

**APSEC: APPLICATION PROGRAMMING INTERFACE FOR
SECURE TRANSMISSION OF DATA**

By

Ali Ammar

(2000-NUST-BIT-797)



A project report submitted in partial fulfillment of
the requirements for the degree of
Bachelors in Information Technology

In

**NUST Institute of Information Technology
National University of Sciences and Technology
Rawalpindi, Pakistan
(2004)**

ACKNOWLEDGMENTS

I am thankful to Almighty Allah for the successful completion of the project. I thank my parents for their excellent support not only during the course of this project, but also throughout my life for without them, all this would have been impossible. I am deeply beholden to my advisor Mr. Ejaz Ahmed for his continuous and valuable suggestions and guidance, especially for the provision of all kinds of facilities throughout my work. His ability of management and foresightedness trained me a lot of things which will be more helpful for me in my practical life.

I am also thankful to my co-advisor Mr. Zaheer Abbas Khan for his constant guidance and insightful comments.

I would like to sincerely thank my committee members, Mr Kashif Sharif and Mr. Ehsan Ahmad Chaudhary for their valuable suggestions and comments to improve the dissertation.

I am also obliged to all my friends who have extended their kind and co-operative support in providing vital information and keeping my morale high.

TABLE OF CONTENTS

Topic No.	Topic	Page No.
CHAPTER 1	INTRODUCTION	1-6
1.1	Network Threats	1
1.2	Security	2
1.2.1	Transit Security	2
1.2.1.1	Confidentiality and Security	3
1.2.2	Traffic Regulation	4
1.2.3	Security Tools	5
CHAPTER 2	IPSEC	7-18
2.1	Architecture	8
2.1.1	Transport Mode	8
2.1.2	Tunnel Mode	9
2.2	Encapsulating Security Payload	12
2.3	Authentication Header	14
2.4	Internet Key Exchange	15
2.5	Benefits	17
CHAPTER 3	REQUIREMENT ANALYSIS	19-22
3.1	Project Title	20
3.2	Project Statement	20
3.3	Functional Requirements	20
3.4	Non Functional Requirements	21
CHAPTER 4	ARCHITECTURE AND DESIGN	23-38
4.1	Architecture	23
4.2	Design	24
4.2.1	IPSec	24
4.2.2	AES Algorithm	28
4.2.3	Authentication Server	29
4.2.4	Chat Application	29

4.2.5	File Transfer Application	32
4.2.6	Graphical User Interface	33
4.3	Sequence Diagrams	34
4.4	Class Diagrams	36
4.4.1	Association Between Classes At Sender	36
4.4.2	Association For Receiver	37
4.4.3	Association For Database Updation	37
CHAPTER 5	IMPLEMENTATION SPECIFIC DETAILS	39-46
5.1	IPSec	39
5.2	Chat Application	42
5.3	File Transfer Application	44
5.4	Database Updation	45
5.5	AES Algorithm	45
5.6	Constraints and Limitations	45
5.7	Technologies Used	46
CHAPTER 6	PERFORMANCE EVALUATION	47-49
6.1	Performance of Chat Application	47
6.2	Performance of File Transfer Application	48
CHAPTER 7	CONCLUSION AND RECOMENDATIONS	50-51
7.1	Conclusion	50
7.2	Recommendations	50
	REFERENCES	52-53
	APPENDICES	54-71
	Appendix A	54
	Appendix B	62
	Appendix C	66

LIST OF ABBREVIATIONS

IPSec	Internet protocol security
AH	Authentication Header
ESP	Encapsulating Security Payload
IKE	Internet Key Exchange
LAN	Local Area Network
DES	Data Encryption Standard
AES	Advanced Encryption Standard
MD-5	Message Digest Algorithm # 5
SHA	Secure hash Algorithm
FTP	File Transfer Protocol
UDP	User Datagram Protocol
TCP	Transmission Control protocol
GUI	Graphical User Interface

LIST OF FIGURES

Figure No.	Topic	Page No.
2.1	IPSec Transport and Tunnel Mode	9
2.2	ESP Protected IP Packet	13
2.3	AH Protected IP Packet	15
4.1	Architecture diagram	23
4.2	IPSec flow for TCP on sender side	25
4.3	IPSec flow for TCP on receiver side	26
4.4	Flow of IPSec for UDP on the sender side	27
4.5	IPSec flow for UDP at the receiver side	28
4.6	Flow of chat application for login and message sending	30
4.7	Flow of chat application for logout	31
4.8	Flow of File transfer application	32
4.9	Sequence diagram for file transfer application	34
4.10	Sequence diagram for chat application	35
4.11	Associations for sender	36
4.12	Associations for receiver	37
4.13	Associations for database updation	38
6.1	Thread view of chat application	47
6.2	CPU view of chat application	48

ABSTRACT

Networks today are a widespread information infrastructure, a mechanism for information dissemination, and a medium for collaboration and interaction between individuals, government agencies, financial institutions, academic circles and businesses of all sizes, without regard for geographic location. But at the same time they are very much unsecured. Authentication, message integrity and encryption are required for the security of information. . Therefore, a robust and flexible system for secure data transfer is required. The API APSec is designed specifically for solving security problem of the information moving over the networks. It provides secure transmission of data over the network using IPSec with modern algorithms and an authentication server. Any application either UDP or TCP based can use this API.

INTRODUCTION

This chapter explains the basics of network security. The issues and the problems involved in the transmission of information over the network. What level of security is being provided and what is needed.

1.1 NETWORK THREATS

Networks are used to conduct transactions and communications among businesses, government agencies and individuals. There are basically two types of networks, Private networks and Public networks. Private networks are those which are within one organization or company like a LAN. Public network are those which are open to public access. The simple example of public network is Internet. Either private or public the networks badly need security for their resources and data. Some of the threats to networks include:

Viruses: Computer programs written by devious programmers and designed to replicate themselves and infect computers when triggered by a specific event.

Trojan horse programs: Delivery vehicles for destructive code, which appear to be harmless or useful software programs such as games.

Vandals: Software applications or applets that cause destruction.

Attacks: Including reconnaissance attacks (information-gathering activities to collect data that is later used to compromise networks); access attacks (which exploit network

vulnerabilities in order to gain entry to databases, or the corporate network); and denial-of-service attacks (which prevent access to part or all of a computer system).

Data Interception: Involves eavesdropping on communication or altering data packets being transmitted.

Social Engineering: Obtaining confidential network security information through non-technical means, such as posing as a technical support person and asking for people's passwords.

1.2 SECURITY

Network security is the protection of networks and their services from unauthorized modification, destruction, or disclosure, and provision of assurance that the network performs its critical functions correctly and there are no harmful side-effects. Network security also includes providing data integrity.

There are two basic types of network security which when combined can help guarantee that the right information is securely delivered to the right place,

- Transit security
- Traffic regulation

1.2.1 Transit Security

Methods used to secure data as it transits a network. To keep the data secure we need to take care about data confidentiality and integrity.

1.2.1.1 Confidentiality and security

Security of data is concerned with two main areas: maintaining the confidentiality and integrity of electronic data.

- Data has *confidentiality* when only those who have been authorized to access particular data are able to.
- Data has *integrity* when it is not modified during transit.

To ensure that only authorized persons or computers can access or modify data on a network, there must be a method for establishing a user's identity on the system, along with a means to verify the identity. This is where identification and authentication come in.

Although these two terms are closely interrelated they describe two separate functions:

- *Identification* refers to the process that occurs during initial login whereby a person provides some type of security token, typically a unique username or user ID, to identify that user on the system. In effect, identification is to tell the system, "This is who I am."
- *Authentication* is a verification process that requires the user to provide another token, typically a password known only to the user, to affirm that the identity is being assumed by its rightful owner. The user is essentially telling the system, "Here is some private information to prove that I am who I claim to be."

While other forms of authentication have been proposed, such as smart cards and biometrics (fingerprint or retinal scanners, for example), passwords continue to be used almost exclusively in today's networks.

Two general approaches for transit security are:

- Virtual Private networks
- Packet level encryption

1.2.2 Traffic Regulation

Methods which regulate what packets may transit the network. It is the most common form of network security on the Internet today. If a packet which may do something malicious to a remote host never gets there, the remote host will be unaffected. Traffic regulation provides this screen between hosts and remote sites. This typically happens at three basic areas of the network: routers, firewalls and hosts. Each provides similar service at different points in the network.

Router traffic regulation: Any traffic regulation that occurs on a router or terminal server (hosts whose primary purpose is to forward the packets of other hosts) and is based on packet characteristics

Firewall traffic regulation: Traffic regulation or filtering that is performed via application gateways or proxies.

Host traffic regulation: Traffic regulation that is performed at the destination of a packet. Hosts are playing a smaller and smaller role in traffic regulation with the advent of filtering routers and firewalls.

The two common methods for traffic regulation are:

- Filters and Access lists
- Address translation

1.2.3 Security Tools

Some of the tools which could be used for network security are:

- *Antivirus software packages:* These packages counter most virus threats if regularly updated and correctly maintained.
- *Secure network infrastructure:* Switches and routers have hardware and software features that support secure connectivity, perimeter security, intrusion protection, identity services, and security management.
- *Dedicated network security hardware and software:* Tools such as firewalls and intrusion detection systems provide protection for all areas of the network and enable secure connections.
- *Virtual private networks:* These networks provide access control and data encryption between two different computers on a network. This allows remote workers to connect to the network without the risk of a hacker intercepting data.
- *Identity services:* These services help to identify users and control their activities and transactions on the network. Services include passwords, digital certificates, and digital authentication keys.
- *Encryption:* Encryption ensures that messages cannot be intercepted or read by anyone other than the authorized recipient.

These tools can be used to avoid security breach. One can use a single tool or combination of these depending on the requirements of the system.

Most companies' host computers can be accessed by their employees whether in their offices over a private communications network, or from their homes through public networks.

As the communication is increasing day by day several relatively minor issues have changed status from low priority to extreme importance. Secure transfer of data is probably the most well known of these problems. It is the need of every person using data transfer over the network. When businesses send private information across the net, they place a high value on it getting to its destination intact and without being intercepted by someone other than the intended recipient. Individuals sending private communications obviously desire secure communications. Connecting a system to a network can open the system itself up to attacks. If a system is compromised, the risk of data loss is high.

It is therefore very important to transfer data securely over the network either private or public. The first main type of network security i.e., Transit security deals with secure transfer of data. One has to insure confidentiality and integrity of data while transferring it.

IPSEC

IPSec is a framework of open standards developed by the Internet Engineering Task Force (IETF). IPSec provides security for transmission of sensitive information over unprotected networks such as the Internet. IPSec acts at the network layer, protecting and authenticating IP packets between participating IPSec devices ("peers"). With IPSec, data can be transmitted across a public network without fear of observation, modification or spoofing.

IPSec provides security by securing IP and upper layer protocols. IPSec protects IP data grams by defining a method of specifying the traffic to protect, how that traffic is to be protected, and to whom the traffic is sent. IPSec can protect traffic between hosts, network gateways and between hosts and network gateways.

IPSec provides the following security services. These services are optional. In general, local security policy will dictate the use of one or more of these services:

Data Confidentiality: The IPSec sender can encrypt packets before transmitting them across a network.

Data Integrity: The IPSec receiver can authenticate packets sent by the IPSec sender to ensure that the data has not been altered during transmission.

Data Origin Authentication: The IPSec receiver can authenticate the source of the IPSec packets sent. This service is dependent upon the data integrity service.

Anti Replay: The IPSec receiver can detect and reject replayed packets.

These security services are provided by using one of the protocols, the Encapsulating Security Payload (ESP) or Authentication Header (AH). AH data origin authentication, data integrity and anti replay protection. ESP provides all that AH provides with addition to optional data confidentiality and limited traffic flow confidentiality.

Ultimate security provided by AH or ESP is dependent on the cryptographic algorithms. The security services that IPSec provides requires shared keys to perform authentication and/or confidentiality. The keys should be added manually or dynamically using the key management protocol called IKE -- the Internet Key Exchange.

2.1 ARCHITECTURE

The Architecture Document for IPSec, RFC2401 [2], defines the base architecture upon which all implementations are built. It defines the security services provided by IPSec, how and where they can be used, how packets are constructed and processed, and the interaction of IPSec processing with policy.

The IPSec protocols—AH and ESP—can be used to protect either an entire IP payload or the upper-layer protocols of an IP payload. This distinction is handled by considering two different "modes" of IPSec transport mode and the tunnel mode. Both IPSec protocols, AH and ESP, can operate in either transport mode or tunnel mode.

2.1.1 Transport Mode

Transport mode is used to protect upper-layer protocols. In transport mode, an IPSec header is inserted between the IP header and the upper-layer protocol header. Because of the method of construction, transport mode can only be used to protect packets where the communications endpoint is also the cryptographic endpoint.

2.1.2 Tunnel Mode

Tunnel mode is used to protect entire IP datagrams. In tunnel mode the entire IP packet to be protected is encapsulated in another IP datagram and an IPSec header is inserted between the outer and inner IP headers. Tunnel mode may be used in place of transport mode, and in addition, may be used by security gateways to provide security services on behalf of other networked entities (for example, a virtual private network). In this latter case, the communication endpoints are those specified in the inner header that's protected and the cryptographic endpoints are those of the outer IP header.

