addition, IPSec takes advantage of the latest technologies for encryption and authentication. IPSec supports the Internet Key Exchange (IKE) protocol, which authenticates the users and handles key exchange. It is also compatible with X.509 certificate management [2].

Negotiation and Key Exchange make IPSec different from other protocols. Encryption keys are parameters needed to perform encryption/decryption algorithms. In order to use any encryption in a network environment, the communicating nodes must first exchange the keys. IKE gives IPSec a unique, powerful and well-conceived way to deal with this issue. Most importantly, IPSec makes it practical to create secure VPNs over networks ranging from tens of nodes to tens of thousands of nodes easily, conveniently and transparently with these negotiation and key exchange schemes.

1.2 Thesis Objective

Very often, when deploying VPN services, a Service Provider (SP) will be confronted with a trade-off between scalability and security. Many serious concerns regarding the scalability of current VPN technology have been addressed. An important factor that affects the VPN performance is the amount of control information an IPSec-switch has to process and exchange. This becomes a serious issue when an SP builds a large network infrastructure.

The main goal of this research is to study the performance aspects of the negotiation and key exchange approaches within the current technologies, and to propose improvements where possible. The following research activities were performed in the thesis:

- Analyze the scalability and security of VPN systems with current technologies. Identify potential scalability problems with traditional negotiation and key exchange scheme.
- Propose a new negotiation and key exchange approach – *aggregation key exchange approach*, which is more scalable and manageable for large network infrastructure. It can be applied over multiple service providers' shared network. This solution represents a scalability-security trade-off, which is also discussed in the thesis.

We conduct a simulation model to evaluate this approach. Simulations are performed for different scenarios with particular experimental parameters, such as network size, network mesh density and frequency of information exchange between two

IPSec nodes. The same simulation scenario was conducted on both traditional and aggregation key exchange schemes, in order to compare them.

1.3 Thesis Contributions

The contributions of this research can be divided in two parts:

Contributions to knowledge:

- Studied and analyzed VPN scalability issues within the current IPSec technology, and identified what factors have an impact on the system performance. The potential VPN scalability problems related to security key management are addressed in the next item.
- Proposed a new negotiation and key exchange approach in order to build more scalable IP-based VPN systems. This approach is compatible with the current IPSec suite protocol and X.509 public key infrastructure standard.

Practical Contribution:

- Designed and implemented an IKE Agent for the Network Simulator environment *NS*, according to the IKE protocol. The IKE Agent is implemented by using both C++ and OTcl languages required by *NS*. This agent can be shared with the community of *NS* users.

1.4 Thesis Outline

The thesis is composed of six chapters, as follows.

Chapter 1 gives the introduction of the thesis. It includes: the background, objectives, the main contributions of the research and summary of each chapter of this thesis.

Chapter 2 gives a bibliographic overview of current VPN technologies, focusing on security issues. It addresses the security threats over a public network. It describes the technology of IPSec suite protocol including: Authentication Header, Encapsulation Security Protocol and Internet Key Exchange protocol. It describes how the protocols work together to fit into the IPSec protocol architecture.

Chapter 3 describes the architecture and features of the Network Simulator. It also summarizes how to develop a new Agent object in *NS* for a new protocol. The last section of this chapter describes a tool, GT-IMT, for generating network topology models.

Chapter 4 is the core chapter of the thesis. It analyzes the scalability and security issues for different VPN system models, such as Customer Premises (CP) VPN model and IP-based VPN model. It briefly describes the technology of public key and traditional key exchange scheme. Due to a potential scalability problem of the traditional exchange scheme, a new aggregation exchange approach is proposed. The last part of this chapter describes the *NS* simulation model for the two schemes, including the agent development,

network topology selection, traffic design, and experimental parameters considered for simulations. At last it summarizes different simulation scenarios.

Chapter 5 describes first the methodology for analyzing the simulation output, including the computation of confidence intervals. The simulations of both traditional and aggregation exchange schemes are performed for different scenarios, as listed in the previous chapter. The results for the two schemes are compared, and the effect of different experimental factors on system performance is discussed.

Chapter 6: This chapter gives the summary and conclusions of this research. Future work related this research is also identified in this chapter.

Chapter 2 Background: VPN and IPSec

This section gives a high-level overview of VPN, IPSec and Key management protocol. The first part of this chapter introduces the motivation of deploying VPN, VPN technologies and VPN applications. The second part describes the IPSec protocol and the last section covers the discussion of key management.

2.1 Virtual Private Network – VPN

2.1.1 Motivation of VPN

As the Internet became more and more accessible and bandwidth capacities grew, companies began to offload their Intranets to the web and create extranets to link internal and external users. Using the Internet, companies can connect their remote branch offices, projects teams, business partners and e-customers by using local ISP (Internet Service Providers). However, as cost-effective and quickly-to-deploy as Internet is, there is one fundamental problem – security.

The fundamental motivation for VPNs is the communication *privacy* [5, 14]. To ensure security in a network environment, the following three secure aspects must be considered:

- Authentication
- Confidentiality
- Integrity

VPN solutions are designed to deal with these security factors. Using special tunneling protocols and complex encryption procedures, data integrity and privacy is achieved in point-to-point connections. Because the VPN provides such operations over public networks without changing the existing router hardware equipment, VPNs can cost significantly less to implement than privately owned or leased services. Therefore, communication economics is another basic motivation for deploying VPNs.

2.1.2 VPN Technologies

As the Internet is a shared public network of networks with open transmission protocols, IP-based VPNs must include measures for packet encapsulation, encryption, and authentication to ensure that the sensitive data reaches its destination without being tampered by unauthorized parties. Companies are using an Internet VPN to establish links to the local access points of their ISP. From here, they let the ISP ensure that the data is transmitted to the appropriate destinations via the Internet, leaving the rest of the connectivity details to the ISP's network and the Internet infrastructure. The following is a brief review of the most common technologies for implementing VPNs [3, 5, 17, 19, 27].

Firewall

A firewall prevents unauthorized users and data from getting in or out of a private network, by using rules to specify acceptable communications from given locations and/or individuals. Such rules are specific to certain protocols, as for example filtering IP packets based on the IP address, or on the user ID. Firewalls can be implemented in either software or standalone hardware. [3] However, firewalls do not protect the data from threats in the Internet network itself. Once the data gets outside the firewall, the user names, passwords, account number and other sensitive information are visible to hackers. Therefore the concept of a VPN tunnel was invented to give the ability to protect the data transmission in the public network after it leaves the protective firewalls.

Tunnel

Tunneling technology provides data integrity, data source authentication, data confidentiality, and protection against replay attacks [15, 31] Tunneling encrypts and encapsulates any private network protocols within the Internet protocol (IP). In this manner, it basically encrypts the users' original IP packet and encapsulates into another IP header. When its packets are routed through the shared public network, nobody can directly access the true information. Currently different encapsulation schemes are available such as IP-in-IP Encapsulation, Generic Routing Encapsulation (GRE) [25]. Encapsulated packets are processed by routers and bridges as any other packets.

Encryption

Encryption is a technique for scrambling and unscrambling data. At either end of a VPN tunnel, which is bi-directional, the source VPN gateway encrypts the data and sends the encrypted information through the tunnel over the public Internet. The VPN gateway at the receiving end decrypts the information. A question arises: since the most common encryption algorithms, such as Data Encryption Standard (DES), are well-known in the industry, how is the data kept secure if everyone knows the algorithm? The answer is the key.

Key

A key is the secret code used by an encryption algorithm to create a unique version of non-readable data. In VPN encryption, the algorithm may be the same, but the keys are different for various users. The security strength depends on the length of the key. For example, an 8-bit key can generate 256 combinations, and a 168-bit key which, is commonly used in VPN products in the current market, gives 2^{168} possible combinations. Even the fastest computer today needs a considerable amount of time to crack a code out of these combinations.

There are two types of keys used for encryption: symmetrical and asymmetrical keys [3, 24]. Symmetrical means that the same key is used at each end of the tunnel to encrypt and decrypt information. Since a symmetrical key is being shared by both parties and must be kept confidential, these keys are more difficult to distribute. Key exchange becomes unmanageable for the large VPN groups [8, 32].

Asymmetrical keys allow to encrypt the information with one key and to decrypt it with a different key. They are slightly more complicated but a lot easier to manage. The two asymmetrical keys used in such a scenario are referred to as public and private keys.

Authentication

Authentication is used by the recipients of data to determine whether the sender really is the one he says he is (*user/system authentication*), and whether the data was redirected or corrupted en route (*data authentication*) [3]. User/System authentication can be achieved by using encryption and decryption as follows. Consider two communication pairs A and B. When A receives a message signed from B, A picks a random number and encrypts it using a key only B should be able to decode. B then decrypts the random number and re-encrypts it using a key only A should be able to decode. When A gets its number back, it can be assured it really is B on the other end. In order to verify that the data packets received have not been altered, VPN often uses the following procedure: a hash function creates a sort of fingerprint of the original data, by calculating a unique number, called a hash value, based on variable length values of unique bit strings. The sender attaches the number to the data packet before the encryption. When the recipient receives the data and decrypts it, he can calculate his own hash independently. The output of his calculation is compared to the stored value appended by the sender. If these two hash values do not match, it is considered that the data has been corrupted or altered by a hacker [3, 15, 32, 38].

A hash function is used for modern cryptography purposes because the output of a hash function will be identical for identical input. A hash function H takes a variable-size

input x and returns a fixed-size string, which is called the hash value h (that is, $h = H(x)$). Given a hash value h , it is computationally infeasible to find some input x such that $H(x) = h$ and find any two messages x and y such that $H(x) = H(y)$. Popular hash functions in use today are: MD5 (Message Digest 5), SHA (Secure Hash Algorithm) and RIPEMD [15]. The hash values will be exchanged via IKE protocol in the first round trip messages by sender and recipient.

2.1.3 VPN Applications

This section will describe different features of a typical secure VPN usually, a VPN allows remote users to access the enterprise VPN over a shared infrastructure. VPNs are probably the most cost-effective method of establishing point-to-point connections between remote users and the enterprise customer's network. IP-based VPN meet the customers' requirements to extend intranets to remote offices, mobile users and telecommuters. Further, an enterprise VPN can use extranet links to the partners, suppliers and key customers of that enterprise. Based on the user point of view, there are three basic types of VPN [12, 13]:

- Client – initiated Access VPN
- Intranet VPN
- Extranet VPN

Client – initiated Access VPN

Client – initiated Access VPNs provide secure connections for remote access for individuals (for example, mobile users) a corporate intranet or an extranet over a shared

service provider network with the same policies as a private network. It allows for remote users to use clients to establish an encrypted IP tunnel across the Internet service provider's (ISP) shared network to the enterprise customer's network. The main advantage of client-initiated access VPNs over NAS-initiated access VPNs is that they use IPSec tunnel mode to secure the connection between the client and the ISP over the public shared network.

Intranet VPN

An Intranet VPN is used to connect a corporate headquarter with remote offices or branch offices over a shared infrastructure by using dedicated connections. The advantages of this type of VPN are the sharing of the same policies for managing the security and QoS. . With an intranet VPN, any single employee of a company, regardless of location, can share the same security policy and connectivity to the corporate headquarter or to any branch offices as in a private network.

Extranet VPN

The role of an Extranet VPN is to link customers, business partners or trusted third parties to a corporate intranet over a shared infrastructure using dedicated connections. Actually, extranet is the extension of intranet access by providing authorized access to telecommuters and remote offices. The difference with respect to intranet is that remote users outside of the enterprise are given access to the enterprise VPN. The business advantage of an extranet is that it improves customer satisfaction, and at the

same time reduces business costs by allowing greater access to the company resources for the customers, suppliers and partners.

2.2 Overview of IPSec Protocol

As discussed in the introduction, the Internet as it is today is missing some key components, such as security, quality of service, reliability, and manageability. The Internet is subject to security threats including *loss of privacy*, *loss of data integrity*, *identity spoofing* and *denial of service*. The goal of IP Security (IPSec) Protocol is to address all of these threats in the network infrastructure, without requiring expensive host and application modifications [5, 31].