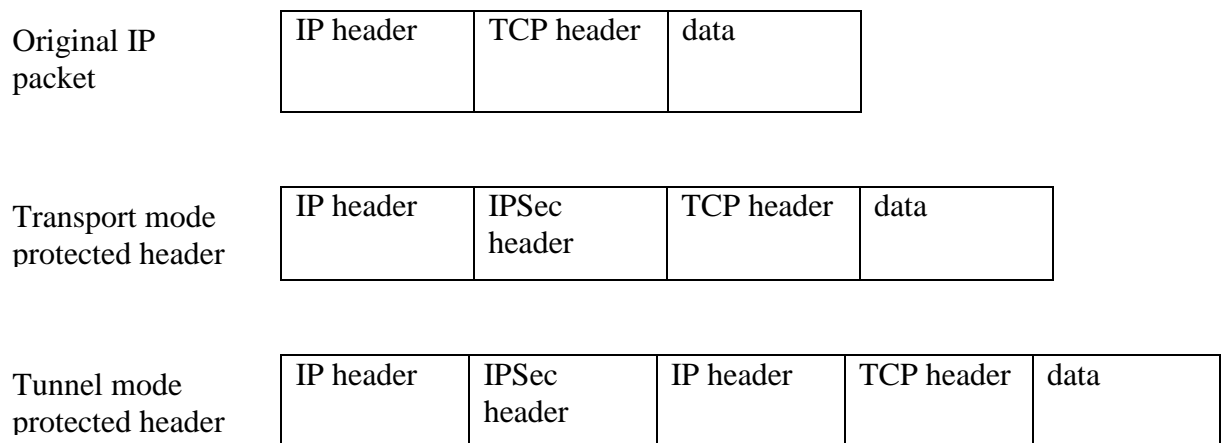


Figure 2.1: IP packets protected by IPSec in transport and tunnel mode

IPSec may be implemented in end systems or on security gateways such as routers and firewalls. Typically this is done by directly modifying the IP stack to support IPSec natively. When access to the IP stack of a machine is not possible, IPSec may be implemented as a "Bump in the Stack" (BITS) or "Bump in the Wire" (BITW). The former is typically a shim that extracts and inserts packets from the IP stack. The latter is typically an external, dedicated crypto device that may be independently addressable.

To properly encapsulate and decapsulate IPSec packets it is necessary to have a way to associate security services and a key. It is done through a "Security Association" (SA). An IPSec SA is unidirectional. It defines security services for either inbound packets or outbound packets. SA's are identified by a Security Parameter Index (SPI) which exists in IPSec protocol headers, the IPSec protocol value, and the destination address to which the SA applies which whether SA is inbound or outbound. Typically, SA's exist in pairs, one in each direction. They may be created manually or dynamically. SA's reside in the Security Association Database (SADB).

When created manually, an SA has no lifetime. It exists until it is manually deleted. When created dynamically, an SA may have a lifetime associated with it. A lifetime is important because the amount of traffic protected by a key must be carefully managed. Excessive use of a key can give an attacker an entry into your work.

The IPSec Architecture defines the granularity by which a user may specify his or her policy. This allows for certain traffic to be identified coarsely and have one level

of security applied while allowing other traffic to be identified more finely and have a completely different level of security applied.

IPSec policy is maintained in the Security Policy Database (SPD). Each entry of the SPD defines the traffic to be protected, how to protect it, and with whom the protection is shared. For each packet entering or leaving the IP stack, the SPD must be consulted for the possible application of security. An SPD entry may define one of three actions to take upon traffic match: discard, do not let this packer in or out; bypass, do not apply security services to an outbound packet and do not expect security on an inbound packet; and apply ,apply security services on outbound packets and require inbound packets to have security services applied. SPD entries that define an action of "apply" will point to an SA or bundle of SA's to be applied on the packet. IP traffic is mapped to IPSec policy by selectors. A selector identifies some component of traffic and may be either coarse or fine. IPSec selectors are: destination IP address; source IP address; name of the computer; upper-layer protocol; source and destination ports; and data sensitivity level (if an IPSec system also provides flow security). The values of these selectors may be specific entries, ranges, or "opaque." A selector in a policy specification may be opaque because that information may not be available to the system at that time.

If an SPD entry defines apply as an action and does not point to any existing SAs in the SADB, then in case of inbound traffic those SAs will have to be created before any traffic may pass. And in case of outbound traffic those packets must be dropped.

Both IPsec protocols, AH and ESP provide an antireplay service. IPsec packets are protected against replay attacks by using a sequence number and a sliding receive window. Each IPsec header contains a unique and monotonically increasing sequence number. When a SA is created, the sequence number is initialized to zero and prior to IPsec output processing, the value is incremented. New SAs must be created prior to the sequence number wrapping around back to zero — before 2^{32} packets since the sequence number is 32 bits long.

2.2 ENCAPSULATING SECURITY PAYLOAD

ESP is the IPsec protocol that provides confidentiality, data integrity, and data source authentication of IP packets, and also provides protection against replay attacks. It does so by inserting a new header, an ESP header, after an IP header (and any IP options) and before the data to be protected, either an upper-layer protocol or an entire IP datagram, and appending an ESP trailer. ESP is a new IP protocol and an ESP packet is identified by the protocol field of an IP header. If its value is 50 it's an ESP packet and immediately following the IP header is an ESP header. RFC2406 [4] defines ESP.

Since ESP provides both confidentiality and authentication, it has multiple algorithms defined in its SA, one for confidentiality and the other for authentication. Each ESP SA will have at most one cipher and one authenticate. It is possible to define NULL for both and do ESP without encryption or ESP without authentication, but it is illegal to have both null. This is illegal because not only it is a pointless burden on the system, it provides no security.

The ESP header is not encrypted but a portion of the ESP trailer is. Enough information is in clear text, to allow a recipient to process the packet. Since the SPI is used, along with the destination IP address of the IP header of this packet, to identify SA it must be in the clear i.e., not encrypted. In addition, the sequence number and authentication data are also in the clear. This is due to the specified order of processing of ESP packets: First verify the sequence number, then verify the integrity of the data, then decrypt the data. Since decryption is last, the sequence number and authentication data must be in the clear.

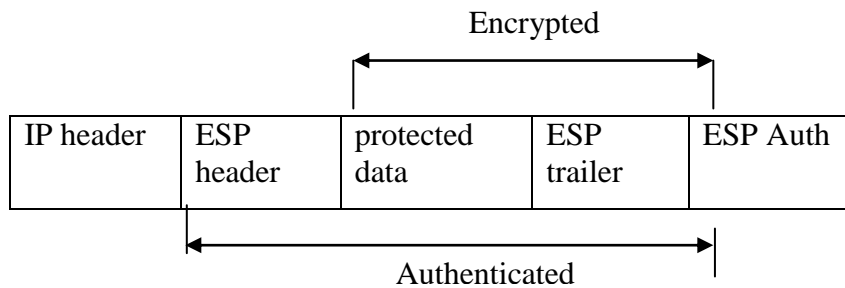


Figure 2.2: ESP protected IP packet

As mentioned, ESP contains both a header and trailer. The header portion contains the SPI and sequence number, the trailer contains the padding (if any), an indicator regarding the length of the pad, the next protocol of the data after ESP, and the authentication data. The size of the authentication data is dependent on the authentication algorithm used. Compliant implementations are required to support both HMAC-MD5 and HMAC-SHA with an output of 96 bits. These two MACs produce different-sized digests though. HMAC-MD5 produces a 128-bit digest while HMAC-

SHA produces a 160-bit digest. This is alright because the high-order 96 bits of the digest are used as ESP's authentication data. Ninety-six bits was chosen because it ensured alignment for IPv6.

2.3 AUTHENTICATION HEADER

Like ESP, AH provides data integrity, data source authentication, and protection against replay attacks. It does not provide confidentiality because of this the AH header is much simpler than ESP; it is merely a header and not a header plus trailer. In addition, all of the fields in the AH header are in the clear text.

The AH header, like the ESP header, contains an SPI to help locate the SA with which the packet is to be processed, a sequence number to provide against replay attacks, and an authentication data field to contain the digest from the keyed MAC used to secure the packet. Like ESP the length of the digest field is defined by the particular transform used. The default, mandatory-to-implement keyed MACs for AH are HMAC-MD5 and HMAC-SHA, both truncated to 96 bits. RFC2403 [6] for HMAC-MD5-96 and RFC2404 [7] for HMAC-SHA-96, used to define how to use these MACs with ESP. are used to define how to use them with AH.

Since AH does not provide confidentiality using a symmetric cipher in CBC mode, there is no explicit padding requirement imposed' on it. Some MACs may require padding, for example DES-CBC-MAC, but the technique of addition of the pad is left to the document describing the MAC itself.

The authentication coverage of AH differs from that of ESP. AH authenticates the outer IP header of the IPsec packet. Therefore various fields of the IPv4 and IPv6

headers that are mutable i.e., they may be changed by routers while the packet is in transit from source to destination must be zeroed prior to computation of the authentication data.

RFC2402 [3] defines the current incarnation of AH while RFC1826 described an older, deprecated version of AH. It defines the format of the AH header, where that header is placed in transport mode or tunnel mode, output data processing, input data processing, and other information such as handling fragmentation and reassembly of the packets.

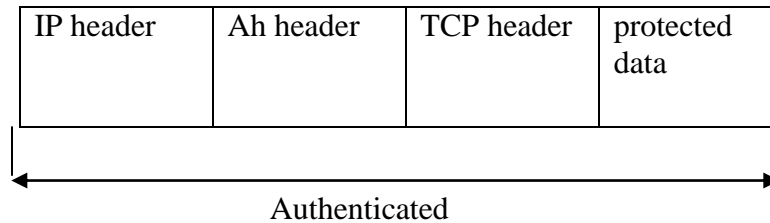


Figure 2.3: An AH protected IP header

2.4 INTERNET KEY EXCHANGE

Security associations are used with IPSec to define the processing done on a specific IP packet. An outbound packet produces a hit in the SPD and the SPD entry points to one or more SAs, a SA bundle. If there is no SA that instantiates the policy from the SPD it is necessary to create one. That is where the Internet Key Exchange (IKE) comes into play. The whole purpose of IKE is to establish shared security

parameters and authentication keys, in other words, security associations between IPSec peers.

The IKE protocol is a hybrid of the Oakley and SKEME protocols and operates inside a framework defined by ISAKMP, the Internet Security Association and Key Management Protocol. ISAKMP defines packet formats, retransmission timers, and message construction requirements, in effect, the language. Oakley and SKEME define the steps two peers must take to establish a shared, authenticated key. IKE uses the ISAKMP language to express these and other exchanges.

IKE is actually a general purpose security exchange protocol. It uses the concept of the security association but the physical construct of an IKE SA is different than an IPSec SA. The IKE SA is used to produce any number of IPSec SAs. Therefore the action that an IPSec implementation takes when an SPD entry has a null SADB pointer is to communicate the security requirements from SPD to IKE and instruct it to establish IPSec SAs.

The IKE protocol is performed by each party performing IPSec. The SPD of the IPSec is used to instruct IKE what to establish but not how to establish. How IKE establishes the IPSec SAs is based on its own policy settings

Creating IPSec SAs from IKE is a two phase process. Creation of the IKE SA is referred to as phase one. Once phase one is completed, phase two, creation of IPSec SAs may commence

The complete description of IKE can be found in three documents: the base ISAKMP specification RFC2408 [10], the Domain of Interpretation for IPSec RFC2407 [9], and the IKE specification itself RFC2409 [5].

2.5 BENEFITS

IPsec protects sensitive data that travels across unprotected networks and IPsec security services are provided at the network layer, so you do not have to configure individual workstations, PCs, or applications. This benefit can provide a great cost savings. Instead of providing the security services you do not need to deploy and coordinate security on a per-application, per-computer basis, you can simply change the network infrastructure to provide the needed security services.

Because IPsec is standards-based, many different devices (like routers of different vendors e.g., Cisco) will be able to interoperate with other IPsec-compliant networking devices to provide the IPsec security services. IPsec-compliant devices include many devices such as PCs, servers, and other computing systems.

A mobile user will be able to establish a secure connection back to his office. For example, the user can establish an IPsec "tunnel" with a corporate firewall requesting authentication services in order to gain access to the corporate network; all of the traffic between the user and the firewall will then be authenticated. The user can then establish an additional IPsec tunnel requesting data privacy services with an internal router or end system.

IPsec provides support for the Internet Key Exchange (IKE) protocol and for digital certificates. IKE provides negotiation services and key derivation services for IPsec. Digital certificates allow devices to be automatically authenticated to each other without the manual key exchanges.

IPSec due to its dynamic nature is preferred for use with medium-sized, large-sized, and growing networks, where secure connections between many devices are required.

REQUIREMENT ANALYSIS

Networks are open insecure medium. They have revolutionized the computing and communications world for the purpose of development and support of client and server services. The availability of the Networks, along with powerful affordable computing and communications, has made possible a new paradigm of commercial world. This has been tremendously accelerated by the adoption of browsers and World Wide Web technology, allowing users easy access to information. Networks have truly proven to be an essential vehicle of information trade today.