The architecture of IPSec is one of key technologies for providing security against all of these threats in the network environment. IPSec is designed to provide interoperable, high quality, cryptographically based security for IPv4 and IPv6. It offers the data authentication, protection against replay (integrity), and algorithms for encryption/decryption (confidentiality). These services are provided or implemented at the IP layer to protect the IP and upper layer protocols.

These objectives are accomplished by two traffic security protocols, Authentication Header (AH) and Encapsulating Security Payload (ESP), and by the cryptographic key management protocols [5, 14, 15, 31]. The IPSec protocol suite is basically comprised of the following three pieces:

- An authentication header (AH) for IP, by which communicating parties can verify that data was not modified during the transition and that it genuinely came from its apparent source;
- An encapsulating security payload (ESP) format for IP that encrypts data to secure it against eavesdropping during the transition;
- A protocol negotiation and key exchange protocol, Internet Key Exchange (IKE), which allows communicating parties to negotiate methods of secure communication.

An advantage of the IPSec suite protocol is that it is algorithm independent. In other words, it allows the designer to select different sets of algorithms without affecting the other parts of the implementation. However, a standard default algorithm is specified by the IETF group for Internet.

This section gives an overall description of how IPSec works and how all the components of the IPSec protocol fit together to provide security services.

2.2.1 Security Problems over Internet

Loss of Privacy

A perpetrator may observe confidential data as it traverses the Internet. This potential for danger is probably the largest inhibitor of business-to-business communications today. Without encryption, any message sent over the Internet may be read by an unauthorized party, as shown in Figure 2.1. Any unprotected data (even a key)

can be intercepted and opened up by unauthorized parties; this is referred to as "man-in-the-middle attack".

Figure 2.1 Intercepted and Opened Up by Unauthorized Parties

Loss of Data Integrity

Sniffing is another attack in Ethernet-based IP network. Ethernet LANs make up a large part of most networks. It is cheap, universal, well known and easy to extend. However, in Ethernet LANs, packets are available to every Ethernet node on the network. Conventionally, each node's Network Interface Card (NIC) only listens and responds to packets addressed to it. It is relatively easy to put sufficient Ethernet NICs to collect every packet that passes on the wire. A type of software called "sniffer" has been developed to take advantage of this feature of the Ethernet technology. Such tools can record the network traffic passing by them. Therefore, if someone wants to listen in some sensitive communication, this is a powerful tool to eavesdrop. Figure 2.2 illustrates the sniffing action by a hacker.

Identity Spoofing

Many security systems today rely on IP addresses to uniquely identify users. Unfortunately, this system is quite easy to fool and has led to numerous break-ins. The technique term for this attack is called IP spoofing.

At the IP level, information is broken into small, manageable chunks of data called packets. Each packet contains three things: the data, IP source address and IP destination address. When two nodes on an IP network are communicating, the data stream between them is broken into these packets and transmitted over the network. The route each packet takes to get to the destination is not controlled by any particular network element, forwarding node or any device in between. Most of devices in between only need to know the relative directions of the sender and the receiver, and are only responsible for pointing the packet the right way based on traffic levels and bandwidth availability. So the packet is forwarded through whatever nets and subnets are available to get to the destination. Two adjacent parts of data may not travel the same route. All the receiver needs to do is to take various segments and put them together. However, the source IP address in the IP packet header is easily changed. The spoofing can make a packet coming from one particular source appear as coming from somewhere else. An example of an intruder's spoofing attacks is shown in Figure 2.2.

Figure 2.2 Identity spoofing

Denial of Service

A "denial-of-service" attack is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service [7]. Examples include:

- attempts to "flood" a network, thereby preventing legitimate network traffic
- attempts to disrupt connections between two machines, thereby preventing access to a service
- attempts to prevent a particular individual from accessing a service
- attempts to disrupt service to a specific system or person

Not all service outages, even those that result from malicious activity, are necessarily denial-of-service attacks. Other types of attack may include a denial of service as a component, but the denial of service may be part of a larger attack.

IPSec offers three inter-cooperating protocols, *AH*, *ESP*, *IKE*, meant to defeat the threats to IP-based networks described above.

2.2.2 IPSec Traffic Security Solutions – AH and ESP Protocols

There is no simple solution for counteracting all the security threats described in the previous section. Every solution requires many elements, such as a security policy, standards that define what they are going to protect, and a complete set of procedures for implementing the policy. IPSec is a framework of open standards for ensuring secure private communications over IP networks, based on Internet Engineering Task Force (IETF) standards [31]. IPSec provides a flexible solution for deploying a network-wide security policy.

IPSec implements network layer encryption and authentication as shown in Figure 2.3. First, it provides an end-to-end security solution in the network architecture, so that the end systems and application level will not need any changes. Second, the encrypted packets look like ordinary IP packets. They can be easily routed through any IP network without any modification on the intermediate networking equipment. Only the end points need to know the encryption algorithm and key. This reduces the cost and implementation of the end-to-end security solution. In the following sections, the IPSec

major components used to provide the confidentiality, integrity and authenticity will be discussed.

Figure 2.3 IPSec Network Layer Encryption

IPSec security solution is defined by several standard protocols referred to as RFC 2401–2411 and 2415. These standards have the following roles:

- Add the proper information into an IP packet to enable authentication and security controls of the data transfer;
- Define how to encrypt the packet data;
- Negotiate the security association and Internet Key Exchange between two communicating entities.

Basically, IPSec uses two protocols to provide traffic security as mentioned above: Authentication Header [29] and Encapsulating Security Payload [30]. These two protocols may be applied alone or in combination to provide the desired security service in IPv4 or IPv6. Each protocol supports two transmission modes: transport mode and tunnel mode. In transport mode the protocols provide the protection primarily for upper layer protocols, and in tunnel mode the protocols are applied to secure entire IP packets.

Either the AH or ESP protocol can be performed only after the IKE protocol is completed, because IKE performs the security policy negotiation and generates the Security Parameter Index (SPI) to be filled in the SPI field of the AH and ESP headers (see Figure 2.3). When an IPSec node performs an AH or ESP encapsulation, it will look up the security policy where SPI is pointing to. The IPSec node will encrypt the data based on the policy negotiated by IKE. Similarly, in the output process when an IPSec node gets a packet, it will de-capsulate the AH or ESP header and decrypt the data based on the security policy indicated by the SPI value.

2.2.2.1 Authentication Header (AH)

The Authentication Header (AH), currently defined in RFC240, is an IPSec protocol used to provide data integrity and data origin authentication [29]. However AH does not provide data confidentiality, as ESP does. Therefore, AH does not encrypt any portion of the protected IP datagram. AH defines the method of protection, the placement of the header, the authentication coverage and input-output processing rules. AH can be used to protect the upper-layer protocol in transport mode, or the entire IP datagram in

tunnel mode [36]. . The following subsection gives the standard format of the AH header as illustrated in Figure 2.4, and the definition each AH field.

Next Header	Payload Len	Reserved
Security Parameters Index (SPI)		
Sequence Number Field		
Authentication Data (variable)		

Figure 2.4 the Standard Format of AH Header

- The payload length field is an 8-bit field specifying the size of the AH, in 32-bit words (groups of 4 bytes), minus two.
- The next header field indicates what the higher-level protocol following the AH is (ESP or TCP, for example).
- The reserved field is reserved for future use and is currently set to zero.
- The SPI, as in the ESP packet, identifies a set of security parameters (algorithms and keys) to use for this connection.
- The sequence number, also as in the ESP packet, is a number that increases each time a packet is sent with a given SPI, for the purposes of keeping track of the order in which the packets go, and making sure that the same set of parameters is not used for too many packets.
- The authentication data is the actual Integrity Check Value (ICV), or digital signature for the packet. It's much the same as the ICV used in the ESP

authentication field. It may include padding to bring the length of the header to an integral multiple of 32 bits.

Authentication Algorithms

The authentication algorithm employed for the ICV computation is based on either keyed Message Authentication Codes (MACs) or on symmetric encryption algorithms like DES, MD5, *etc* [33].

2.2.2.2 Encapsulating Security Payload (ESP)

ESP is designed to provide confidentiality, data origin authentication and anti-replay services to IP. Similar to AH, ESP may be applied in different modes in which it is inserted between the IP header and the upper-layer protocol, or it may be used to encapsulate an entire IP datagram [36]. ESP also defines both encryption and authentication mechanisms to provide security services to IP. It is currently defined in RFC2406 [30].

The main differences between AH and ESP are that:

- AH provides everything that ESP does except the encryption;
- AH does not need the payload padding but ESP does;
- AH has less overhead than ESP;
- Finally AH is much simpler than ESP.

The standard format of ESP header is shown in Figure 2.5 and defines all the fields in the ESP header.

Figure 2.5 Standard Format of ESP Header

- The security parameter index (SPI) is an arbitrary 32-bit number that specifies to the device receiving the packet what group of security protocols the sender is using for communication — which algorithms, which keys, and how long those keys are valid.
- The sequence number is a counter that increases each time a packet is sent to the same address using the same SPI. It indicates which packet is which and how many packets have been sent with the same group of parameters. The sequence number provides protection against replay attacks — in which an attacker copies a packet and sends it out of sequence to confuse communicating nodes.
- The payload data is the actual data being carried by the packet.
- The padding - from 0 to 255 bytes of data - allows that certain types of encryption algorithms require the data to be a multiple of a certain number of

bytes - to confuse sniffers trying to estimate how much data is being transmitted.

- The pad length field specifies how much of the payload is padding as opposed to data.
- The next header field, like a normal IP next header field, identifies the type of data carried and the protocol above.
- Authentication field, an optional field in the ESP, contains an integrity check value (ICV) - essentially a digital signature computed over the remaining part of the ESP. It varies in length depending on the authentication algorithm used. It may also be omitted entirely, if authentication services are not selected for the ESP.

Encryption Algorithms

ESP is designed for use with symmetric encryption algorithms. Since IP packets may arrive out of order, each packet must carry any data required to allow the receiver to establish the cryptographic synchronization for the decryption. This data is carried explicitly in the payload field.

2.2.2.3 Protection Mode

As mentioned above about IPSec protection mode, there are four possible combinations of modes: AH in transport mode, AH in tunnel mode, ESP in transport

mode and ESP in tunnel mode. The following sections describe how AH and ESP is used in these two modes.

Transport Mode

In transport mode, AH and ESP only provides protection for upper layer protocols (TCP/UDP above). AH and ESP Header goes after the IP header immediately and before other higher level protocol headers as shown in Figure 2.6 and 2.7.

Figure 2.6 AH Transport Mode

Figure 2.7 ESP Transport Mode

Tunnel Mode

In tunnel mode, a packet has two IP headers: inner and outer IP header. The inner IP header carries the ultimate source and destination addresses while an outer IP header may contain distinct IP addresses such as the security gateway address. Tunnel mode protects the entire inner IP packet including the entire inner IP header. The position of AH and ESP in tunnel mode is shown in Figure 2.8 and 2.9.

Figure 2.8 AH Tunnel Mode

Figure 2.9 ESP Tunnel Mode

The difference of these transports and tunnel modes is:

- The coverage of data to be protected. AH does not protect all of the fields in the IP header because they might be changed in transit;
- Level of security protection. AH does not address confidentiality;
- If both AH and ESP are present in the transport mode, AH goes before ESP header.

2.2.3 Internet Key Exchange – IKE

Prior to an IP packet being secured by AH or ESP, a security association (SA) must exist. The Internet Key Exchange (IKE) protocol is used to create negotiates SAs on behalf of IPSec and populates the SA database on the IPSec switch. This research will focus on how the performance of the key exchange impacts the scalability of the IPSec protocol. The next few sections give a high level overview of the key exchange and key management.