Networks today are a widespread information infrastructure, a mechanism for information dissemination, and a medium for collaboration and interaction between individuals, government agencies, financial institutions, academic circles and businesses of all sizes, without regard for geographic location.

People have become increasingly dependent on Networks for personal and professional use regardless of whether it is for e-mail, file transfer, remote login, Web page access or commercial transactions. With its increased awareness and popularity, Network security problems have been brought to the fore. Network security is not only extremely important, but more technically complex than in the past. The mere fact that business is being performed online over an insecure medium is enough to entice criminal activity to the Networks. The Internet access often creates a threat as a security flaw. To protect users from Internet based attacks and to provide adequate

solutions when security is imposed, cryptographic techniques must be employed to solve these problems.

This project was taken to solve the security problem over the networks. Authentication, message integrity and encryption are very important in cultivating, improving, and promoting network security. Without such authentication procedures, an attacker could impersonate anyone and then gain access to the network. Message integrity is required because data may be altered as it travels through the Internet. Without confidentiality, information may become truly public.

This project caters for all these three issues i.e., authentication, Message integrity and encryption.

3.1 PROJECT TITLE

Application Programming Interface (API) for secure transmission of data.

3.2 PROJECT STATEMENT

An Application programming interface that is used to provide authentication of users from an authentication server and confidentiality of exchanged information based on modern algorithms used in IPSEC.

3.3 FUNCTIONAL REQUIREMENTS

The following points clearly describe the functional requirements of the project:

- User should authenticate from an authentication server and only authorized users would be able to use the application. Thus an authentication server should be implemented.
- Every packet going out should be either authenticated or encrypted or both.
- Each outgoing and incoming packet should be checked for policy.
- Data base should be accordingly updated after each packet send.
- User input should be converted to Hexadecimal form before sending and on receiving that input should be converted back.
- Following algorithms would be used,
 - For authentication
 - HMAC-MD5-96
 - HMAC-SHA1-96
 - For encryption
 - DES-ECB
 - DES-CBC
 - AES

3.4 NON FUNCTIONAL REQUIREMENTS

The following points describe the non-Functional requirements of the project:

- It should be scalable i.e., It should be able to integrate in any application involving data transfer.
- The system development process and deliverable documents should authenticate each other.

- The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.
- WinPcap should be installed on each computer using this API.

APSec would be used to transfer data securely over the network using IPSec. Confidentiality and authentication of data would be provided by modern algorithms like DES, triple DES, AES, SHA-1 and MD-5. An authentication server will also be present for the authentication of the users using the application.

ARCHITECTURE AND DESIGN

This chapter describes architecture and design for the project. The architecture section describes the generic architecture when any application is using this API. And the design section describes the design of each module.

4.1 ARCHITECTURE

The following figure explains a general architecture when any application will be using the API provided.

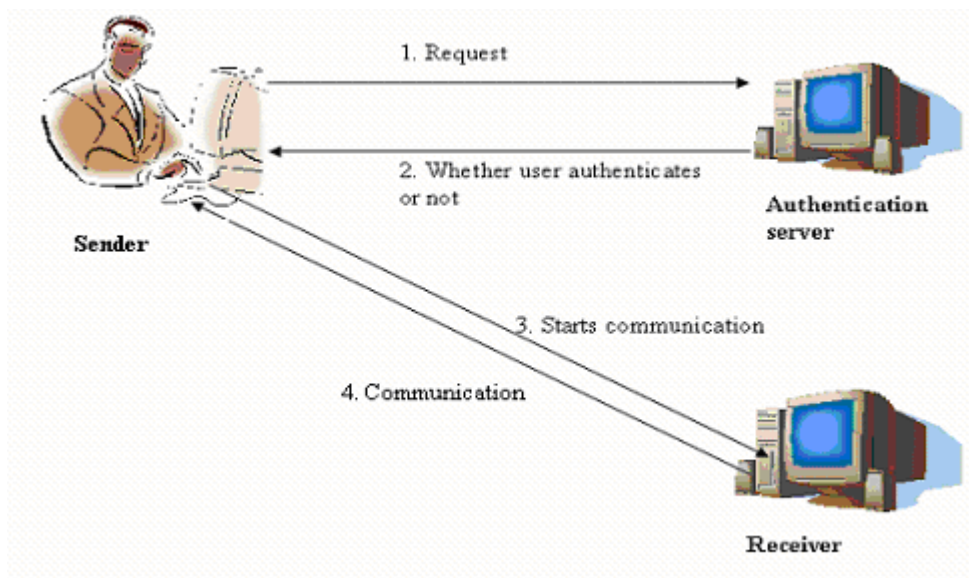


Figure 4.1: Generic architecture for any application using the API

Figure 1 above explains a generic architecture for any application using the API. Sender is the user who wants to communicate with the other user (Receiver)

using this API. The Sender first contacts the authentication server for authentication and after authentication contacts the receiver. Following series of steps take place:

- Sender first sends the request to authentication server for authentication.
- The authentication server authenticates the Sender on the basis of secret passed.
- The authentication server then sends the reply to the sender.
- If Sender is not authenticated he is not allowed to continue further.
- If Sender authenticates then he contacts the receiver to whom he want to transfer data or from whom he wants to receive data. Hence the communication starts.
- All the above communication i.e., the request for authentication, the authentication reply of server and communication after authentication is done using the API APSec.

4.2 DESIGN

The APSec is a very scalable API. The main modules which I included in this project are as follows:

4.2.1 IPSec

This module is the implementation of IP Security protocol defined by Internet Engineering Task Force. IPSec is implemented in two modes one for TCP based applications and the other for UDP based application. Whether the application is TCP based or UDP based it can use any version of IPSec. It depends on the application programmer.

Following figure describes the flow for TCP on the sender side.

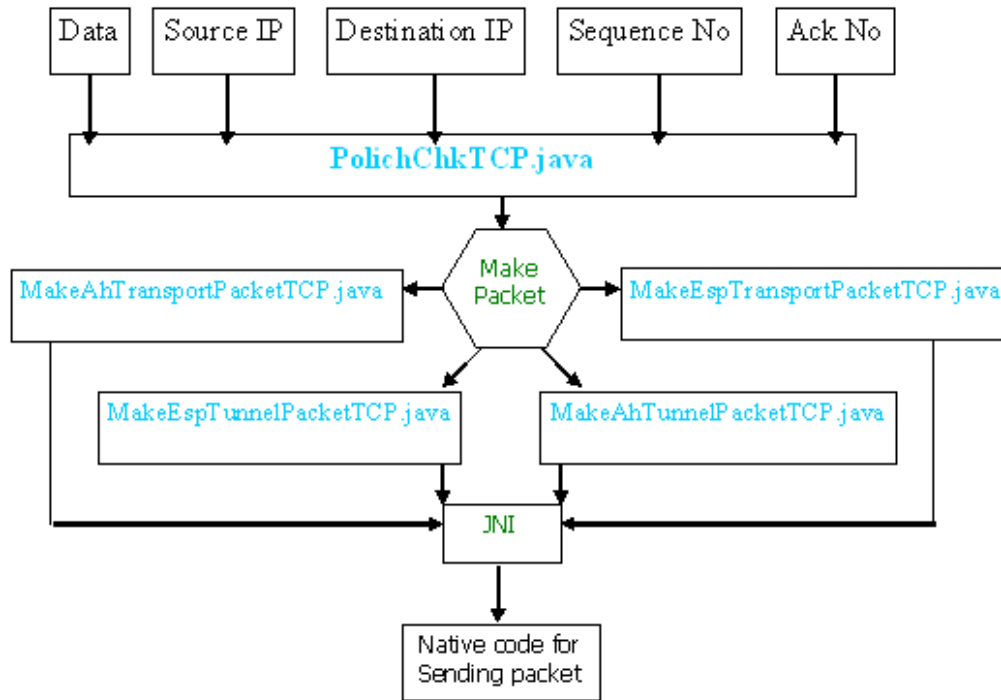


Figure 4.2: IPsec flow for TCP on sender side

On the sender side the input to this module is the data to be transferred, source IP, destination IP, sequence number and acknowledgement number. It calls one of the four classes shown in the figure 4.2 and makes packets appending header of each layer i.e., network and transport layer. It then adds the IPsec headers and the data. Then it performs specific operations on that packet depending upon the policy already defined to protect the packet. The policy defines whether the packet is protected through encryption or authentication or both. The packet when completely secured according to the policy is then transferred on the wire.

Following figure explains the flow of IPsec for TCP on the Receiver side.

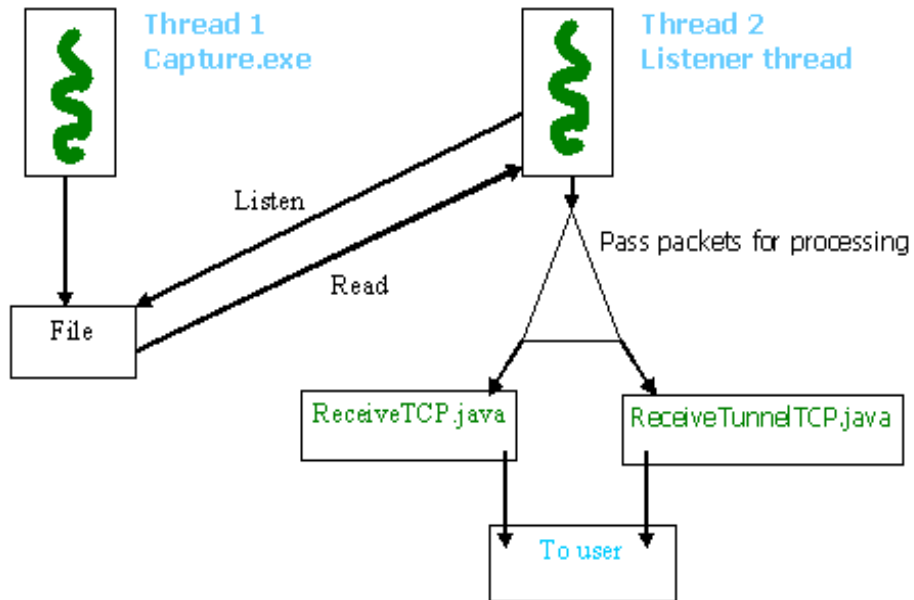


Figure 4.3: IPSec flow for TCP on receiver side

On the receiver side two threads are running. One captures the packets and writes to a file. The other thread gets the packets from that file and gives it to one of the two classes shown in the figure 4.3 to process. It then decodes the packets by removing headers one by one and performing operations defined in the policy. After the packet processing the data received is passed to the user.

The UDP version of the IPSec has the same flow of data but the working is different. The flow is same as the working of IPSec remains same whether it is TCP or UDP but the working is different as the processing for TCP header is different and UDP header is different. Following figure explains the flow of IPSec for UDP on the sender side.

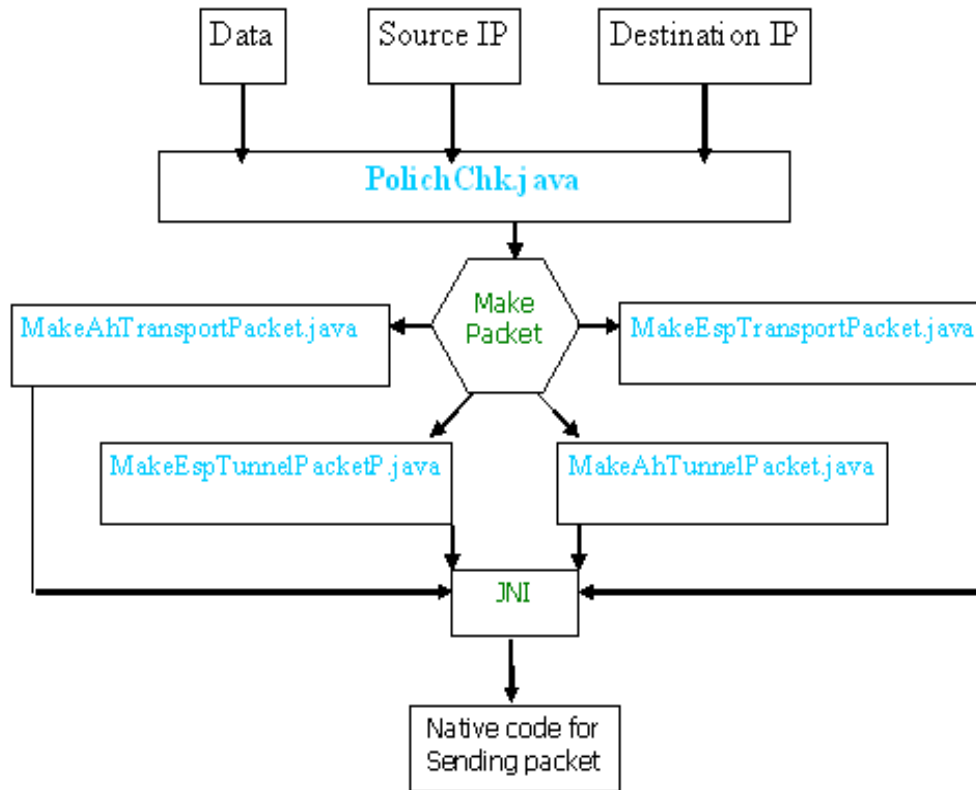


Figure 4.4: Flow of IPsec for UDP on the sender side

On the sender side the input to this module is the data to be transferred, source IP and destination IP. It calls one of the four classes shown in the figure 4.4 and makes packets. It performs specific operations on that packet depending upon the policy already defined to protect the packet. The policy defines whether the packet is protected through encryption or authentication or both. The packet when completely secured according to the policy is then transferred on the wire.