2.2.3.1 Security Association

To properly encapsulate and de-capsulate IPSec packets, it is necessary to have a way to associate the security services with the traffic to be protected and the remote peer with which IPSec traffic is being exchanged. This is called a Security Association (SA). It defines security services and is identified by a Security Parameter Index (SPI) which is specified in IPSec Header. An SA is a contract between two communicating entities, which is used to determine the IPSec algorithm for securing the packets and the keys. In IPSec, a SA specifies the following [5, 31]:

- The authentication algorithm used in AH and the keys to that authentication algorithm
- How to authenticate given communications – using what protocol, what encryption algorithm and key
- How often those keys change
- The lifetime of the keys

While the Security Association (SA) is a concept, the Security Parameter Index (SPI) is more concrete. More exactly, SPI is a number that uniquely identifies an SA. An SPI together with its corresponding SA, keeps track of keys and protocols. An SPI is an arbitrary 32-bit number the system picks to represent the SA whenever a security association negotiation takes place between two nodes *A* and *B*. When *A* negotiates with *B*, the recipient node *B* assigns an SPI which isn't already in use. From then on, whenever node *B* wishes to communicate with *A* using that SA, it uses the chosen SPI to refer to it. Node *A* on receipt would look at the SPI to determine which SA it needs to use. Then it will authenticate and/or encrypt the packet according to the policy of that SA. Also it verifies whether a) the data really comes from the node it claims to, b) no data has been modified, and c) no one reads the data between *A* and *B* (based on the agreed-upon keys and algorithms). The next step is to work out how to actually negotiate an SA. Since the IPSec architecture allows a user to specify his security policy, this means that a certain traffic has one level of security while another traffic may have a completely different level of security.

IPSec policy is maintained in the Security Policy Database (SPD) located on the IPSec switch. For each packet entering or leaving the IP stack, the SPD is consulted, and according to the corresponding SPD entry (which points to a certain SA), one of three possible actions is performed: discard, bypass or protect.. When an SPD entry defines protect action, SAs have to be created before any traffic may pass. This is where IKE comes into play. The purpose of IKE is to establish shared security parameters and authenticated keys or SA between two IPSec peers. IKE provides a means to:

- Agree on which protocols, algorithms, and keys to use
- Exchange material for generating these keys safely
- Manage the keys.

2.2.3.2 Key Exchange

IKE is actually a hybrid protocol. It integrates the Internet Security Association and Key Management Protocol (ISAKMP) with a subset of the Oakley key exchange scheme [22, 32, 34]. It provides a means to:

- agree on which protocols, algorithms and keys to use;
- ensure that the source node is exchanging the key with the node it really wants to talk to;
- manage those keys after both parties agreed on;
- exchange the necessary information for generating those keys safely.

Key exchange is a service closely related to SA management. When there is a need to create an SA, there is also a need for key exchange. Therefore, IKE wraps them both up together and delivers them as an integrated package.

IKE Phase and Exchange Modes

IKE functions in two phases. In phase one, two IKE peers establish a secure channel for negotiating the keys and security services. In phase two, two peers negotiate general IPSec SAs through the secure channel built in phase one. An IKE peer is also called a "security gateway", because it negotiates security services on behalf of the users.

IKE provides three modes for exchanging the key information and setting up SAs, two for IKE phase change and one for phase one exchange. Each exchange mode will be briefly discussed in the next sections.

- 1) The *main mode* accomplishes a phase one IKE exchange by establishing a secure channel. This mode will be used in the *NS* simulation model of the IKE Agent
- 2) The *aggressive mode* is another way of accomplishing a phase one exchange. It is simpler and faster than the main mode, but it does not provide the identity protection for the negotiating nodes since they must transmit their identities before having negotiated a secure channel.
- 3) The *quick mode* accomplishes a phase two exchange by negotiating an IPSec SA for general purpose communications.

Main Mode

The *main mode* provides a mechanism for establishing the first phase IKE SA, which is used to negotiate the future IPSec SAs. The objective of the first phase is to agree on authentication, confidentiality algorithms, hashes and keys to be able to communicate securely long enough to set up an IPSec communication.

The main mode is achieved by three round trip exchanges between the initiator and the recipient as illustrated in Figure 2.10. In the first exchange shown as message 1 and 2, the IKE peers agree on basic algorithms and hashes. In the second exchange, represented by message 3 and 4, the peers exchange public keys and pass a random number called “nonce” to other end which is generated by both sender and recipient. In the third exchange, message 5 and 6 shown below verify identities for each other. As seen in Figure 2.10, each message, such as Key and Nonce etc, is carried in its own payload; either a single or multiple payloads can be packed into a single IKE packet.

Figure 2.10 IKE Main Mode

Aggressive Mode

The aggressive mode provides the same services as the main mode. It establishes the original IKE SAs by two exchanges rather than three, with only one round trip exchange rather than three and total three packets forwarded rather than six.

In aggressive mode, an IKE peer proposes an SA, passes the public key, sends a nonce for the other party to sign and sends its ID payload so that the responder can verify its identity. The responder then sends back everything that's needed to complete the exchange. As illustrated in Figure 2.11, message 2 in aggressive mode is really an amalgamation of three response steps in main mode exchanges. The last message is for the initiator to confirm the exchange. Obviously, the advantage of the aggressive mode is less delay and the faster speed. However, aggressive mode also limits its negotiating power.

Figure 2.11 IKE Aggressive Mode

Quick Mode

Once two communicating parties have established an IKE SA using the aggressive or main mode, they can use the quick mode as illustrated in Figure 2.12. The quick mode has two purposes: negotiating general IPSec security services and generating fresh keying material.

Figure 2.12 IKE Quick Mode

The basic quick mode is completed by three packet exchanges, similar to the aggressive mode. The initiator sends a packet with the hash, proposal SA and nonce. The responder replies with a similar packet, but generates its own nonce and includes the initiator's nonce in hash for confirmation. The initiator sends back a confirming hash of both nonces. Finally, both ends will perform a hashing of the concatenation of the nonces, the SPI and the protocol values from the header that both agreed on. The resulting hash value will be the new password for that IPSec SA.

IKE Header Format

Figure 2.13 and 2.14 give the standard format for the IKE header and the definition for each field. IEEE defined another thirteen different payloads for the key exchange mechanism such as Key, Nonce, SA, ID etc. Each payload format is presented

in Appendix A. Figure 2.15 gives an example how payloads are chained to form a message to be exchanged.

Initiator Cookie			
Responder Cookie			
Next Payload	Version	Exchange Type	Flags
Message ID			
Message Length			

Figure 2.13 Format of IKE Header

Next Header	Reserved	Payload Length
Payload Data		

Figure 2.14 Format of General Header for All Types of Payload

- Initiator Cookie – initiates action of SA establishment, notification, deletion
- Responder Cookie – responding to SA establishment, notification, deletion
- Next Payload – indicates the type of the first payload in the message
- Version – version of protocol
- Exchange type
- Flag
- Message ID
- Message Length
- Next Header
- Reserved
- Payload Length

To form a message, all different payloads can be chained with the IKE header and the general header as shown below. The message can be chained as much as the user likes until the “next payload” field value of the last payload is zero.

Initiator Cookie			
Responder Cookie			
Key	Version	Exchange Type	Flags
Message ID			
Message Length			
Nonce	0	Key Payload Length	
Key Payload Data			
0	0	Nonce Payload Length	
Nonce Payload Data			

Figure 2.15 IKE Payloads Chained to Form a Message

Scalable Key Exchange

Altogether, IPSec’s IKE protocol offers network users a great deal more than other existing schemes. The thesis will focus on this point and will propose a new integrated key exchange mechanism to improve even more the overall performance.

In conclusion, the IPSec suite of protocols and different related algorithms provide the following advantages

- Can form SAs with a range of attributes, layering the network
- Can build a secure VPN over multiple communication domains
- Can update the keys as frequently as desired; the mechanism to do so is silent and easy to use

- Can apply IPSec in any IP network environment without any change on the user's side.
- Use IPSec for any network size and any enterprise VPN.
- Finally, IPSec is one of flexible protocols for secure communication.

2.3 VPN Security

This section will discuss VPN security issues, and scalability related to security key management of VPN.

2.3.1 VPN Security models

A VPN typically consists of a number of geographically distributed private customer sites that are attached to provider edge (PE) routers via customer edge (CE) devices (e.g. gateway). The sites communicate with each other across a shared ISP network, as shown in Figure 2.16. Depending on who has the responsibility for VPN deployment, there are two types of VPNs: *customer premises equipment-based (CE-based) VPN* and *network-based VPN* [8, 10, 39]. The security model used by the former is called CE-to-CE security model, and that used by the latter PE-to-PE security model.

Figure 2.16 Typical VPN Architecture

CE-to-CE VPN Model

In the case of *CE-based VPN*, the VPN functions are implemented at the customer premises. The customer has to install the VPN equipment or the software on their existing router or gateway. A secure channel must be established between each pair of CEs of the different “sites” of the same VPN as illustrated in Figure 2.17. It is clear that this solution provides a strong protection for inter-site traffic through the Internet.

However, the drawback of this VPN model is that it is very expensive and intensive information needs to be exchanged and maintained on both CE and PE.

Figure 2.17 Traditional CE-to-CE VPN Architecture

Network Based Layer 3 VPN

Network based IP VPN [10, 20, 39] is the solution for the service provider to offer VPN services to a large number of customers over the shared backbone in a scalable and manageable way, without changing the existing customer network. With this solution, customers will not be required to implement new VPN equipments or to buy new software that has specific VPN functions. As shown in Figure 2.18, the connection between customers and SP is the normal network link. Customers' packets can reach the provider edge router (PE) by using regular IP routing and forwarding. However, each PE needs to maintain a separate context for each different VPN, to ensure that the traffic of one VPN will not be injected into another VPN. For every VPN, virtual channels are established and maintained between each pair of PE connected to the different sites. Therefore, SPs are completely responsible for protecting the customer data. To achieve this, secure backbone tunnels must be established between each pair of PEs. The advantage of this VPN model is that it may provide better scalability, as the network implements the tunnels and may aggregate the VPN traffic. However, this comes at a price in terms of security, due to the weakness or non-security of the connection between CE and PE. This model offers a trade-off between scalability and security, and will be discussed further in the following section.

Figure 2.18 Network Based Layer 3 VPN Model

2.3.2 Scalability and Security Issues

This section discusses two basic issues related to the VPN models: scalability and security [8, 43].

Scalability

A simple way of addressing the scalability of VPN is to focus on three factors: memory, processing, and management.

- Memory – numbers of VPNs which can be supported by a provider edge router (PE), especially considering the IP routing and forwarding tables.
- Processing – the amount of state or information to be maintained by a PE for each VPN
- Management – the amount of control information for each VPN that a PE has to exchange with other PEs, and the frequency of this information exchange.

This research only focuses on a component of the last point namely the part concerned with the security key management.

For the CE-based VPN model, the traffic protection is achieved by creating IPSec tunnels between pairs of CEs. Figure 2.17 illustrates a fully meshed VPN, where an independent IPSec secure tunnel has to be established between each pair of CEs belonging to the same VPN. Let's consider a VPN with a 100 different sites, and then each CE corresponding to a site needs the capability to concurrently establish 99 IPSec tunnels. Clearly, this VPN model will put a lot of management loads on the SP backbone network for supporting a large number of VPNs and customer sites.

Unlike the CE-based VPN model, the scalability of network IP based VPN is less dependent on the number of sites per VPN and the number of VPNs supported by each

PE. The scalability concerns focus on the capacity of the Provider edge (PE). Today's PEs have already been designed to maintain and process large global routing tables, so they can support well a certain number of VPNs. Another advantage is that the PE can aggregate a large number of small secure channels into a larger pipe (secure tunnel between edge routers - PEs) as illustrated in Figure 2.19. This issue will be discussed later in Chapter 4.

Figure 2.19 Comparison of Scalability for CE-based VPN Model vs IP Network-based VPN Model

The frequency of the information exchange has a huge impact on the scalability of the system. The tunnels between PEs need to be maintained and may have to be reestablished due to security concerns.

Security

The most important security requirement is to protect the customers' traffic over the shared network, because the customer may forward the data across different SP networks, which may be not fully trusted. In order to secure traffic by using VPN, two basic models are recommended: PE-to-PE (network based) and CE-to-CE (customer equipment- based). From a security point of view, the CE-to-CE security model provides maximum security for the customer. It achieves end to end secure protection for each VPN site separately as shown in Figure 2.17. But it is difficult to design a scalable network with the CE-to-CE model. Apart from its scalability issues, the CE-to-CE model is still not fully secure. A hacker can still spoof the traffic on CE-PE links, because PE is not able to authenticate the source (authentication is done in CE). Therefore, it is important to consider protecting the traffic in the shared infrastructure, to provide an overall manageable VPN and flexible secure choice for the customers.

In a network-based VPN, the customer can still choose the CE-to-CE protection. Besides, the customer can rely on the SP to secure the VPN traffic between PEs by using IPSec. In this model, a PE can establish a secure tunnel with each remote PE counterpart. This solution is more scalable because the number of security associations maintained per PE does not depend on the number of VPN, but only on the number of PE.

This results in a trade-off between security and scalability. With the same amount of memory and processing capabilities, an increase in the number of cryptographically secured VPN tunnels will cause a decrease in the overall performance.

2.3.3 Traditional Key Distribution and Public Key

Today millions of people use cryptography on Internet for their credit cards and other sensitive information to avoid the undesirable effects of spying on the shared network infrastructure. Except for the cost of encryption, the key distribution is another main problem for users who use encryption to secure their communications.

The classic key distribution or exchange method is to exchange the keys over an independent secure channel. [24, 35, 42] Users on each end of the communication share the same key, called *symmetric key*, and both have to agree on the same encryption and authentication algorithms. Both ends will hold the key till the communication is completed. As indicated in Figure 2.17, the secure tunnel and the key exchange must be achieved separately. This classic method is slow, expensive and affects the performance of the entire network. However it is more secure from the user's point of view.

A public key infrastructure [2, 24, 35, 42], which is called *asymmetric key* system, has two keys, one for enciphering (public key value) and one for deciphering (private key value). They are related but it is not easy to derive the deciphering key from the enciphering key. The enciphering key can be made public so that anyone can encipher his messages but only the specific recipient who holds the private key can decipher these messages.

A well-known known asymmetric key algorithm, Diffie-Hellman, is assumed to be applied for the key generation and exchange in this research [16, 35, 40, 41]. The algorithm has two system parameters, p and g . They are both public and may be used by all the users in a system. Parameter p is a prime number and parameter g (usually called a generator) is an integer less than p . The length of the prime p in octets is the integer k restricted by the following condition:

$$2^{8(k-1)} \leq p < 2^{8k} \quad (2.1)$$

Private-Value and Public –Value Generation

An integer x , the private value, shall be generated privately and randomly. This integer shall satisfy $0 < x < p-1$. The value g shall be raised to the private value x modulo p to give an integer y , the integer public value as show in Equation 2.2.

$$y = g^x \bmod p, \quad 0 < y < p \quad (2.2)$$

Suppose that both communicating parties generate the public/private key values, x_1 and x_2 , as described above. Then both parties exchange the public values. After both parties get the peer's public value, they re-compute the public key for the encryption as shown in Equation 2.3.

$$\begin{aligned} g^{(x_1)x_2} &= Key_{(shared-public)} \\ g^{(x_2)x_1} &= Key_{(shared-public)} \end{aligned} \quad (2.3)$$

This shared public key will be exchanged with the help of the Internet Key Exchange protocol defined by IPSec (discussed in section 2.23). Each party will hold their own private key value (un-exchanged) to decrypt the secure data.

Chapter 3 Network Simulator

This chapter gives an overview of the network simulator (*NS*) used for this research and also gives the overall approach for building the simulation model in the thesis. Section 3.1 describes the *NS* background, the architecture of the *NS* environment and the main components. Section 3.2 discusses the development of an IKE simulation model based on the current *NS* implementation. The model focuses on the main mode exchange and the quick mode exchange described in chapter 2. Section 3.3 describes briefly the network topology generator – GT-IMT.

3.1 Network Simulator – NS

3.1.1 NS Background

NS is a discrete event simulator used widely in the networking research [18]. *NS* provides substantial support for the simulation of TCP, routing, and multicast protocols over wired and wireless networks. Since 1989, *NS* has been developed to support different platforms: UNIX, LINUX and Microsoft Windows by a joint effort of UC Berkeley, Xerox PARC, SUN Micro system, USC/ISI and private researchers. The following table gives a snapshot of the protocols covered by the latest version *NS* [18].

Table 3.1 Classes of *NS* Protocols

Application Level	<ul style="list-style-type: none"> • HTTP, web caching and invalidation, TcpApp • Telnet and ftp sources • Constant-Bit-Rate (CBR) sources
Transport Level	<ul style="list-style-type: none"> • TCP • UDP • RTP (real time protocol) • SRM (scalable reliable multicast)
Network Level	<ul style="list-style-type: none"> • Static routing - Dijkstra's all-pairs SPF algorithm • Distance vector routing • Centralized Multicast routing – PIM-DM • Broadcast routing
Link Layer Level	<ul style="list-style-type: none"> • LANs, with CSMA/CD MAC protocols • Snoop

3.1.2 NS Architecture

NS is an object-oriented simulator, whose kernel is written in C++ and uses OTcl as command language. OTcl is used for configuration, network setup and manipulation of C++ objects for simulation. It basically provides the user interface to control the C++ functions. However, all the functionality for packets inbound and outbound processing and all the details of protocols are implemented in C++.

NS designers built this simulator in such a way that it supports two hierarchy classes: one class in C++ called compiled hierarchy and the other in OTcl called interpreted hierarchy. The two hierarchies are closely associated with each other. From the user's perspective, there is a one-to-one correspondence between the interpreted and the compiled hierarchy. A new simulator object created in the interpreter will be mirrored into a C++ class. The relationship between C++ and OTcl objects is illustrated in Figure 3.1.

Figure 3.1 Dual Objects Built in NS Appear in both C++ and OTcl Classes

Obviously, it is extremely important to establish appropriate linkage between OTcl and C++ objects because of the *NS* dual language architecture. The next several sections will describe the fundamental concept on how to link the objects, methods and variables between C++ and OTcl. Here we only focus on three classes used in *NS*: *TclObject*, *TclClass* and *Tcl*.

Class TclObject

The class *TclObject* is the root of *NS* object hierarchy and the base class for all simulator objects that are also mirrored in the compiled hierarchy. To link the C++ member variables to OTcl object variables, two basic built-in methods *bind* () and

command () are developed in *NS* to link the variable values and the access of methods to C++ via OTcl.

In the following example, the user can access OTcl variable (line 3) *packetSize_* via method **bind** (). The user can set any value for this variable in his simulation (line 6) and the same value will be set to corresponding C++ variable *size_*.

In C++:

```

1      IkeAgent::IkeAgent() : Agent(PT_IKE)
2      {
3          bind("packetSize_", &size_);
4      };

```

In OTcl:

```

5      set ike [new Agent/Ike]
6      $ike set packetSize_ 512

```

In the next example, C++ defines the OTcl procedure **negotiate** in method **command()** in line 4. When this procedure is instantiated in simulation (line 17), method **command()** will be invoked and if the string of name of this procedure matches the **negotiate** (line 4), C++ will execute the method **sendadv()** (line 5). Otherwise, it returns the base class of the **command()** (line 13).

In C++:

```

1      int IkeAgent::command(int argc, const char*const* argv)
2      {
3          if (argc == 2) {
4              if (strcmp(argv[1], "negotiate") == 0) {
5                  sendadv ();
6                  return (TCL_OK);
7              }
8              if (strcmp(argv[1], "group_negotiate") == 0) {
9                  sendaggregation ();
10                 return (TCL_OK);

```

```

11         }
12     }
13     return (Agent::command(argc, argv));
14 };

```

In OTcl:

```

15     set sender [new Agent/Ah]
16     $ns attach-agent $n($a1) $sender
17     $ns "$sender negociate"

```

Class TclClass

TclClass defines the interpreted class hierarchy and the methods to permit the user to instantiate TclObjects. *TclClass* is written in C++, a pure virtual class. It provides two main functions: construct the OTcl class hierarchy to mirror the C++ class hierarchy and instantiate new TclObjects. An example code of *IkeClass* derived from base class *TclClass* is shown below. It demonstrates how *TclClass* is associated with the corresponding compiled class (C++) and how it instantiates a new object.

From the code given above it can be observed that:

- The class *IkeClass* defines a constructor and one method *create()* that instantiates an instance of *TclObject*.
- When a static object *class_ike* is created, the constructor defines the interpreted class named as *Agent/Ike* into OTcl.
- This class is associated with the class *IkeClass* and it creates a new object when *create()* method returns *TclObject* in the class *IkeAgent*. Note that when the new *Agent/Ike* is specified, the routine *IkeClass::create()* is invoked.

Tcl Class

This class provides the methods used in C++ code to access to and communicate with the OTcl interpreter. For example, *tcl.result()* is used to pass results to/from the interpreter; *tcl.eval()* is used to invoke OTcl procedure from the compiled (C++) side and etc.

3.1.3 Network Simulation Basics in NS

Each simulation requires a single instance of class *Simulator* to control and operate the simulation. The overall simulator provides a set of interfaces for configuring a simulation and for choosing the type of the event or the scheduler to drive the simulation. A simulation script generally begins with creating an instance of this class and calling various methods to create nodes, topologies, and configure other aspects of the simulation. In order to build an NS model and conduct simulation experiments, one should take care of three basic aspects: create the network topology, set up the routing and create the protocol agents.

Create the Topology

One aspect of creating a topology in *NS* is to create the network nodes. The basic primitive to construct a node is as follows:

```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
```

By default, nodes in *NS* are constructed for the unicast simulation. The basic structure of a node is illustrated in Figure 3.2. Each node is constructed out of multiple classifier objects, such as address classifier, port classifier etc. The function of *classifier* class is to look at a specific portion of the incoming packet, usually its destination address, and map the values to an outgoing interface object that is the next hop for this packet and forward the packet through the node.

Figure 3.2 Structure of Unicast Node

The second aspect of creating a topology is to define a link that connects two nodes. Class **Link** provides the ability to create a point-to-point link between two nodes.

Two subclasses, *SimpleLink* and *DuplexLink*, are derived from this base class. *NS* provides two instance procedures, *simplex-link* and *duplex-link*, to establish an unidirectional and bi-directional link between nodes, respectively. The syntax for defining a simplex or duplex link is as follows:

```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
$ns simplex-link $n0 $n1 <bandwidth> <delay> <queue_type>

OR

$ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>
```

Such a command creates a link from node0 to node1 with specified bandwidth and delay characteristics on the link, and defines one of the possible queue schemes for the link. This procedure also creates a TTL checker for this link. The overall structure of a duplex-link is presented in Figure 3.3. The definition of the instance variables listed in this figure, such as *enqT_*, *link_* etc, are given in the *NS* manual [18].

Figure 3.3 Composite Construction of a Duplex Link

Routing Setup

Routing is another basic requirement for the simulation in *NS*. Generally, two types of routing protocols are supported by *NS*: Unicast Routing and Multicast Routing.

For Unicast Routing, *NS* provides three mechanisms: Static, Session and Dynamic routing. Static and session routing use Dijkstra's all-pairs SPF algorithm, whereas the dynamic routing is based on Distributed Bellman-Ford (Distance Vector) algorithm. The routing structure is illustrated in Figure 3.4.