Following figure explains the flow of IPsec for UDP on the Receiver side.

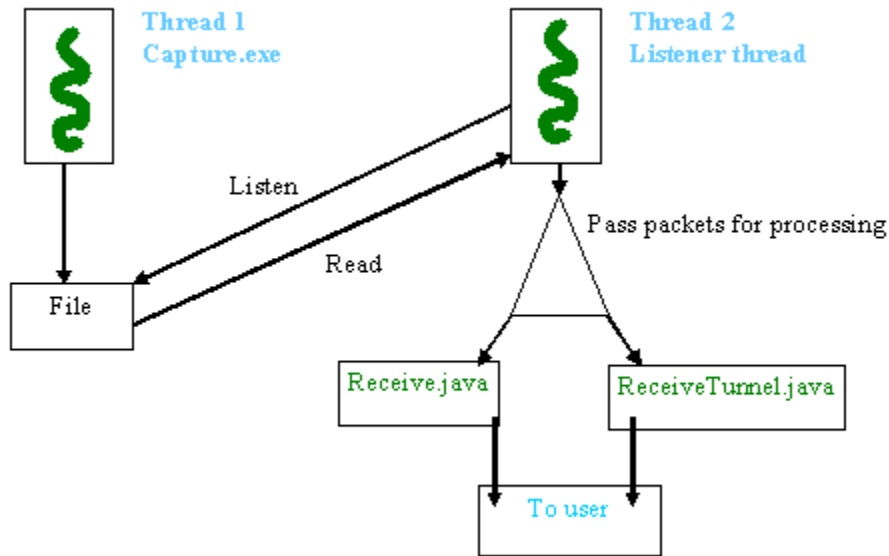


Figure 4.5: IPSec flow for UDP at the receiver side

Again on the receiver side two threads are running. One captures the packets and writes to a file. The other thread gets the packets from that file and gives it to one of the two classes shown in the figure 4.5 to process. It then decodes the packets by removing headers one by one and performing operations defined in the policy. After the packet processing the data received is passed to the user.

4.2.2 AES Algorithm

AES algorithm is used in IPsec to provide confidentiality of information through encryption. It takes 128 bits block as input and outputs 128 bits block. AES is a relatively new and much difficult to break algorithm that's why I have used it in IPsec. Other algorithms provided in this application which are:

For Authentication:

- HMAC- MD5-96
- HMAC-SHA1-96

For Encryption:

- DES-ECB
- DES-CBC

4.2.3 Authentication Server

Authentication server is used to authenticate the users for different applications. It is the part of the API not the application using this API. In this way any application using this API would not be dependent on any domain controller for authenticating its users. If the domain controller goes down the application would still be running. Applications can use either domain controller for authentication or the authentication server provided with the API. It in one way provides fault tolerance for the application using the API APSec.

4.2.4 Chat Application

Chat Application is used to demonstrate the use of API being provided along with the authentication server. The figure below describes the flow when user logs in and starts sending messages.

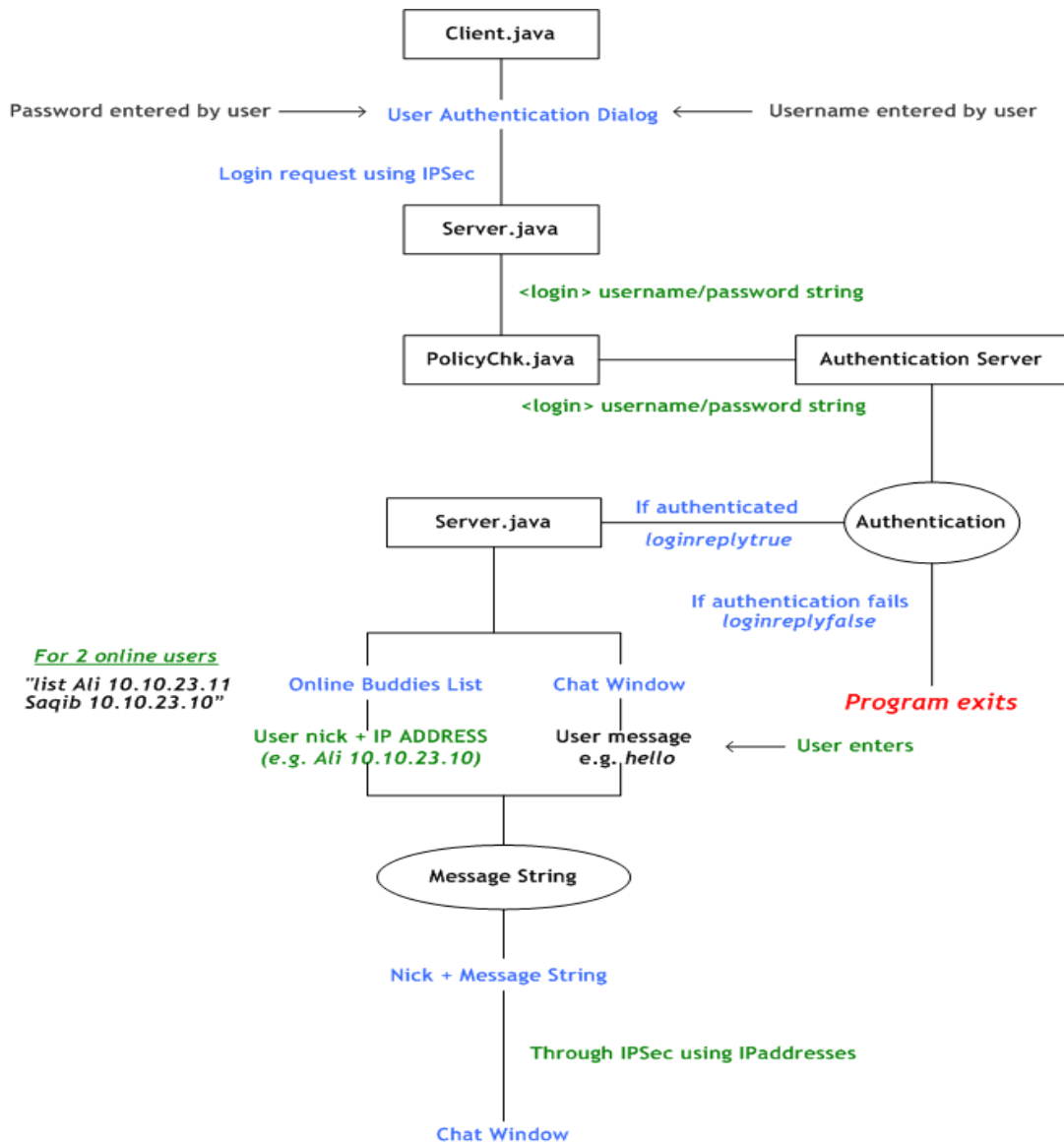


Figure 4.6: Flow of chat application for login and message sending

The user first logs in by authenticating through the authentication server. If the user does not authenticate the application exists. But if the user authenticates two windows appear to it one has the online buddies list and in other the user can chat. The authentication server maintains the list for all the users online at that time. It sends the list to all the users when a new user logs in or a user logs out. All the users in this way

get to know who else is online which is showed to them in the GUI. The user can then talk to each other by clicking at the name of the other user and sending message.

The following figure shows the flow when user logs out.

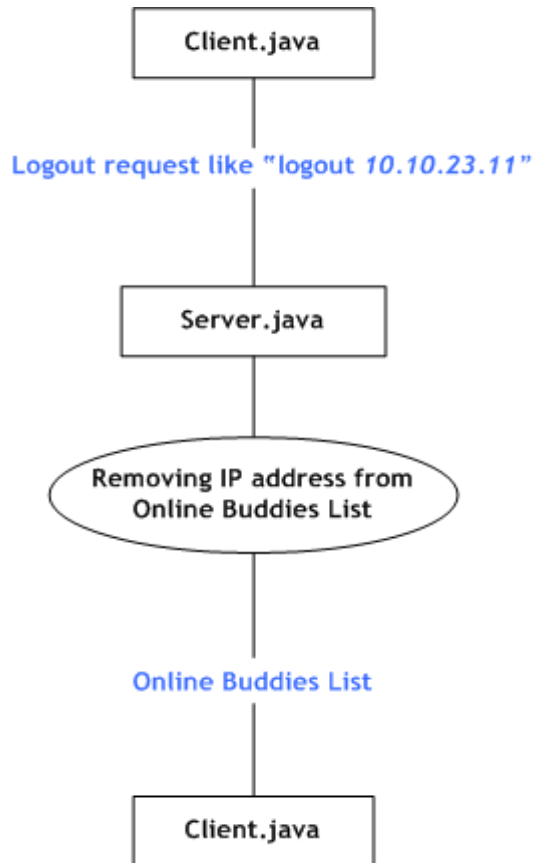


Figure 4.7: Flow of chat application for logout

The user when logs out sends a request to the server. Server first updates the list maintained with it and then sends that list to all the users online at that time.

All the communication, the request for login, the request for logout, the response of server, the sending of list by server and the chat between online users is done using the API APsec. Chat applications are basically UDP based so this application uses UDP version of IPsec. But it can also use the TCP version.

4.2.5 File Transfer Application

It is the second application which demonstrates the use of this API. . Following figure shows the flow for chat application

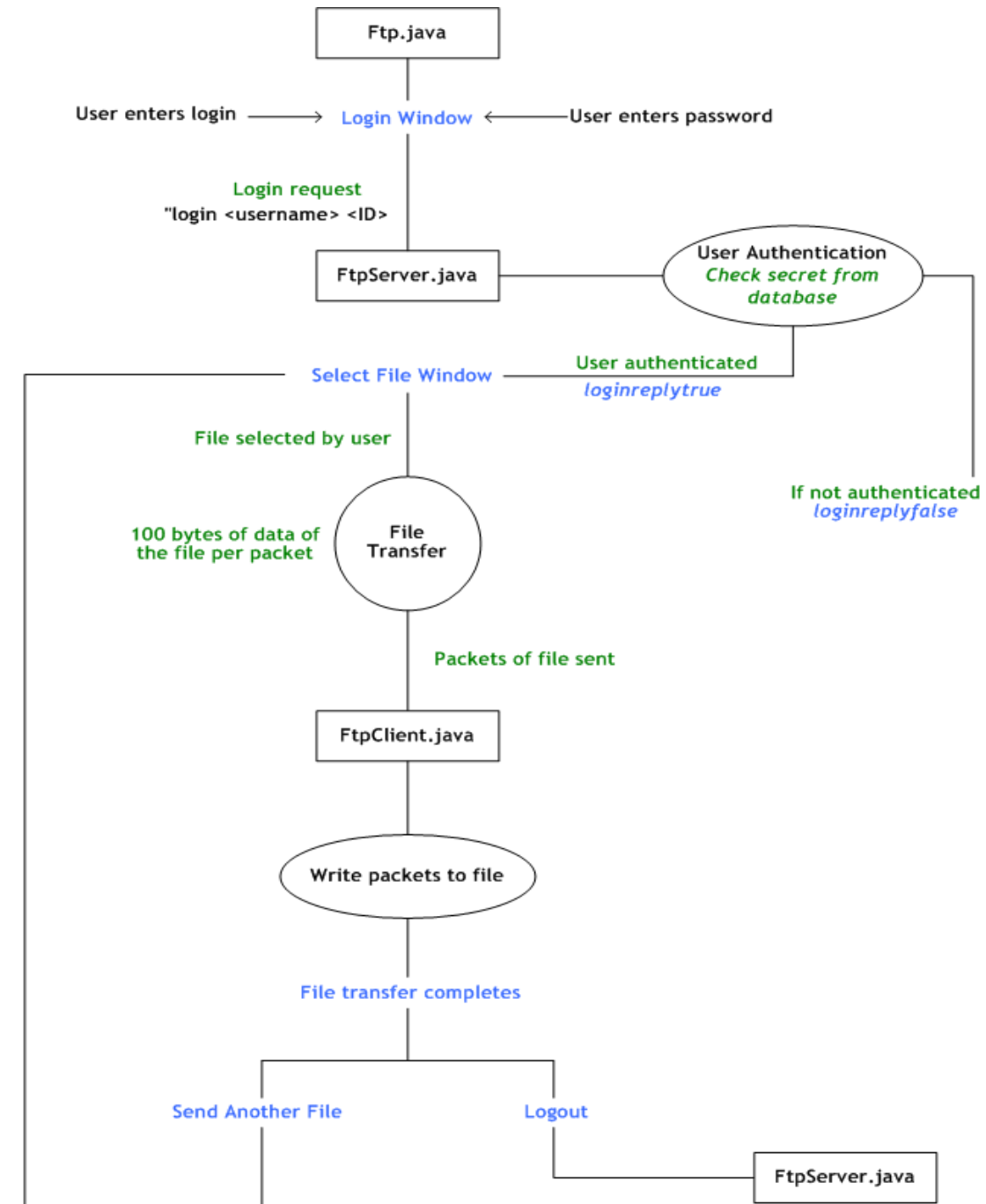


Figure 4.8: Flow of File transfer application

The user first authenticates from the authentication server and then he is allowed to transfer the file. The user then selects the file and sends it by pressing the send button. After the file is transferred the user can then logout after notifying the authentication server.