Figure 3.4 Structure of Unicast Routing between Two Nodes with Duplex Link

Currently the static route computation is the default mechanism in *NS*. This algorithm runs exactly once prior to the start of the simulation by computing the adjacency matrix and linking the cost of all links. Session routing is almost identical with the static routing, with the difference that it recomposes the routes when the topology is

changed. Therefore, the session routing has the capability of instantaneously recomputing and updating routing changes in the present topology. Distance Vector (Dynamic) routing mechanism is implemented to update the routes periodically, and the default value is 2 seconds. In addition to the updates, this mechanism also sends updates whenever the routing table changes in a given node. The following is an example of commands for configuring the static routing mechanisms for a simulation.

```
$ns rtpproto <type>
<type>: Static, Session, DV
```

Multicast Routing is an enhancement of Unicast Routing. When Multicast Routing is enabled, *NS* creates additional classifiers and explicators for each node besides *address classifier* and *port classifier* shown in Figure 3.2. The current *NS* version supports three multicast algorithms: Centralized multicast (PIM-SM), Dense mode (PIM-DM) and Shared tree mode. Users can enable multicast in two ways in their simulation. The following code is an example of enabling multicast by two different means, as highlighted.

```
set ns [new simulator - multicast on]

or

set ns [new Simulator]
$ns multicast
$ns mrtproto <type>
<type>: CtrMcast, DM, ST
```

Agents

An important basic component in the simulation is the creation of Agents. Agents represent endpoints where network-layer packets are constructed or consumed, and are used in the implementation of protocols at various layers. All C++ code agents are derived from the base class *Agent* that supports the packet generation and reception.

Users can specify an agent for any transport or application level protocols by overwriting or adding member functions. One member function is generally not over-ridden by the derived class:

Packet allocpkt t()* - allocate new packet and assign its fields

At least two basic member functions have to be implemented by an user who wants to define a new agent in *NS*: inbound packet and outbound packet processing. The following Figure 3.5 gives the overall structure of a point-to-point connection between two nodes and the packet flow. A TCP Agent is attached to each node.

Figure 3.5 Bi-direction Traffic Flow between Two Agents Attached on Nodes

Overall, the users of *NS* need to create the topology (nodes and links), to define the routing scheme, to attach the specific agents and/or application agents and to execute the simulation. The following is an example of the script that simulates the network shown in Figure 3.5.

```

1      set ns [new Simulator]                # instantiate new simulation object
2      set n0 [$ns node]                     #create unicast nodes
3      set n1 [$ns node]
4      $ns duplex-link $n0 $n1 1.5Mb 10ms DropTail
                                           #establish duplex link
                                           #between two nodes and
                                           #specify characters on this
                                           #link

5      set tcp [new Agent/TCP]
6      set sink [new Agent/TCPSink]

                                           #initiate new TCP agent
7      $ns attach-agent $n0 $tcp             #attach agents on two nodes
8      $ns attach-agent $n1 $sink
9      $ns connect $tcp $sink                #connect two nodes by using
                                           #static routing
10     set ftp [new Application/FTP]         #initiate new application
                                           #layer agent FTP
11     $ftp attach-agent $tcp                #attach FTP agent on top of
                                           #TCP agent on n0
12     $ns at 1.0 "$ftp start"               #start simulation at 1 second

```

3.2 Creation of New Protocol for NS

In this section, we will discuss how to develop a new agent based on the framework provided by *NS*. As described in the previous section, agents are used in the implementation of protocols and all the protocol functionalities are implemented in C++. *NS* provides a flexible hierarchy for users to add any specific agent. Eventually a user can add all the IP-based protocols to *NS*. Figure 3.6 illustrates the current hierarchy of *NS* implementation and indicates where the new agents fit.

Figure 3.6 Partial C++ Class Hierarchy in NS

The basic actions required to create an agent are summarized as follows:

1. Decide its inheritance structure and what it is derived from, related to the layer at which this agent will operate. Class **Agent** is the base class of all agents.

For example:

```

class IkeAgent : public Agent {
public:
    IkeAgent();

    virtual int command(int argc, const char*const*
                        argv);
    virtual void recv(Packet*, Handler*);
    virtual void sendadv();
    virtual void sendaggregation();
};
  
```

2. Define the structure of the new header for the specific agent. For example:

```

struct hdr_ike{
    long cookie_i_;
    long cookie_r_;
    char next_payload_;
    char version_;
    char ex_type_;
};
  
```

```

        char flag_;
        long msg_id_;
        long msg_len_;
    };

```

3. Define the *recv ()* and *send ()* methods. Their actions are to process the received packets and forward them. The simple example of *recv ()* method is defined as follows:

```

void IkeAgent::recv(Packet* pkt, Handler*) {
    /* Access the IKE header */
    hdr_ike *ike_hdr = hdr_ike::access(pkt);

    /* Access the IP header */
    hdr_ip *ip_hdr = hdr_ip::access(pkt);

    /* record the fields from received the packet */
    long ci = ike_hdr -> cookie_i_;
    char np = ike_hdr -> next_payload_;
    .....

    /* Discard the packet */
    Packet::free(pkt);

    /* Initialize new IKE packet */
    Packet *rep_pkt = allocpkt();
    hdr_ike* rep_ike_hdr = hdr_ike::access(rep_pkt);

    /* build the exchanging message */
    rep_ike_hdr -> cookie_i_ = ci;
    rep_ike_hdr -> next_payload_ = np;
    .....

    /* Forward the new packet to sender */
    send(rep_pkt, 0);
};

```

4. Define the linkage between the C++ and OTcl. As discussed in earlier sections, three items must be handled to link the agent with OTcl. First, establish a mapping between the OTcl name for the class and the actual object created. Secondly, set the link between C++ member variables and corresponding variables in the OTcl name space. Thirdly, once the object and variable binding is completed, create a *command()* method in C++, which

will be invoked from OTcl. A more detailed discussion and example codes are given in Section 3.2.

5. In some cases, specific OTcl code needs to be created to access the new agent and to define the user's own output..

3.3 GT-IMT Topology Generator

A tool kit called GT-ITM (Georgia Tech Inter-network Topology Model), embedded within the *NS* package, is used to generate random graphs for modeling the network topology [28]. GT-ITM is built on the top of the Stanford Graph Base (SGB) [18], a platform of data structures and routines for representing and manipulating graphs. GT-ITM is capable of generating three different types of topology graphs: flat random, hierarchical, and transit-stub graph.

The syntax of a script for generating a transit-stub topology graph is given below.

```
<method keyword> <number of graphs> [<initial seed>]
<# stubs/xit> <#t-s edges> <#s-s edges> <node-connectivity>
<n> <scale> <edgemethod> <alpha>
```

where,

<method keyword> is one of:

- 1) "geo": flat random graph;
- 2) "hier": N-level hierarchical graph;
- 3) "ts": transit-stub graph;

<number of graphs>: number of graphs of specified type to generate;

<initial seed>: initial random number seed;

<# stubs/xit>: avg number of stub domains attached per transit node

<#t-s edges>: number of extra transit-stub edges

<#s-s edges>: number of extra stub-stub edges

<node-connectivity>: possibility of link established between two nodes

<n>: number of nodes in graph

<scale>: one-sided dimension of space in which nodes are distributed

<edgemethod>: method for generating edges; valid range 1 to 6

1: Waxman 1

2: Waxman 2

3: Pure random (used to generate the topology for this research)

4: Doar-Leslie

5: Exponential

6: Locality

<alpha>: random graph parameter ($0.0 \leq \alpha \leq 1.0$)

Although the transit-stub model may not explicitly reflect the structure of real networks, it is attractive for its simplicity and is commonly used to study networking problems. We selected this model for this research. It provides a random hierarchical graph by composing interconnected transit and stub domains as illustrated in Figure 3.7. The next chapter will discuss the details of modeling selections and simulation design used in the thesis.

Figure 3.7 API of Transit-Stub Model Graph Created by GT_IMT

In this Figure, each node in a *transit* domain models a backbone node in the public network. To each transit domain node are attached a number of random graphs, each representing a *stub* domain. Six network topologies with different parameters were created for the simulation experiments performed for the thesis, and will be described in more detail in the next chapter.

Chapter 4 Aggregation Approach for Key Exchange

This chapter proposes an aggregation approach for key exchange that can be applied to network-based VPN. The goal of this approach is to increase the scalability of key management, which in turns contributes to the overall VPN scalability. The first section proposes the new approach for the key exchange, which represents a trade-off between the scalability and security. The second section discusses the design of the *NS* simulation model. Simulation scenarios and other simulation considerations are also discussed in this section.

4.1 Aggregation Approach for Key Exchange

The various VPN models and key exchange schemes described in section 2.3 bring up an important issue: the scalability of the information exchange over the public infrastructure [4, 8, 21]. This is part of an on-going problem for the service providers: how to establish a manageable, scalable and secure public network.

When passing encrypted data around in a public network, users are concerned with the risk that an attacker can read their "secret" data. Users can reduce this risk by using larger and larger encryption keys. However, the larger the key is, the slower and

more complex the encryption/decryption is. This certainly impacts the entire network performance.

Another basic difficulty is to keep track of all the keys issued. The more keys one has, the harder it is to keep track of them. As the traditional Key Negotiation and Exchange scheme stands, both sides of the communication will have to exchange all the information until both agree.

Let's imagine that 20 stock traders are interested in talking to each other privately and safely through the public network. If they use a separate key for every private communication, each trader will need 19 keys to communicate with the other members of the group. In total, the system has to keep track of 190 keys and 190 key exchanges. Obviously, the key exchange will be more complicated if more traders join in the group, or if they want to change the keys more frequently. This gives an idea of the magnitude of the problem in a large public network.

As described in section 2.2, IKE protocol is the standard for key exchange, authentication and encryption algorithm negotiation. All this information exchange is completed in Phase one, which adds up to a six-message exchange. Phase two containing a three-message exchange, actually swaps the SPI value and confirms the protocol that both ends agreed on. If using symmetric key, every user has to generate his own key and repeat this process of two-phase exchange, because the sender and the recipient use the same key to encrypt and decrypt data.

To improve this situation, a new approach, named Aggregation Approach for Key Exchange is proposed in this section. The first assumption is that asymmetric encryption rather than symmetric is used. With the asymmetric scheme, each user can generate a public/private pair key, as described in section 2.3. The sender and the recipient only exchange the public key over the public Internet, and each holds his own private key. Thus, both the sender and recipient will have the combination of two keys information: the peer's public key and his own private key. The sender uses the peer's public key to encrypt the data and forward the packet to the recipient. The recipient uses his own private key to decrypt the data when he receives the packet. This does not reduce the amount of information exchanged if every pair of users exchange their own public keys.

Therefore, we need to have one entity able to manage all these negotiations for the users who are associated with it. As shown in Figure 4.1, the provider edge (PE) router is the one who plays this role. Each PE can generate a public key and negotiate to another PE to create the secure tunnel over the public network. For establishing each PE-to-PE tunnel, two public keys between those two PEs have to be exchanged. Assume that p PEs are deployed in one public network and every PE servers on average c CEs. If one PE wants to create secure tunnel with every other PEs within this public network, total $p(p-1)/2$ tunnels need be created and $p(p-1)$ public keys have to exchanged. In order to reduce the processing load for PE, we recommend to assigns public/private key values to the security gateways that are connected to this PE. Individual user's security relies on the service provider's edge router, and each PE is responsible for negotiating the key, algorithm, and protocol etc. on their behalf. Each CE can do the actual encryption and decryption of data, by using the key agreed-on by the local PE.

In this case, each PE will need fewer public keys to exchange with peer PEs and create the secure tunnel over the public network. Different VPNs may share the same secure path between a pair of PEs. The following paragraphs give an example how this aggregation approach works.

Figure 4.1 Aggregated Key Exchange by Establishing Public Tunnel in Backbone Network when Using Public Key

As illustrated in Figure 4.1, a group of users of VPN1 are located at different sites and connected to the gateway CE1, CE3, CE4 and CE5. When any of users in CE1, CE3 and CE4 initiates a VPN communication to users in CE5, each provider edge PE1 and PE2 will generate the public/private pair key for this communication. Both public keys are exchanged between PE1 and PE2 and other information as well such as encryption,

authentication algorithm etc. After the phase one exchange (using IKE protocol), they will establish an encrypted tunnel between them. Now PE1 can assign this public/private pair keys to gateway CE1, CE2, CE3 and CE4. The same process can be done between CE2 and CE6. They can share the same public tunnel with CE1, CE2, CE3 and CE4. In this way, the number of keys generated and exchanged is reduced and each PE must take full responsibility to make sure the each pair of public/private key are safely generated and exchanged by using IPSec Protocol such as IKE, authentication algorithm (MD5) etc. Also, it is the responsibility of PE to make sure that the forwarding tables are correct and updated properly.