The authentication of user and the transfer of file, all is done using APSec. File transfer application works on TCP so it uses the TCP version of APSec. It can also use the UDP version; it depends upon the application developer.

4.2.6 Graphical User Interface

A graphical user interface is made to update Security Policy database and Security Association database for IPSec. This graphical user interface is made in order to support the manual SA generation. In this graphical user interface the user is first presented with a tab to enter the values for Security Policy database. The user first enters the values for Security Policy database and when these values are successfully updated in the database the user is shown the tab for Security Association database. He then enters the values of Security Association database. In this way one SA is created which is either for inbound or outbound traffic.

This GUI must be present on every machine using the API APSec because if there is a change in any database on any machine all the other machines could also be updated accordingly to ensure uniformity of databases at all machines.

4.3 SEQUENCE DIAGRAMS

The first sequence diagram is of the process when the user application is the file transfer application. The user here successfully sends a message to another user.

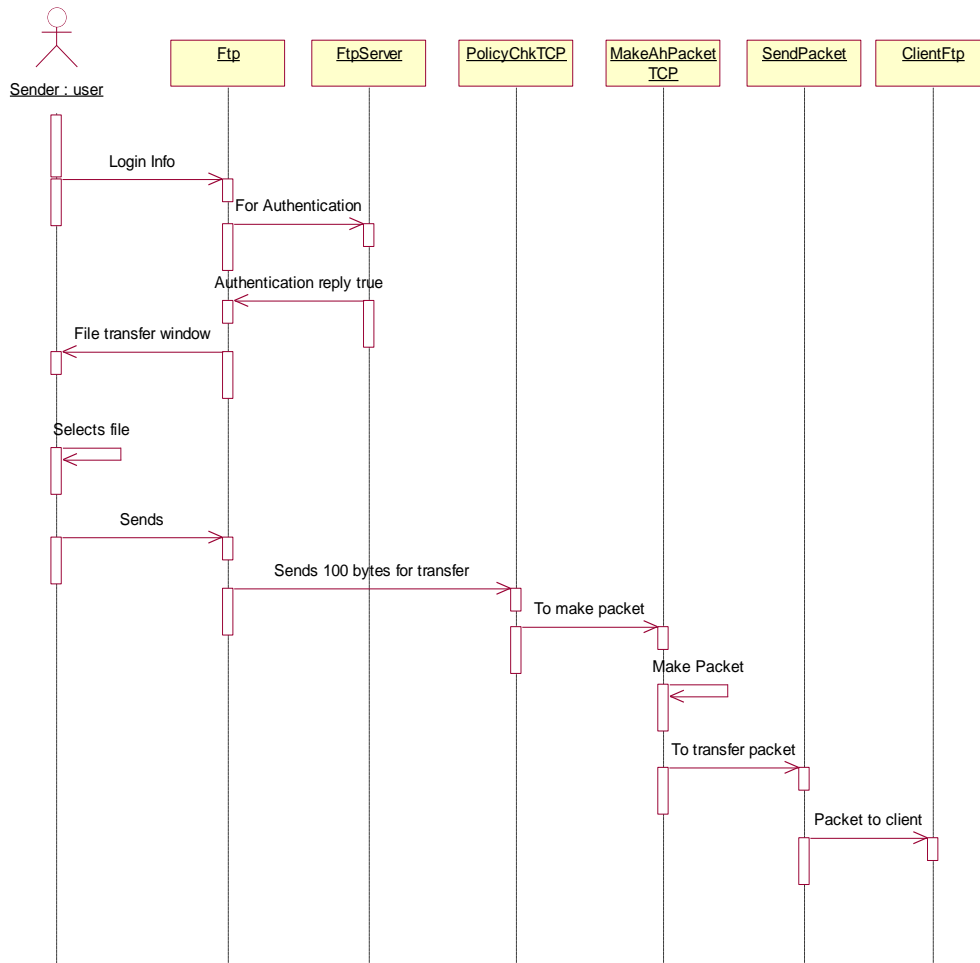


Figure 4.9: Sequence diagram for file transfer application

The second sequence diagram is of the process when the user application is the chat application. The user here successfully transfers a packet in the process of file transferring.