The proposed aggregation approach reduces the information exchange over the public network when large numbers of VPNs are supported. However, this comes at a price, in the sense that it reduces some of the security capabilities. The proposed approach represents a trade-off between security and scalability. The disadvantage of this approach is summarized as follows:

- Since the link between CE and PE is unprotected, it is possible that the key can be attacked on this local link. This will threaten the entire set of traffic flows within the secure channel between the respective pair of PEs, because all these traffic flows are encrypted and decrypted by the same pair of keys.
- All users of the same VPN can eavesdrop on each other's conversation.
- Client can change destination address of particular data stream in PE due to the same key and cause PE to send data to the wrong destination.
- It is lack of multi-level encryption

The proposed aggregation approach also provides some flexibility in realizing different security levels, always by making a trade-off between security and scalability.. Basically, it is possible to implement several hierarchical key exchanges to strengthen the security. The following two bullets can give some levels of security enhancement for the aggregation approach.

- In phase two exchange, the sender and the recipient can generate the new public/private pair keys and exchange them to enhance the security before they transmit the data.
- On each VPN customer site behind the gateway, users can be divided into different trust groups. One group will share the same public key rather than the entire VPN sharing the same public key. In this way, PE will have to do multiple negotiations and exchanges for different groups, which represents the security/scalability trade-off.

4.2 The Simulation Model

This section will discuss the design and implementation of the *NS* simulation model for the aggregation approach for key exchange introduced in the previous section. This section is composed of four main blocks: development of IKE agent, the design of the network topology, the design of the simulation model and other simulation considerations.

4.2.1 Development of IKE Agent

The code development is contributed in two parts: C++ and OTcl. The C++ code is responsible for processing incoming and outgoing packets. It also performs the key exchange sequence according to a) the traditional scheme, and b) the aggregation scheme. The OTcl hierarchy is implemented to control the simulation for different simulation scenarios.

Development of IKE Agent in C++

As discussed in chapter 3, the basic functionality for each agent object is to process the inbound and outbound IP packet. Thus, two main functions have been implemented in IKE agent object: packet sending and receiving methods.

IKE Agent object is able to process the receiving packet and construct the new packet using different message specified by standard. According to RFC2408 and RFC2409, thirteen types of messages are defined in the implementation, such as Hash, Nonce, IKE SA, Key etc. They are used to construct the exchange packet according to the exchange modes. Here, two exchange modes are implemented: main exchange mode (shown in Figure 2.11) and aggressive exchange mode (shown in Figure 2.12). Each IKE agent constructs the proper chain of messages to respond to the peer based on the received message. When the initiator starts the key exchange, it will set one timer till the exchange sequence is completed. At the end of this exchange, the last operation is to calculate and output the propagation time used to complete the exchange.

```
sprintf(out, "%s recv %d %3.1f", name(),
ip_hdr->src_.addr_ >>  Address::instance().NodeShift_[1],
```

```
(Scheduler::instance().clock()-ike_hdr->rtt())*1000);
Tcl& tcl = Tcl::instance();
tcl.eval(out);
```

In the IKE header, two additional important variables are introduced: *msg_id_* and *opt_*. Message Id is used to indicate which message has been received during the sequence and option flag, either 0 or 1, is used to indicate whether traditional or aggregation exchange approach has been selected during the simulation.

4.2.2 Internetworking Topology Modeling

A proper network topology needs to be generated to simulate the IKE schemes. as discussed in section X, the network topology is an abstraction or a model of the actual network structure. Typically the topology is modeled by a graph in which each node represents the switch or router and edges represent the connection between pairs of nodes.

Almost all the networks are modeled by using random graphs [9, 46]. The standard graph model distributes the vertices at random locations in a plane and adds edges between a pair of vertices with a given probability factor. Even though this random model has little constraint over the topology structure, it can not reflect accurately the hierarchical nature of the real network.

The transit-stub model, a hierarchical random graph, is designed to overcome this shortage. It creates a topology by composing interconnected transit and stub domains. A simple example of transit-stub model structure is illustrated in Figure 4.2. Each node in stub domain represents an individual PC or workstation in a LAN and each node in a Transit domain represents the router or switch in the backbone infrastructure. The

number of transit domains, the number of stub domains connected to one transit domain and the connectivity between the stub-to-stub and stub-to-transit can be randomized by users. Therefore it is not difficult to model various graph sizes. However, the connectivity degree is applied to the entire topology. Stub and transit's connectivity can't be controlled separately.

Figure 4.2 Transit-Stub Domain Random Graphic Structure

In a VPN system, the traffic initiated by different customer sites collected by VPN gateway is sent across the backbone shared infrastructure and reaches the different recipients via the destination gateways. The transit-stub graph provides the proper structures for simulating the network that offers VPN services. In order to compare the performance of the traditional and aggregation key exchange approaches, six network topologies were generated by using the GT-IMT tool. The parameters are the network

size, and the connectivity degree of transit and stub domains, as shown in the following table.

Table 4.1 Summary of Simulation Topology Characteristics

Topology	Network Size - Nodes	Connectivity Degree
Topo_N50A	52	2.679
Topo_N50B	52	3.692
Topo_N100A	114	2.682
Topo_N100B	114	3.686
Topo_N150A	150	2.684
Topo_N150B	150	3.69

The topologies from the table are given in Figure 4.3 to Figure 4.8, respectively.

The node connectivity in those topologies is defined as the average possibility of the connection between two nodes.

Figure 4.3 Topo_N50A with 52 Nodes and Node Connectivity Degree 2.679

Figure 4.4 Topo_N50B with 52 Nodes and Node Connectivity Degree 3.692

Figure 4.5 Topo_N100A with 114 Nodes and Node Connectivity Degree 2.682

Figure 4.6 Topo_N100B with 114 Nodes and Node Connectivity Degree 3.686

Figure 4.7 Topo_N150A with 150 Nodes and Node Connectivity Degree 2.684

Figure 4.8 Topo_N150B with 150 Nodes and Node Connectivity Degree 3.69

4.2.3 Simulation Model Design

This section describes the randomization of the model, as one of key features that need to be implemented. Another important component is to track all the communication sessions between two stub nodes and decide whether their communication session needs to be aggregated into an existing public tunnel as shown in Figure 4.4, or a new public tunnel needs to be established.

Randomization

- Randomized topology - this provides the various localities for the stub domain connected to each transit node and different localities for the nodes in every

stub domain. Also the possibility of the link between the two nodes is randomly placed depending on the average node degree specified by users.

- Randomized traffic – a traffic generator is attached to the nodes in stub domain to emulate different customers' activities in one shared infrastructure. Traffic generator is attached on the randomly selected node. That means it may have unbalanced traffic in the overall network and one node may have a lot more traffic generators than the other.
- Randomized communication session for the key exchange – each node that requests key exchange is randomly selected in a similar way as the traffic generator is selected. Therefore one customer may initiate multiple exchanges or may not request any at all. Also the initial starting time for each key exchange is randomly calculated based on the average time specified by the user. This will model more closely the key negotiation from the user's point of view.

Simulation Model for Traditional Key Exchange Approach

For the traditional approach, every communication initiated by a stub node must have independent secure tunnel for the key exchange. Each communication session follows the normal IKE protocol to exchange the information following the two phase exchange, according to the two modes (main mode and the aggressive mode) as described in section 2.2.3.2. The transit domain (backbone) nodes are not involved in any security management function. All the security key exchange is handled by VPN gateways, modeled by stub nodes.

Simulation Model for Aggregation Key Exchange Approach

The simulation model for this approach needs to track each communication session between a pair of stub nodes and decide whether it should go through the existing public tunnel across the transit domain or a new tunnel should be created. The basic rules for aggregation key exchange approach are as follows:

1. All nodes initializing the key negotiation that pass through the same pair of edge routers to reach the destination node will use the same backbone tunnel across the transit domain. The first user is responsible for establishing the public tunnel, as shown by the dotted line in Figure 4.4.
2. If the source and destination nodes are in the same stub domain they will not require a secure tunnel, as shown by the dash line in Figure 4.4.
3. Any node initializing the key negotiation that passes through only one edge router to reach the destination node does not require a public tunnel, as shown by the dash-dot box in Figure 4.4.

4.2.4 Traffic Design

One more piece of the puzzle that needs to fit in is the simulation of the random traffic across the entire network. Across one shared public infrastructure, many varieties of traffic exist at any moment. They have different rates, different types of traffic passing through same path or the same stream of data passing through different routes. At any moment a new customer may join in or an old customer may leave the system. The utilization of the bandwidth can be very different on the same link every moment.

To model the background traffic, NS has provided three different traffic modes: EXPOO_Traffic (exponential), POO_Traffic (Pareto distribution) and CBR_Traffic (Constant Bit Rate). EXPOO_Traffic was selected for our simulation experiments. This traffic-generator mode generates traffic according to an exponential ON and OFF distribution. Packets are sent at a fixed rate in every ON period and no packets are sent during OFF period. The restriction of this mode is that the packet size is fixed.

The following is an example of exponential on/off traffic generator with its specific parameters defined by *NS*.

```
Set Exp [new Application/Traffic/Exponential]
$Exp set packetSize_ 512
$Exp set burst_time_ 500ms
$Exp set idle_time_ 500ms
$Exp set rate_100k
```

Here is the definition of member variables:

packetSize_: the constant size of the packets generated
burst_time_: the average “ON” time for the generator
idle_time_: the average “OFF” time for the generator
rate_: the sending rate during “ON” time

Idle period and burst period alternatively is processed till the simulation is completed. During the simulation, an exponential traffic generator is randomly placed on the nodes in the stub domains.

4.3 Simulation Experiments

4.3.1 Key Parameters for the Simulation Experiments

Several parameters listed below were chosen as experimental factors. The goal of the simulation experiments was to study their impact on the performance of the key exchange protocol. A pre-determined set of values for these parameters were given as input for each simulation experiment.

- **Node Connectivity** – This parameter is used to examine the performance of key exchange for various connectivity levels in the network. It will give the service provider an idea how dense the connections among the core routers in the public network should be, and what level of mesh is more efficient for different cases. Two connectivity degrees have been applied to perform the simulation: a low water marker of 2.68 and high water marker of 3.69.
- **Size of Network** – The larger the network is, the longer it will take to communicate through it. A larger public network covers more area and can carry more data traffic. So the propagation delay becomes the main concern from the performance point of view. This research will examine three different sizes of network, around 50, 100 and 150 nodes. (The actual topologies used are given and have actually 54, 108 and 150 nodes).
- **Public Tunnel Lifetime** – As discussed before, the frequency of the information exchange has a huge impact on the entire network. This is always a trade-off between security and scalability. This research defines the lifetime of a public tunnel as a given number of single communication sessions

aggregated into the same public tunnel. After certain number of communication has been aggregated, this public tunnel is considered no longer secure. It will be torn down and re-established if needed.

- Latency – End-to-end protocol execution time will be calculated by the simulation. This is the output parameter for analyzing the performance of the key exchange for VPN system.

4.3.2 Simulation Cases

This section gives a summary of the simulation experiments that were performed to compare the two key exchanged approaches. The simulation results are analyzed in the next chapter.