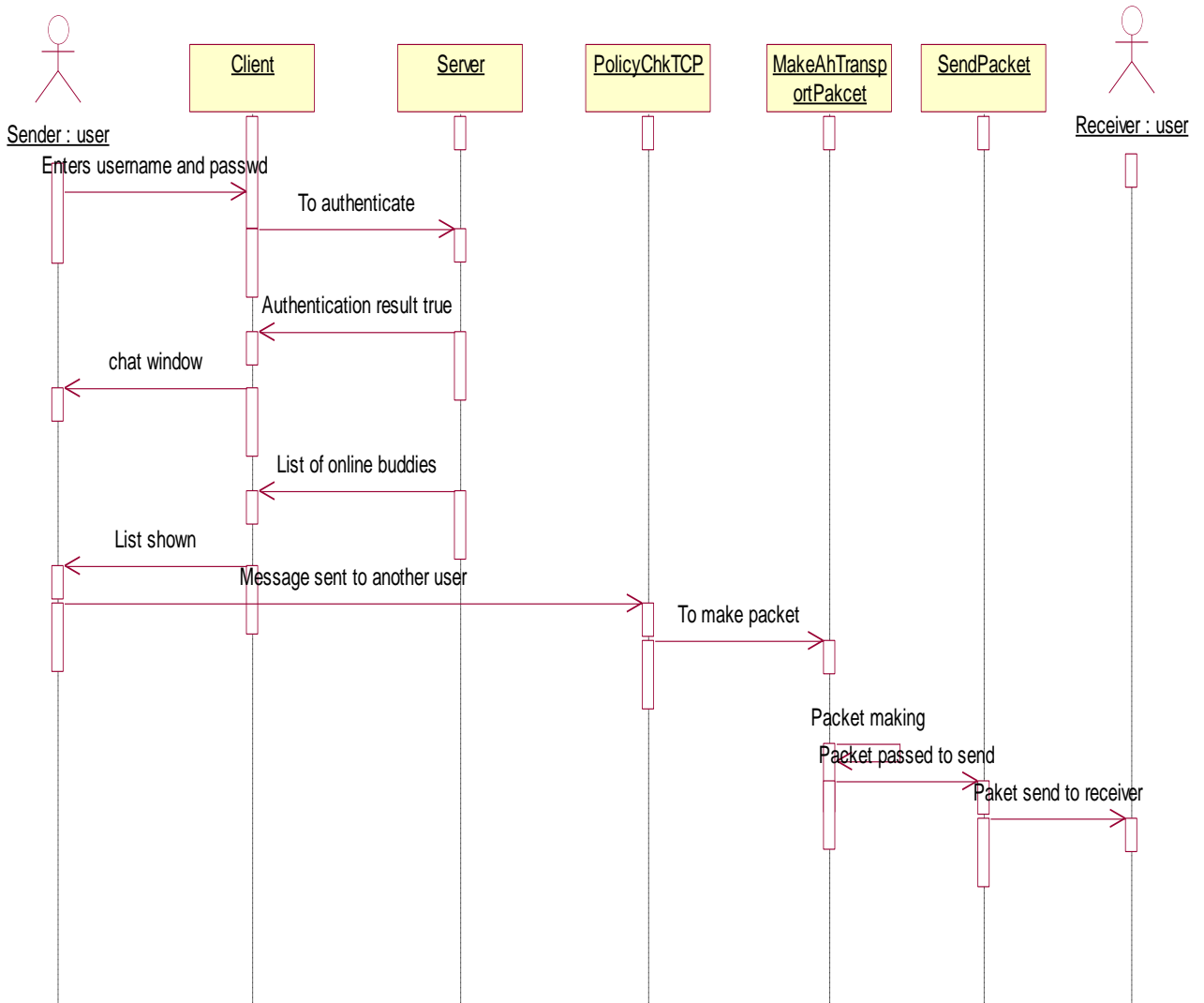


Figure 4.10: Sequence diagram for chat application

4.4 CLASS DIAGRAMS

4.4.1 Association Between Classes At Sender

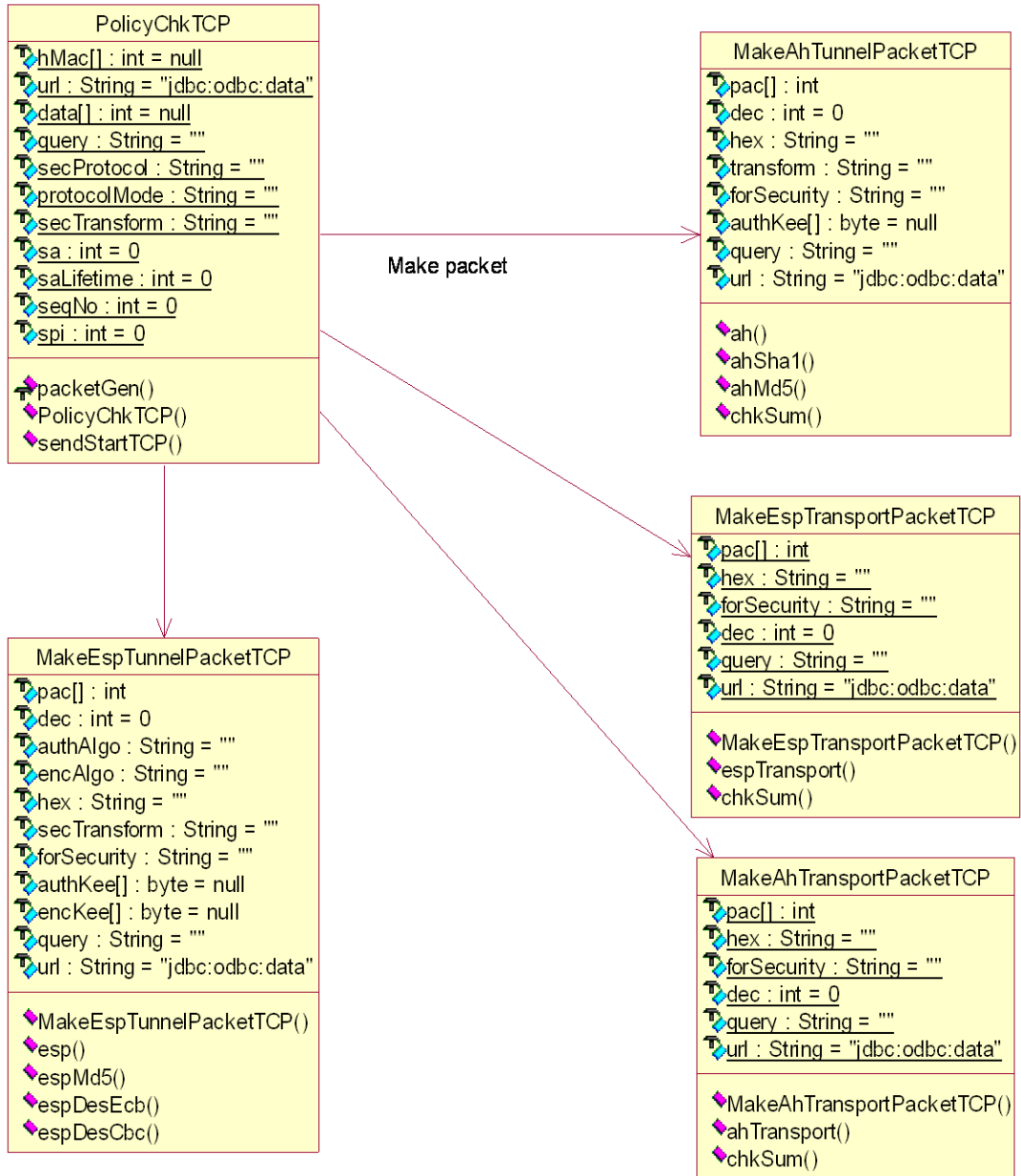


Figure 4.11: Associations for sender

4.4.2 Association For Receiver



Figure 4.12: Associations for receiver

4.4.3 Association For Database Update

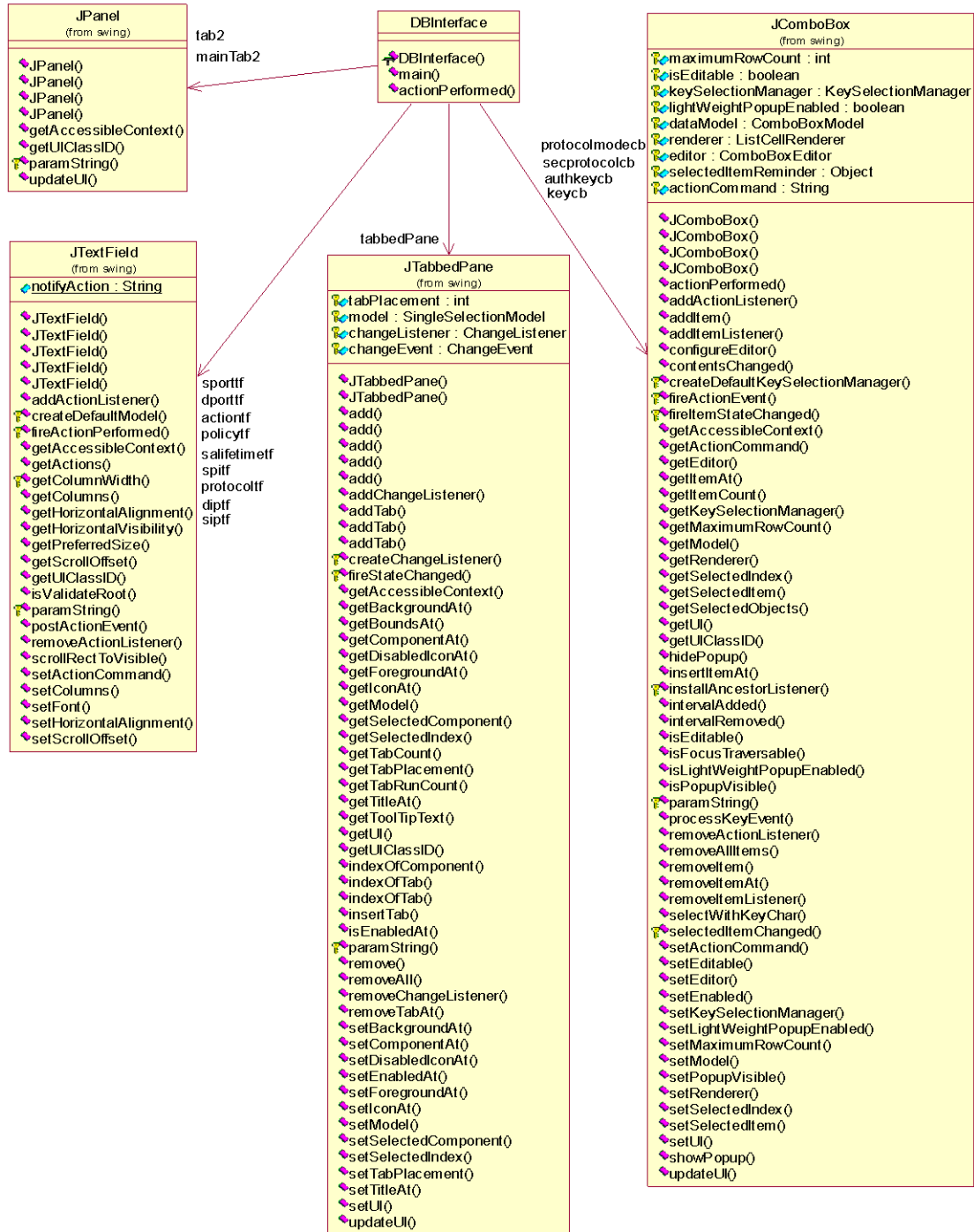


Figure 4.13: Associations for database update

IMPLEMENTATION SPECIFIC DETAILS

This chapter explains the implementation specific details that are a crucial part of the project. It tells the flow of data between different classes. Each section describes one specific module of the project.

5.1 IPSEC

IPSec is used to secure the information while transferring it over the network. IPSec works in two modes UDP and TCP. First the details of TCP mode are given.

The data, source IP address, destination IP address, sequence number and acknowledgement number are given from the sender of user application is given to the "*PolicyChkTCP.java*" class. The data is in the form of byte array, the source IP and destination IP are in the form of strings and the sequence and acknowledgement number are in the form of integers. This class then on the basis of source IP and destination IP gets the SPI for that particular traffic from this source to that destination. Then on the basis of SPI it checks whether to use Tunnel or Transport mode or this traffic and what header format to include either AH or ESP. After getting the policy confirmed a method of one of the following classes is called depending upon which algorithm to use.

- "*MakeAhTransportPacketTCP.java*"
- "*MakeAhTunnelPacketTCP.java*"
- "*MakeEspTransportPacketTCP.java*"

- "*MakeEspTunnelPacketTCP.java*"

MakeAhTransportPacketTCP.java is used to make the packet for transport mode using the authentication header. It has methods for authentication with MD5-96 and SHA1-96 algorithms.

MakeAhTunnelPacketTCP.java is used to make packet in tunnel mode using authentication header. It has methods for authentication with MD5-96 and SHA1-96 algorithms.

MakeEspTransportPacketTCP.java is used to make packet in transport mode using encapsulating security payload header. It has methods for authentication with MD5-96 and SHA1-96 and encryption with DES-CBC, DES-ECB and AES algorithms.

MakeEspTunnelPacketTCP.java is used to make packet in tunnel mode using encapsulating security payload header. It has methods for authentication with MD5-96 and SHA1-96 and encryption with DES-CBC, DES-ECB and AES algorithms.

All the methods of above four classes return integer array which contains the values of all the fields of all the headers along with the data thus making the complete packet. The *PolicyChkTCP.java* class then passes this array to the native method written in C language using WinPcap library to transfer the packet over the network. This is done through JNI.

On the receiver side we have two classes *ReceiverTCP.java* and *ReceiveTunnelTCP.java*. The receiver starts with the *ReceiverTCP.java* class. it has two threads running. One thread calls an executable capture.exe written in C language using WinPcap library to capture packets at the other end. This executable captures only those Packets which have IPsec headers which is done through making a filter

which checks that if the protocol field in IP header is 50 or 51 then capture the packet otherwise leave it. *Capture.exe* then writes the data and some specific values which are used in authentication or decryption process on a text file named "DataGotC.txt". For example in case of authentication header the SPI, hash calculated by sender and the fields on which the hash is calculated are written on the text file. All the values for one packet are written in one line, the values for second packet in second line and so on.

The other thread in the *ReceiverTCP.java* class is continuously listening on the text file and when the text file get a line it reads that line from the file and deletes it from there. Then as SPI being the first value in the text file is checked. And if the values correspond to the transport mode then these values are passed to the *chakHash()* method of the *ReceiverTCP.java* class but if the values correspond to the tunnel mode then these are passed to the *ReceiveTunnelTCP.java* class for processing. After the processing the actual data received, which is in the form of byte array is written on another text file named "*ClientDataGot.txt*". The receiver of the application can then get the data from that file and process it.

For UDP *PolicyChk.java* takes only data, source IP address and destination IP address. It then checks the SPI on the basis of source IP and destination IP and calls the method of one of the following class.

- "*MakeAhTransportPacket.java*"
- "*MakeAhTunnelPacket.java*"
- "*MakeEspTransportPacket.java*"
- "*MakeEspTunnelPacket.java*"

These classes work for the same purpose as the TCP classes do i.e., make packets but the procedure for making packets is different. While making packet here UDP header is used instead of TCP header. Here the header length is less due to less fields of UDP header, less fields are included in processing either in case of calculating hash or for encrypting and the processing is also different as the fields are different than those of TCP.

The packet then received in the form of an array is passed to the native method written in C using WinPcap. This method sends the packet on the network.

On the receiver side there are two receiver classes *Receiver.java* and *ReceiveTunnel.java*. One thread of *Receiver.java* starts the receiver by executing Capture.exe which captures the packets based on the filter and writes corresponding values to a text file "DataGot.txt" for processing. The other thread is listening on the file line by line and when it gets the data it checks from SPI whether it is transport mode packet or tunnel mode packet. If transport mode packet then it processes it itself otherwise passes it to the *ReceiveTunnel.java* class. After processing the data is written on the text file "*ClientDataGot.txt*" from where the receiver of application can get it.

5.2 CHAT APPLICATION

Chat application is made in order to demonstrate the use of the API APsec. It uses the UDP version of IPsec.

This application has two classes *Client.java* and *Server.java*. *Client.java* starts by opening a dialogue box for the authentication of the user. The user enters his user name and password and presses submit. The user name and password are added with a

tag login and send to the server using IPsec. The String *"login <username> <password>"* is passed to the server. Then server sends the reply of authentication. If the user authenticates the reply is *"loginReply true"* otherwise *"loginReply false"*. If the user does not authenticates programme exits. But once the user authenticates two windows appear to it after exiting the login window. One window shows the buddies online at that time and the user can chat in the other window. User gets the list of all the buddies' online form the server. This list contains the nick of the person and its IP address. Server sends this list to all the online buddies when a new user logs in or a user logs out.

Now if the user wants to chat with a friend he first selects the nick of the person in the list being shown to him in the online buddies' window. Then writes the message and sends. . The IP address of that buddy is selected from the list corresponding to the nick. After that a string is made which contains the message and the nick of the user sending the message and is directly send to the other user through IPsec by using the IP address.

e.g., if we have two users Ali and Saqib online then the list will be *"list Ali 10.10.23.11 Saqib 10.10.23.10"*. This list is given by server to both. Now if Ali selects Saqib to send a message "hello" to Saqib then the message string would be *"msg Ali hello"* and the message is sent to 10.10.23.10.

When the user wants to logout a string *"logout <IP address of the person to logout>"* is sent to the server. When a logout request is received by the server it removes the corresponding entry from the list and sends the new list to all the online buddies at that time.

Continuing the above example now if Ali wants to logout then the string would be *"logout 10.10.23.11"*. The server will remove the entry from the list on the basis of the IP address and send the list *"list Saqib 10.10.23.10"* to Saqib.

5.3 FILE TRANSFER APPLICATION

It the second application which demonstrates the use of this API. It works on the TCP version of IPSec.

File transfer application uses three classes *"Ftp.java"*, *"FtpClient.java"* and *"FtpServer.java"*. *FtpServer.java* does the authentication, *Ftp.java* is the class which sends the file and *FtpClient.java* is the class which receives the file.

Ftp.java is provided with a login window at the start. It sends the login request to the *FtpServer.java* and when the user authenticates another window appears through which the user can select a file and send it to the receiver. It sends 100 bytes of data of the file per packet and sends till the entire file has been sent. After the file is sent the user can logout or can select another file to send.

FtpServer.java is used to authenticate the user. It only gets the request of the user in the form *"login <username> <ID>"* and it checks the secret with the database. If the user authenticates *"loginReply true"* is sent to the user otherwise *"loginReply false"* is sent.

FtpClient.java receives the packets. It on receiving a packet writes it on the file thus starts making the file. When all the packets are received the file is completely written to a file in this way completing the file transfer.

5.4 DATABASE UPDATION

A single class “*DBInterface.java*” is used to update the security policy data base and security association data base. This class presents a GUI to the user in which only tab for SPD is shown and when user updates the SPD successfully another tab for SAD is gets displayed and the user then populates SAD.

5.5 AES ALGORITHM

AES algorithm is also implemented in a single class with methods for different operations. Its main methods are keyExpansion, Enciphering and Deciphering. The user inputs the key and the data to be encrypted. First of all keyExpansion method is called and the key is given to it in the from of hexadecimal string array. After expansion of the key Enciphering method is called and data to be encrypted is passed to it in the from of hexadecimal string array. Operations like shift rows, rotate words, mix columns and Xtime are performed to encrypt the data and the result is passed to the user again in hexadecimal string array.

On the other side the encrypted data is passed to the Deciphering method in hexadecimal string form. It performs the inverse of all the functions performed in the enciphering and generates the original data.

5.6 CONSTRAINTS AND LIMITATIONS

- This API is restricted to the number of algorithms it has unless more algorithms are added.
- Requires good processor as it’s a compute intensive API.

- Development constraints of JAVA would be transferred to the project.
- The system only handles ASCII files so it would work with text files, other formats when used would have garbage values as special characters which are not supported.
- WinPcap should be installed on the computers for using this API.

5.7 TECHNOLOGIES USED

- Java Language
- C language
- WinPcap

The IPSec module uses JNI to pass packets to C code written with the help of WinPcap, to transfer it over the network. API javax.crypto is used to incorporate algorithms other than AES. Chat application demonstrates the UDP version of IPSec and file transfer application demonstrates the TCP version of IPSec. And database updation is done manually at each machine with a GUI.

PERFORMANCE EVALUATION

6.1 PERFORMANCE OF CHAT APPLICATION

I did the performance analysis of the chat application by using the JProfiler Tool. JProfiler gives the information about memory views, CPU views, Thread views, virtual memory views. I have done the performance analysis on the basis of two information CPU views and Thread Views. The application when starts takes CPU utilization about 12% as shown in Fig 6.2 but when it reaches at a stable stage it becomes about 7 %. This application is very huge but not taking much CPU Utilizations. Reason being that most of the threads are in waiting state as shown in Fig 6.1, only that threads are in running state which are actually working. If any thread complete his work than that thread comes in waiting state and vice versa.

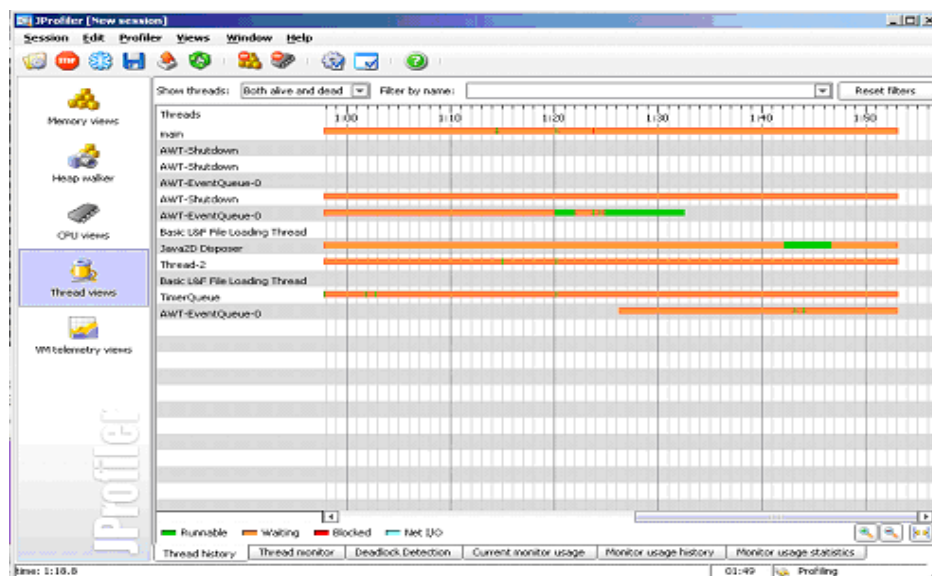


Figure 6.1: Thread view of chat application

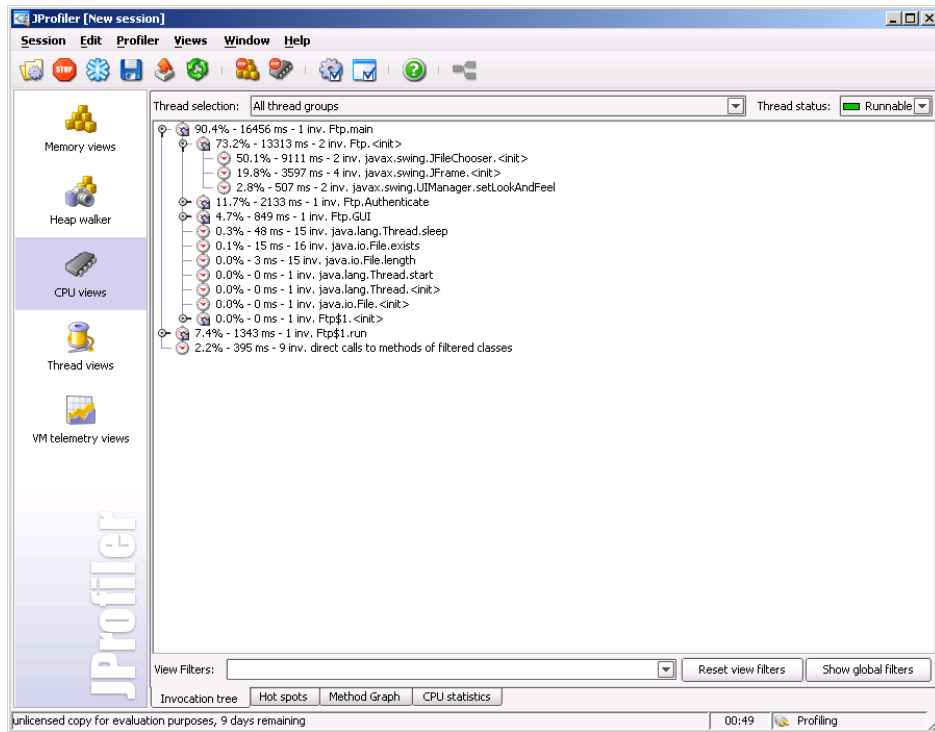


Figure 6.2: CPU view of chat application

6.2 PERFORMANCE OF FILE TRANSFER APPLICATION

I did the performance analysis of file transfer application by checking that how many files are completely transferred and how much time they take. Three types of text files were tested small files: files less than 100kb of size, medium files: files between 100 and 500kb of size and big files: files more than 500kb of size.

I transferred 20 small files and all of them were completely transferred with no delay. Thus showing 100 % accuracy in transferring small files.

Then I transferred 20 medium sized files. From 20 files 18 files were completely transferred but in two some packets were dropped. About two to three

packets were dropped in these files. Thus showing about 90 % accuracy for medium sized files. And there was very little delay in transferring the files.

For big files again 20 files were transferred. From these 20 files 17 were transferred completely. The other three files were also transferred but about four to five packets of each file were dropped. Thus showing about 85 % accuracy for big files. There was some delay in transferring big files.

The delays in transferring files depends upon two things, first the CPU power of the machine and second the policy for that traffic. If the processor of the machine is less than 1 GHz then there would be a delay as IPSec involves much processing. So it is recommended that machines with more than GHz processor should be used. And if the policy for that specific traffic involves authentication and encryption in tunnel mode then it would increase the delay as it is very compute intensive work.

CONCLUSION AND RECOMENDATIONS

7.1 CONCLUSIONS

This thesis report explains the implementation of API APsec which is used for secure transfer of data over the network using IPsec with modern algorithms and an authentication server. As the security threat to the information being transferred over the network is increasing day by day we need a mechanism for complete security of the information. IPsec is very secure mechanism through which data can be transferred over the network without being disclosed in the way. With this API now available the application programmers can provide security for all types of applications whether working on UDP or TCP. All the applications using APsec would firstly authenticate the user form the authentication server and then transfer data securely.

7.2 RECOMMENDATIONS

Improvements are always there in every type of project and as security is a big issue it cannot be completely handled by a single API. We need security over different forms of data and for that we need to further improve this API so that it is more efficient and effective. My recommendations are as follows:

- More encryption algorithms like RC 5, IDEA, and Triple IDEA etc should be added to reduce the probability of hacking information.

- More authentication algorithms like FNV, MD4 etc should be added to provide high level of authentication.
- Special characters should also be handled to provide security to those forms of data also.
- Efficiency should be increased in transferring big files while using the most intense computation of APSec.

REFERENCES

- [1] Java Cryptography Extension Reference Guide
<http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html>
- [2] Kent, S., and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.
- [3] Kent, S., and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.
- [4] Kent, S., and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.
- [5] Harkins, D., and D. Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998.
- [6] Madson, C., and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", RFC 2403, November 1998.
- [7] Madson, C., and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.
- [8] Madson, C., and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm with Explicit IV", RFC 2405, November 1998.
- [9] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.
- [10] Maughan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

- [11] NIST, FIPS PUB 197, "Advanced Encryption Standard (AES)," November 2001.
- [12] Naganand Doraswamy and Dan Harkins, IPsec - The New Security Standard for the Internet, Intranets and Virtual Private Networks. (Prentice Hall, 1999).

APPENDICES

Appendix A--- Code Snippets

The following is the code of *PolicyChkTCP.java* class which is the starting class of APSec API.

```
/**
 * @(#)PolicyTCP.java          1 04/08/31
 *
 * Copyright (c) 2000-2005
 * All rights reserved.
 *
 */

import java.net.*;
import java.sql.*;
import java.io.*;
import java.util.*;
import java.security.*;
import javax.crypto.*;

/**
 *
 * @version      1 31 Aug 2004
 * @author      ALi Ammar
 */

public class PolicyChkTCP
{
    static
    {
        //load the static libraray
        System.loadLibrary("packTCP");
    }

    static int hMac[]=null;
    static Connection conn=null;
    static Statement selectStatement=null;
    static Statement updateStatement=null;
    static ResultSet rs=null;
    static String url="jdbc:odbc:data";
    //static String sip="10.10.23.11",dip="10.10.23.11";
    static int[] data=null;

    static String query=""
    ,secProtocol="",protocolMode="",secTransform="";
    static int sa=0,saLifetime=0,seqNo=0,spi=0;
```

```

////////*****////////////////////////////////////////
static native void packetGen(int arr[]); //native method
////////*****////////////////////////////////////////

////////*****////////////////////////////////////////
////////*****////////////////////////////////////////
////////****Method for sending TCP packet****////////
public static void sendStartTCP(byte[] msg1,String sip,
String dip,int flagTcp,int seqTcp,int ackTcp)
{
    try
    {
        int[] data=new int[msg1.length];
        for(int i=0;i<msg1.length;i++)
        {
            data[i]=msg1[i];
        }

        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        conn=DriverManager.getConnection(url,"","");
        selectStatement=conn.createStatement();
        updateStatement=conn.createStatement();
        query="Select action from spd where sip='"+sip+"'
and dip='"+dip+"'";
        //select to check policy for this packet
        rs=selectStatement.executeQuery(query);
        /** If Policy found */
        if(rs.next())
        {
            /** checks if the policy has apply for the packet */
            if(rs.getString(1).equalsIgnoreCase("apply"))
            {
                query="Select sano from spd where sip='"+sip+"'
and dip='"+dip+"'";
                //select to check if policy has SA for this policy
                rs=selectStatement.executeQuery(query);
                /** If SA found */
                if(rs.next())
                {
                    sa=rs.getInt(1);
                    query="select secprotocol,
protocolmode,sectransform,
salifetime,spi,seqno
from sad where sano="+sa;
                    //select statement to get values from SAD
                    rs=selectStatement.executeQuery(query);
                    /** If all fields of SA found */
                    if(rs.next())
                    {
                        saLifetime=saLifetime-1;
                        seqNo=seqNo+1;

                    }

                }

            }

        }

        /** To check which IPsec protocol to implement */
    }
}

```

```

        if(secProtocol.equalsIgnoreCase("AH"))
        {
            if(protocolMode.equalsIgnoreCase("transport"))
            {
                packetGen(MakeAhTransportPacketTCP.ahTransport
                (sip,dip,data,spi,seqNo,sa,flagTcp,seqTcp,ackTcp));
            }
            else if(protocolMode.equalsIgnoreCase("tunnel"))
            {
                System.out.println("Ah header is used with tunnel
                mode for TCP.");
                MakeAhTunnelPacketTCP mpt=new MakeAhTunnelPacketTCP();
                packetGen(mpt.ah(sip,dip,data,spi,seqNo,sa,
                flagTcp,seqTcp,ackTcp)
                );
            }
        }
    else if(secProtocol.equalsIgnoreCase("ESP"))
    {
        if(protocolMode.equalsIgnoreCase("transport"))
        {
            System.out.println("Calling native method for
            ESP transport");
            packetGen(MakeEspTransportPacketTCP.espTransport(
            sip,dip,data,spi,seqNo,sa,flagTcp,seqTcp,ackTcp));
        }
        else if(protocolMode.equalsIgnoreCase("tunnel"))
        {
            System.out.println("Calling native method for ESP tunnel");

            MakeEspTunnelPacketTCP metp=new MakeEspTunnelPacketTCP();
            packetGen(metp.esp(sip,dip,data,spi,seqNo,sa,
            flagTcp,seqTcp,ackTcp));
        }

        /** IF SA not found */
        else
        {
            System.out.println("No SA found");
        }
        /** If policy not found */
        else
        {
            System.out.println("No policy found.");
        }
    }
}
catch(Exception e)
{
    System.out.println("Exception in PolicyChkTCP class: "+e);
}
}
/////*****
/////**End of Method for sending TCP packet**/////
}

```

This is the code of *MakeAhTransportPacketTCP.java* class which shows how a packet is made in transport mode using authentication header for TCP.

```

/**
 * @(#)MakeAhTransportPacketTCP.java          1 04/08/31
 *
 * Copyright (c) 2000-2005
 * All rights reserved.
 *
 */
import java.security.*;
import javax.crypto.*;
import java.net.*;
import java.sql.*;
import java.io.*;
import java.util.*;
import java.security.spec.*;
import javax.crypto.spec.*;

/**
 *
 * @version      1 31 Aug 2004
 * @author       ALi Ammar
 */

public class MakeAhTransportPacketTCP
{
    static int[] pac;
    static String hex="",forSecurity="";
    static int dec=0;
    static Connection conn1=null;
    static Statement selectStatement=null;
    static String query="";
    static String url="jdbc:odbc:data";

    //*****//
    //*****//
    //Method for AH header in transport mode//
    public static int[] ahTransport(String sip,String dip,int[] data,
    int spi,int seqNo,int sa,int flagTcp,int seqTcp,int ackTcp)
    {
        try
        {
            int[] tempPac=null;
            System.out.println("Inside ahTransport method");
            hex="";
            forSecurity="";
            dec=0;
            pac=new int[14+20+24+20+data.length];

            //MAC header//
            //dest mac//