Table 4.2 Summary of Simulation Cases

	Simulation Models
Case 1 - 2	Node: 54 (4 transit nodes; 50 stub nodes) Connectivity: 2.68 Traffic rate: 100 kpbs Lifetime of Tunnel: long lasting
Case 3 - 4	Node: 54 (4 transit nodes; 50 stub nodes) Connectivity: 3.69 Traffic rate: 100 kpbs Lifetime of Tunnel: long lasting
Case 5 - 6	Node: 108 (6 transit nodes; 102 stub nodes) Connectivity: 2.68 Traffic rate: 100 kpbs Lifetime of Tunnel: long lasting
Case 7 - 8	Node: 108 (6 transit nodes; 102 stub nodes) Connectivity: 3.69 Traffic rate: 100 kpbs Lifetime of Tunnel: long lasting

Case 9 - 10	Node: 150 (8 transit nodes; 142 stub nodes) Connectivity: 2.68 Traffic rate: 100 kpbs Lifetime of Tunnel: long lasting
Case 11 - 12	Node: 150 (8 transit nodes; 142 stub nodes) Connectivity: 3.69 Traffic rate: 100 kpbs Lifetime of Tunnel: long lasting
Case 13 - 14	Node: 54 (4 transit nodes; 50 stub nodes) Connectivity: 2.68 Traffic rate: 100 kpbs Lifetime of Tunnel: 5 and 10 sessions
Case 15 - 16	Node: 54 (4 transit nodes; 50 stub nodes) Connectivity: 3.69 Traffic rate: 100 kpbs Lifetime of Tunnel: 5 and 10 sessions
Case 17 - 18	Node: 108 (6 transit nodes; 102 stub nodes) Connectivity: 2.68 Traffic rate: 100 kpbs Lifetime of Tunnel: 5 and 10 sessions
Case 19 - 20	Node: 108 (6 transit nodes; 102 stub nodes) Connectivity: 3.69 Traffic rate: 100 kpbs Lifetime of Tunnel: 5 and 10 sessions
Case 21 - 22	Node: 150 (8 transit nodes; 142 stub nodes) Connectivity: 2.68 Traffic rate: 100 kpbs Lifetime of Tunnel: 5 and 10 sessions
Case 23 - 24	Node: 150 (8 transit nodes; 142 stub nodes) Connectivity: 3.69 Traffic rate: 100 kpbs Lifetime of Tunnel: 5 and 10 sessions

Note that cases 1 – 12 are applied to both the traditional and aggregation key exchange approaches, whereas cases 13 – 24 are applied only to the aggregation approach.

Chapter 5 Simulation Analysis

In this chapter, the results of different simulation experiments are presented and discussed. The first section describes how the confidence interval concept was used to evaluate the simulation results. The rest of the sections summarize and analyze the results for each simulation scenario given at the end of the previous chapter.

5.1 Using Confidence Intervals for Data Analysis

The analysis of the results of simulation experiments is based on the observation that the output data from a simulation exhibits random variability, because random number generators are used to produce the values for the input variables. That means that two simulation experiments of exactly the same model will generate different sequences of random numbers, and therefore will produce different sequences of output values. Usually one is interested in output value statistics, such as the mean value, variance, standard deviation, etc. For example, in order to approximate the mean value of a population, we will take a random sample from that population and compute its mean value. How well the sample statistic estimates the underlying population values is always an issue. A confidence interval addresses this issue because it provides a range of values, which is likely to contain the population parameter of interest.

Confidence Intervals

Confidence limits or intervals for the mean are an interval estimate of the mean [6]. Interval estimates are often desirable because the estimate of the mean varies from sample to sample. Instead of a single estimate of the mean, a confidence interval generates a lower and upper limit for the mean. The interval estimate gives an indication of how much uncertainty there is in our estimate of the true mean. The narrower the interval, the more precise is our estimate. Confidence intervals are constructed at a confidence level selected by the user, for example 95%. It means that if the same population is sampled on numerous occasions and interval estimates are made each time, the resulting intervals would bracket the true population parameter in approximately 95% of the cases. A confidence stated at a $1-\alpha$ level can be thought of as the inverse of the significance level, α .

It should be noted that a 95% confidence interval does not mean that there is a 95% probability that the interval contains the true mean. The interval computed from a given sample either contains the true mean or it does not. Instead, the level of confidence is associated with the method of calculating the interval. The confidence coefficient is simply the proportion of samples of a given size that may be expected to contain the true mean. That is, for a 95% confidence interval, if many samples are collected and the confidence interval computed, about 95% of these intervals would contain the true mean.

Confidence limits or intervals are defined as follows:

“If one constructs a very large number of independent percent confidence intervals each based on N observations, where N is sufficient large, the proportion of these confidence intervals that contain θ should be $1-\alpha$. We call this proportion the coverage for the confidence interval. – Law and Kelton (1991)”

An approximate confidence intervals is given by following equation:

$$\bar{M} \pm t_{(\alpha/2, N-1)} \frac{\sigma}{\sqrt{N}} \quad (5-1)$$

where, $\bar{M}$ is the sample mean, σ is the sample standard deviation or variance, N is the sample size, α is the desired significance level and $t_{(\alpha/2, N-1)}$ is the upper critical value of the t distribution with $N-1$ degrees of freedom. Note that $1-\alpha$ is the confidence coefficient.

As shown Equation 5-1, it is clear that the width of the interval is controlled by two factors:

- As N increases, the interval gets narrower from the $\sqrt{N}$ term. One way to obtain more precise estimates for the mean is to increase the sample size.
- The larger the sample standard deviation, the larger the confidence interval. This simply means that data with large standard deviation generate wider intervals than data with a smaller standard deviation.

Confidence Intervals with a specified precision

Suppose an error criterion ε is specified by the user, which means that it is desired to estimate θ to within $\pm \varepsilon$ with a probability of at least $1-\alpha$. Assume that an initial

sample size N_0 has been observed, which are considered as N_0 independent replications. The N_0 replications will be used to obtain an initial σ_0 . To meet the half-length (h.l.) criterion, a sample size N has to be chosen with $N \geq N_0$. Therefore it gives:

$$h.l. = \frac{z_{(\alpha/2, N-1)} \sigma_0}{\sqrt{N_0}} \leq \varepsilon \quad (5-2)$$

where:

$z_{(\alpha/2, N-1)}$: cumulative normal distribution of confidence intervals

N : the sample size

σ_0 : the variance of initial sample

ε : the half-length error

So solving for N in Eq. 5-3 we obtain the smallest value of N satisfying the condition $N \geq N_0$. N can be determined by:

$$N \geq \left(\frac{t_{(\alpha/2, N-1)} \sigma_0}{\varepsilon} \right)^2 \quad (5-3)$$

Where:

$1-\alpha$: confidence interval (e.g., $\alpha = 0.02$, $1-\alpha = 0.98$)

$t_{(\alpha/2, N-1)}$: the $100(1-\alpha/2)$ percentage point of a t -distribution (the value can be found by looking up the t -distribution table in most statistics manual).

After determining the final N , collect the additional replications $N - N_0$ or start over to make N replications. Finally, the half-length of the confidence interval is given by Eq. 5-4:

$$h.l. = \frac{t_{(\alpha/2, N-1)} \sigma}{\sqrt{N}} \quad (5-4)$$

This value must be smaller than the precision ε specified by user. However, with the additional $N - N_0$ replications, the variance σ may differ from the initial σ_0 . If the difference is too large, the procedure should be repeated to determine an even larger sample size N .

Procedure for Data Evaluation

Step 1: Run the Tcl program to perform a simulation experiment:

```
ns tradition_ike_n52_2xxx.tcl
```

Step 2: Estimate the initial sample size

In order to get accurate results, the sample size needs to be estimated. The recommended size (i.e., the number of independent replications) of the sample is at least 5, but 10 or more are recommended in [6], depending on the accuracy of the sample data and the final size required. In this research, for each scenario ten statistically independent simulation runs were executed (i.e. $N_0 = 10$).

Step 3: Calculate the mean of the key exchange time (the performance measure used to control the length of the simulation experiments in our case)

$$\bar{M} = \frac{1}{r} \sum_{i=1}^r M_r, \quad r = 1, \dots, N \quad (5-5)$$

where:

$\bar{M}$ is the sample mean of the entire experiment;

M_r is the sample mean of batch r ;

r is the number of batch and $r = N_0$ for this case;

N is the number of batches (replications).

Step 4: Calculate the sample variance σ_0 of N_0 runs

The sample variance σ_0 is defined by the following equation:

$$\sigma_0^2 = \sum_{r=1}^n \frac{(M_r - \bar{M})^2}{n-1} \quad (5-6)$$

Step 5: Estimate the sample size N

Given a half-length for a confidence interval, the sample size can be estimated to meet this criterion. In this research, a confidence interval at the 98% level for a half-length error is set 2% of sample mean for the simulations and first estimate for sample size can be calculated by Eq. 5-2.

Step 6: Adjust the estimated value of the sample size

Step 5 actually gives a lower limit for the possible value of the sample size. The final value of the sample size can be obtained by solving the inequality given in (5-3). If $N \geq N_0$, additional replications of $(N-N_0)$ are needed to be run for this simulation scenario, and the mean value of the sample recalculated. Otherwise, the estimate value of sample size is good enough to give results with the desired accuracy.

5.2 Analysis of Simulation Results

This section analyses the simulation results obtained for different scenarios given at the end of the previous chapter. The main performance measure is the mean time for completing the key exchange as a function of different parameters such as network size, degree of connectivity of the network, frequency of exchange, etc. The results for both the traditional and the proposed aggregation approach are compared under different conditions.

5.2.1 Simulation Output

The average key exchange time for both the traditional and aggregation approach for different network sizes are summarized in Table 5.1 and 5.2. In these cases, the keys negotiated and exchanged last until the communication session is completed. The analysis for different key lifetimes is addressed separately in the section 5.2.3.

Table 5.1 Key Exchange Time for Different Network Size with Node Connectivity of 2.68

Network Size	Key Exchange Time [ms]	
	Traditional	Aggregation
52	8535.224	3293.853
108	12662.529	4802.805
150	13681.08	5100.283

Table 5.2 Key Exchange Time for Different Network Size with Node Connectivity of 3.69

Network Size	Key Exchange Time [ms]	
	Traditional	Aggregation
52	7400.242	2826.052
108	11067.013	4195.087
150	11068.183	4118.231

5.2.2 Relationship between Network Size and Key Exchange Time

Figure 5.1 shows the average key exchange time (both traditional and aggregation) measured for different network sizes with node connectivity of 2.68. Figure 5.2 shows a similar set of results for networks with a higher connectivity degree.

Figure 5.1 Relationship between Key Exchange Time and Network Size with Lower Connectivity Degree

Figure 5.2 Relationship between Key Exchange Time and Network Size with Higher Connectivity Degree

Figure 5.3 shows in the same chart the results from Figure 5.1 and 5.2, giving the complete view of the effect of network size and connectivity degree on key exchange performance. The following can be concluded:

- The average key exchange completion time increases with the network size
- The average key exchange completion time is reduced significantly when the aggregation approach is applied. As illustrated in the figures, the negotiation and key exchange time is improved by approximately 62%.
 - For 52-node topology, the average performance improvement is 61%
 - For 108-node topology, the average performance improvement is 62%
 - For 150-node topology, the average performance improvement is 63%

Figure 5.3 Key Exchange Time Versus Network Size for different key exchange approaches and connectivity degrees

5.2.3 Relationship between Node Connectivity and Key Exchange Time

In general, an increasing number of links in a network of the same size lead to a lower congestion through the network. If the network is designed as a full mesh topology, the traffic has more paths to select for reaching the destination by IP routing protocol. Figure 5.4 shows two curves: the top one is the traditional scheme key exchange time for a lower node connectivity, whereas the bottom one is the similar result for a higher node connectivity. In Figure 5.5 similar charts are presented, but the results are for the aggregation scheme.

Figure 5.4 Effect of Node Connectivity Degree on the Performance of the Traditional Key Exchange Scheme

Figure 5.5 Effect of Node Connectivity Degree on the Performance of the Aggregation Key Exchange Scheme

From Figure 5.4 and Figure 5.5, the following conclusions can be drawn:

- In networks with a higher connectivity degree, the key exchange and negotiation will take less time to complete.
- When the network size grows, a higher connectivity degree is needed to improve the performance. As seen from the simulation output, a change in node connectivity from 2.68 to 3.69 leads to 19% performance improvement for the 150-node network, and to 13% improvement for the 52 and 108-node network.
- Thus, higher connectivity is recommended for larger network to improve the performance.

5.2.4 Effect of Key Exchange Frequency

As discussed in Chapter 4, the frequency of control information exchange has a main impact on the scalability of VPN. In the set of simulation experiments described in this section, the key lifetime is defined as the maximum number of subscribed users allowed to use the same public key. If the number of users making use of the same public key to exchange their information reaches the pre-defined "lifetime" value, it is considered that this channel is no longer secure and renegotiation will take place.

As expected, the more often the exchange takes place, the longer the time taken to execute the exchange. However, we do not know how bad the impact will be on the new proposed aggregation approach. Table 5.3 summarizes the simulation output for evaluating the scalability with different lifetime values.

Table 5.3 Key Exchange Time with Different Lifetime Values

Lifetime = 5			Lifetime = 10			Lifetime >> 10		
Node	Degree: 2.68	Degree: 3.69	Node	Degree: 2.68	Degree: 3.69	Node	Degree: 2.68	Degree: 3.69
52	3423.89	3071.69	52	3382.90	2968.96	52	3293.85	2826.05
108	5235.68	4615.25	108	5000.21	4378.42	108	4802.83	4195.09
150	5491.47	4534.91	150	5254.37	4118.24	150	5100.28	4027.28

In Figure 5.6 through 5.8, each set of curves represents the average key exchange time for different key lifetimes in different topologies for the aggregation approach. From this set of charts, we can observe that:

- The average key exchange completion time decreases when the exchange becomes less frequent.
- For larger networks between 52-node topology and 108-node topology as shown in Figure 5.7 and 5.8, the frequency of information exchange has more influence on the performance of the key exchange.

Figure 5.6 Key Exchange Time with Various Key Lifetimes for 52-node Topology

Figure 5.7 Key Exchange Time with Various Key Lifetimes for 108-node Topology

Figure 5.8 Key Exchange Time with Various Key Lifetimes for 150-node Topology

In this chapter, different simulation scenarios have been considered to evaluate the newly proposed aggregation key exchange approach. The analysis of the simulation results illustrates that a less frequent control information exchange will improve the performance, especially for larger network. Unfortunately, this is another example of the trade-off between scalability and security: the longer a key is used, the less secure the communication becomes.

The new aggregation approach will improve the performance of the VPN system significantly against the traditional exchange solution, as illustrated in Figure 5.1 and 5.2. On top of that, a reasonable increase in node connectivity will also improve the performance, especially for large networks.

Chapter 6 Conclusions and Future Work

6.1 Summary of Research Work

The primary objective of the present research is to study the performance and scalability of key negotiation and exchange protocol. A new aggregation key negotiation and exchange approach is proposed to improve the VPN scalability. A simulation model for this new approach has been designed, evaluated and analyzed. Overall the research can be summarized as follows:

- Overview of VPN systems with latest technologies, which provide a simple solution to transfer sensitive data over unsecured public network. The LAN-to-LAN (extranet) VPN model is used for this research.
- Overview of the IPSec suite protocol: Authentication Header (AH), Encapsulating Security Payload (ESP), Internet Key Exchange (IKE). IKE is the protocol which is responsible for negotiating and exchanging the keys. Key exchange is completed in two steps: main mode and quick mode exchange. After the key negotiation and exchange is completed, all the encryption algorithms associated with that key are determined, and the two communicating parties are keeping them for encrypting and decrypting the data.

- Study of the Network Simulator. *NS* is the simulation tool which provides the environment to develop the protocol model and evaluate the performance of this protocol. It also provides a topology generator tool to create the different random graphs.
- Developed the IKE agent in *NS*, which perform the key negotiation and exchange.
- Generated six network topologies with various combinations of size and connectivity levels for to evaluate the new aggregation key exchange approach.
- Addressed the two main VPN deploying models. Analyzed the scalability implications of these virtual private network mechanisms. Proposed a new aggregation key negotiation and exchange approach, which provides a more scalable and efficient solution for key management in VPN system.
- Design the simulation scenarios to evaluate this new approach. Examined the impact of various factors that influence the key exchange performance, such as network size, connectivity level and frequency of exchange.
- Performed simulation experiments and analyzed the simulation results, which will result in the conclusion of this research.

6.2 Conclusion of Present Research

The main conclusions drawn from the studies undertaken in the present research are summarized as follows:

- The average time for key negotiation and exchange is affected by several main factors: the size and connectivity of the network, the number of users, the frequency of negotiation and exchange, and the number of service providers networks between sender and receiver.
- For the same connectivity level of the network topology, the average key exchange completion time in larger network (more nodes) is higher than in relatively smaller network (fewer nodes). This conclusion applies to both aggregation and traditional key exchange approaches.
- For the same connectivity level and network size, the performance of the newly proposed aggregation key exchange is improved by approximately 60% against the performance of traditional key exchange approach. The improvement is more important for larger networks and more users.
- For the same network size, the higher the connectivity level is, the lower the average time to complete the key negotiation and exchange, especially for larger networks. Therefore it is recommended to design larger networks that have a higher connectivity degree (i.e., more routes between each pair of core routers in the backbone).
- If the key exchange is more frequent, the performance diminishes. It will have more impact on the larger network than the relatively smaller ones. As shown

by the simulation results, when the frequency of exchange is reduced, the performance is improved dramatically for the larger network size. That is because larger networks accommodate many more users, which leads to more control information that has to be exchanged than in smaller size networks.

6.3 Future Work

The following are some recommendations for future work:

- The security issue may need to be evaluated for this new aggregation key exchange approach since this approach provides a trade-off solution between scalability and security.
- The simulation scenarios assume key negotiation and exchange across the same service provider backbone. An extension to multiple service providers may be considered for the future.
- Exchange and negotiation of Group Key for multicast is another open issue that still needs to be standardized. (Currently IETF has a draft for the standard). It will be interesting to model the group key exchange scheme for IPSec-based VPNs, and to evaluate its scalability and performance. Multicast communication requires another level of hierarchical key exchange scheme, so we expect that the aggregation approach is suitable for it.

References

1. 3Com communication, "Private Use of Public Networks for Service Provider",
<http://www.3com.com/>
2. C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure", RFC2510.
3. ADTRAN Technology Guide, "Understanding Virtual Private Networking",
EN286A, September 2001.
4. Agilent Technologies, "Scalability at the Edge: When is enough enough?",
5988-7005EN, June 5, 2002
5. Alcatel, "Understanding the IPSec Protocol Suite", White paper, March 2000.
6. J. Banks, J. S. Carson, B. L. Nelson and D. M. Nicol, "Discrete-Event System
Simulation", Prentice Hall, 2001.
7. CERT Coordination Center, "Denial of service", Software Engineering Institute,
Carnegie Mellon, http://www.cert.org/tech_tips/denial_of_service.html#1.
8. J. D. Clercq, O. Paridaens, "Scalability Implications of Virtual Private
Networks", IEEE Communications Magazine, May 2002.
9. K. L. Calvert, M. B. Doar, E.W. Zegura. "Modeling Internet Topology", IEEE
communications Magazine, pages 160-163, June 1997.

10. R. Callon, M. Suzuki, B. Gleeson, A. Malis, K. Muthukrishnan, E. Rosen, C.Sargor, J. Yu, "A Framework for Provider Provisioned Virtual Private Networks", IETF, Network Working Group, Internet Draft,
<http://www.ietf.org/proceedings/01mar/I-D/ppvpn-framework-00.txt>, 2001.
11. S. Chokhani, W. Ford, "Internet Public Key Infrastructure Certificate Policy and Certification Practices Statement Framework, Internet Draft", September 1997,
<http://www.ietf.org/html.charters/pkix-charter.html>.
12. Cisco Systems, Inc., "Access VPNs and IP Security Protocol Tunneling Technology Overview", Cisco Secure VPN Client Solutions Guide, 2002.
13. Cisco Systems, Inc., "IPSec Tunnels", Cisco Secure VPN Client Solutions Guide, 2002.
14. Cisco Systems Inc. "White Paper: IPSec", 1998.
15. N. Doraswamy and D. Harkins. "IPsec, The New Security Standard for the Internet, Intranets, and Virtual Private Networks", Prentice Hall, 1999.
16. W. Diffie, M. Hellman, "New Directions in Cryptography" IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November 1976.
17. Enterasys Networks Inc., 2002, "Virtual Private Networks A Technology Overview", <http://www.enterasys.com/products/whitepapers/vpn/9011742.html>

18. K. Fall and V. Kannan, 2000, “*ns* Notes and Documentation”
<http://www.isi.edu/nsnam/ns/doc/index.html>
19. P. Ferguson and H. Geoff, 1998, “What is a VPN?”
<http://www.clark.net/pub/timw/vpn/Tech/vpn.pdf>
20. B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis, “A Framework for IP-Based Virtual Private Network”, *RFC 2764*, Feb. 2000.
21. D. Harkeins and N. Doraswamy, “A Secure, Scalable Multicast Key Management Protocol (MKMP)”, draft-ietf-smug-mkmp-00.txt, 2000.
22. D. Harkins, and D. Carrel, “The Internet Key Exchange (IKE)”, *RFC 2409*, Nov 1998.
23. K. Hamzeh *et. al*, “Point-to -Point Tunneling Protocol (PPTP)”, *RFC 2637*, July 1999
24. M. E. Hellman. “An Overview of Public Key Cryptography”, *IEEE Communications Magazine*, November 1978 – Vol 16, Number 6.
25. T. Hanks, D. Li, P. Farinacci, P. Traina, “Generic Routing Encapsulation (GRE)”, *RFC1701*, 1994.
26. H. Harney and C. Muckenhirn, “Group Key Management Protocol (GKMP) Architecture”, *RFC2094*.

27. Intel Inc., "Embedded Intel Architecture in Communications", White paper, <http://www.intel.com/platforms/applied/eiacomm/papers/vpn.htm>
28. ISI , USC School of Engineering, "UCB/LBNL Network Simulator: Topology generation", <http://www.isi.edu/nsnam/ns/ns-topogen.html>
29. S. Kent, R. Atkinson, "IP Authentication Header", RFC 2402, Nov 1998.
30. S. Kent, R. Atkinson, "IP Encapsulating Security Payload", RFC2406, 1998.
31. S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC2401, 1998.
32. D. Maughan, M. Schertler, M. Schneider, J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC2408, 1998.
33. National Institute of Standards and Technology, "Data Encryption Standard (DES)", US Department of Commerce, 1994.
34. H. Orman, "The OAKLEY Key Determination Protocol" RFC2412, 1998.
35. The Open Group, "Architecture for Public-Key Infrastructure (APKI)", May, 1997.
36. C. Perkins, "IP Encapsulation within IP", RFC2003, 1996.
37. R. Perlman, and Charlie Kaufman, "Analysis of the IPsec Key Exchange Standard", Institute of Electrical and Electronics Engineers, 2001. <http://www.computer.org/proceedings/wetice/1269/12690150abs.htm>

38. R. Perlman, and C. Kaufman, "Key Exchange in IPsec: Analysis of IKE", IEEE Internet Computing, Nov/Dec 2000.
39. E. Rosen, J. De Clercq, O. Paridaens, Y. T'Joens, C. Sargor, "Use of PE-PE IPsec in RFC2547 VPNs", IETF, Network Working Group, Internet Draft, <http://www.ieft.org/internet-drafts/draft-ieft-ppvnp-ipse-2547-03.txt>.
40. RSA Laboratories, "Diffie-Hellman Key-Agreement Standard", <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-3/index.html>
41. RSA Laboratories, "What is Diffie-Hellman", <http://www.rsasecurity.com/rsalabs/faq/3-6-1.html>
42. P. Smith, 1993, LUC Public-Key Encryption, A Secure Alternative to RSA, *Dr. Dobb's Journl*, Vol. 18, Issue 1, January 1993.
43. W. A. Simpson, "IKE/ISAKMP Considered Dangerous", Internet Draft, draft-simpson-danger-isakmp- 00.txt, April 1999.
44. W. Townsley, et al, "Layer Two Tunneling Protocol 'L2TP'", RFC2661.
45. A. Valencia, M. Littlewood and T. Kolar, "Cisco Layer Two Forwarding (Protocol) 'L2F'", RFC2341.
46. E. W. Zegura, K. L. Calvert, S. Bhattacharjee. "How to Model an Internetwork", Proc. IEEE infocom, pages 594-602, April 1996.