```



```

pac[0]=255;
pac[1]=255;
pac[2]=255;
pac[3]=255;
pac[4]=255;
pac[5]=255;
/////src mac/////
pac[6]=00;
pac[7]=208;
pac[8]=183;
pac[9]=229;
pac[10]=92;
pac[11]=137;
//////////
pac[12]=8;
pac[13]=0;
/////IP header/////
//version +header Length
pac[14]=69;
//TOS
pac[15]=0;
/////Total length
hex=Integer.toHexString(20+28+20+data.length);
pac[16]=Integer.parseInt(hex.substring(0,2),16);
pac[17]=Integer.parseInt(hex.substring(2,4),16);

/////Identification/////
//hex=identification
hex=Integer.toHexString(60193);
pac[18]=Integer.parseInt(hex.substring(0,2),16);
pac[19]=Integer.parseInt(hex.substring(2,4),16);

//////Flags//////
pac[20]=64;
pac[21]=0;
/////Protocol/////
pac[23]=51;
/////H ChkSum/////
pac[24]=11;
pac[25]=11;
//////SIP//////
hex=sip;
dec=sip.indexOf(".");
pac[26]=Integer.parseInt(hex.substring(0,dec));
pac[27]=Integer.parseInt(hex.substring(
dec+1,sip.indexOf(".",dec+1)));
ec=sip.indexOf(".",dec+1);
ac[28]=Integer.parseInt(hex.substring(
dec+1,sip.indexOf(".",dec+1)));
dec=sip.indexOf(".",dec+1);
pac[29]=Integer.parseInt(hex.substring(dec+1,sip.length()));
/////DIP/////
hex=dip;
dec=dip.indexOf(".");

```

```

pac[30]=Integer.parseInt(hex.substring(0,dec));
pac[31]=Integer.parseInt(hex.substring(
dec+1,dip.indexOf(".",dec+1)));
dec=dip.indexOf(".",dec+1);
pac[32]=Integer.parseInt(hex.substring(
dec+1,dip.indexOf(".",dec+1)));
dec=dip.indexOf(".",dec+1);
pac[33]=Integer.parseInt(hex.substring(dec+1,dip.length()));

/** Setting Ah header */
/////Next header/////
pac[34]=6;
/////length/////
pac[35]=40;
/////reserved/////
pac[36]=0;
pac[37]=0;
/////SPI/////
hex=Integer.toHexString(spi);
pac[38]=0;
pac[39]=0;
pac[40]=Integer.parseInt(hex.substring(0,2),16);
pac[41]=Integer.parseInt(hex.substring(2,4),16);

/////Seq No/////
hex=Integer.toHexString(seqNo);
pac[42]=0;
pac[43]=0;
pac[44]=Integer.parseInt(hex.substring(0,2),16);
pac[45]=Integer.parseInt(hex.substring(2,4),16);

// Get instance of Mac object implementing HMAC-MD5, and
// initialize it with the above secret key
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
conn1=DriverManager.getConnection(url,"","");
selectStatement=conn1.createStatement();
query="select sectransform,authkee from sad where sano="+sa;
ResultSet b=selectStatement.executeQuery(query);
b.next();
String transform=b.getString(1);
byte[] fromDb=b.getBytes(2);
conn1.close();
///If1 that algo is MD5
if(transform.equalsIgnoreCase("MD5"))
{
    /////Putting 0 in place of hash
    for(int i=46;i<58;i++)
    {
        pac[i]=0;
    }
    /** Setyting TCP header */
    /////source port/////
    hex=Integer.toHexString(1377);
    pac[58]=Integer.parseInt(hex.substring(0,2),16);
}

```

```

pac[59]=Integer.parseInt(hex.substring(2,4),16);
/////destination port/////
pac[60]=Integer.parseInt(hex.substring(0,2),16);
pac[61]=Integer.parseInt(hex.substring(2,4),16);

/////Seq No/////
hex=Integer.toHexString(seqTcp);
pac[62]=0;
pac[63]=0;
pac[64]=Integer.parseInt(hex.substring(0,2),16);
pac[65]=Integer.parseInt(hex.substring(2,4),16);
/////Ack No/////
hex=Integer.toHexString(ackTcp);
pac[66]=0;
pac[67]=0;
pac[68]=Integer.parseInt(hex.substring(0,2),16);
pac[69]=Integer.parseInt(hex.substring(2,4),16);

/////HLEN, RES/////
pac[70]=20;
/////Flags/////
pac[71]=flagTcp;
/////Window size/////
hex=Integer.toHexString(64450);
pac[72]=Integer.parseInt(hex.substring(0,2),16);
pac[73]=Integer.parseInt(hex.substring(2,4),16);

/////Chksum/////
Checksum c=new Checksum();
Pac=c.chkSum(pac);
/////Urgent pointer/////
hex=Integer.toHexString(24);
pac[76]=Integer.parseInt(hex.substring(0,2),16);
pac[77]=Integer.parseInt(hex.substring(2,4),16);

/////data/////
for(int i=0;i<data.length;i++)
{
    pac[78+i]=data[i];
}
/////calculating and puttign hash/////
/////Creating hash and putiing in packet/////
forSecurity+=pac[14];
forSecurity+=" 0 ";
forSecurity+=20+28+20+data.length;
forSecurity+=" ";
forSecurity+=pac[18];
forSecurity+=pac[19];
forSecurity+=" 0 0 51 0 "+pac[26];
forSecurity+=pac[27];
forSecurity+=pac[28];
forSecurity+=" "+pac[30];
forSecurity+=" ";
forSecurity+=pac[35];

```

```

forSecurity+=" 0 ";
forSecurity+=pac[42];
forSecurity+=pac[43];
forSecurity+=pac[44];
forSecurity+=pac[45];
forSecurity+=" 0 ";
for(int i=58;i<78+data.length;i++)
{
    forSecurity+=pac[i];
    forSecurity+=" ";
}

/////write the keyobject as byte[] in another file/////
File f1=new File("c.txt");
FileOutputStream oo=new FileOutputStream(f1);
oo.write(fromDb);
oo.close();

/////Read the key object from another file as object/////
ObjectInputStream oi=new ObjectInputStream(new
FileInputStream(new File("c.txt")));
Object skk=oi.readObject();
oi.close();
SecretKey sk=(SecretKey)skk;
Mac mac = Mac.getInstance("HmacMD5");
mac.init(sk);
byte[] result1 = mac.doFinal(forSecurity.getBytes());
for(int i=0;i<(result1.length-4);i++)
{
    pac[46+i]=result1[i];
    System.out.println("HMAC: "+result1[i]);
}
/////end of if1 that algo is MD5
/////else if algo is SHA1
else if(transform.equalsIgnoreCase("SHA1"))
{
}/////End of else If when algo is SAH1
/////Else if anyother algorithm than MD5 or SHA1
else
{
}/////End of else when algo different than MD5 and SHA1

return pac;
}
catch(Exception e)
{
    System.out.println(e);
}
}
}

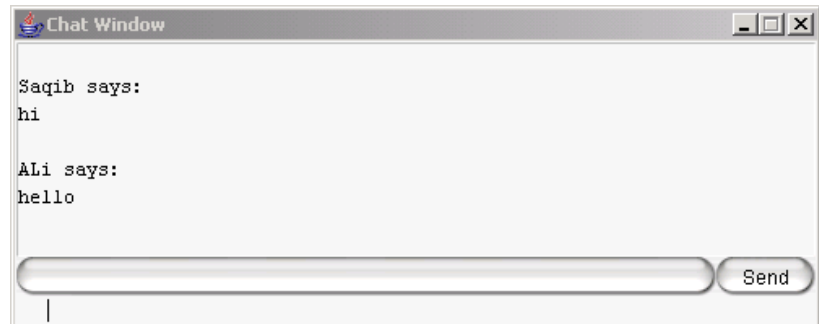
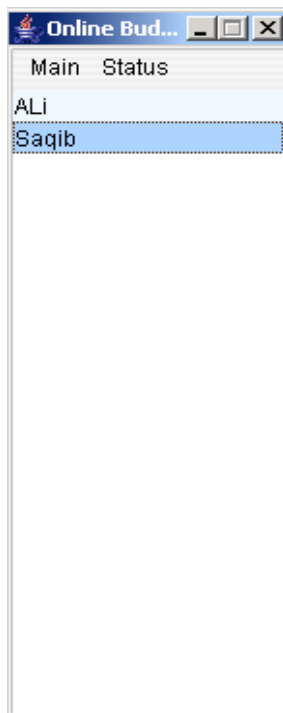
```

Appendix B--- Screen Shots

Chat Application

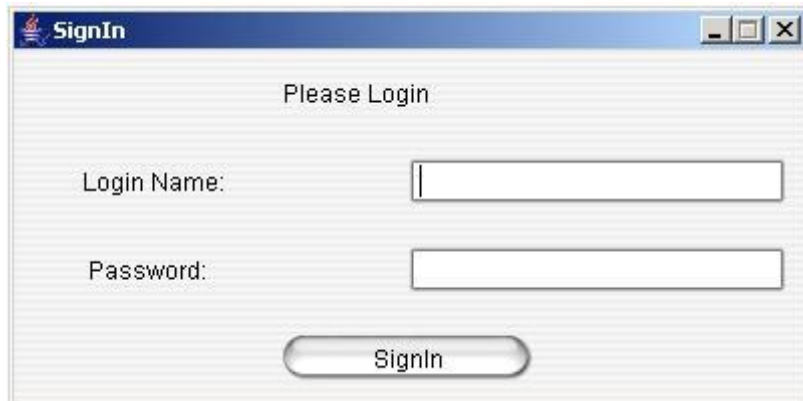


Sign in frame



Chat and online buddies windows

File Transfer Application



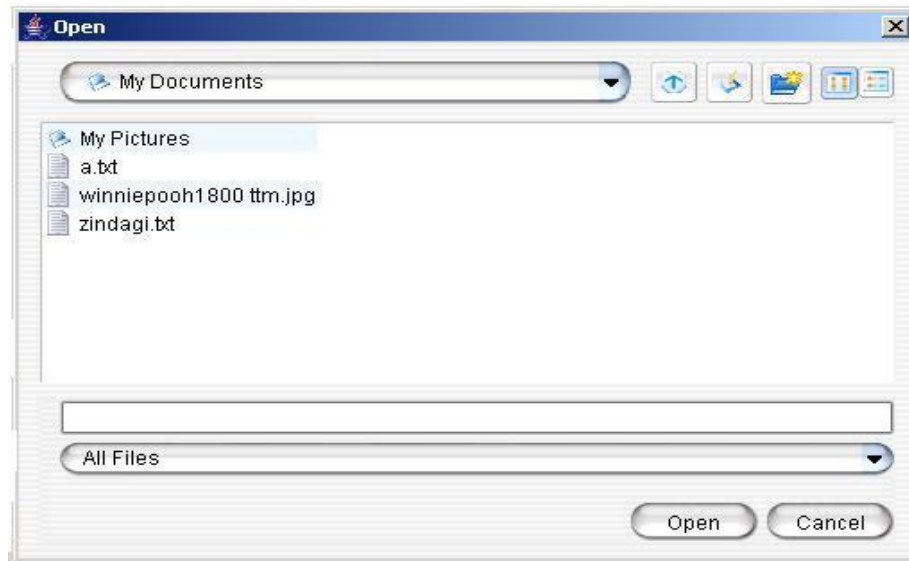
A dialog box titled "SignIn" with a blue header bar. The main content area is light gray and contains the text "Please Login" centered at the top. Below this, there are two input fields: "Login Name:" followed by a text box, and "Password:" followed by a text box. At the bottom center, there is a rounded button labeled "SignIn".

Sign in frame



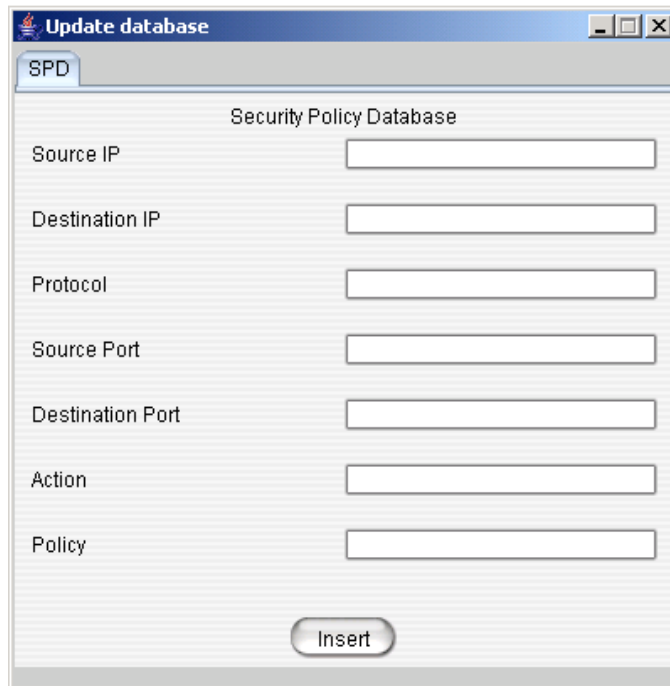
A window titled "File Transfer Application" with a blue header bar. The main content area is light gray and contains the text "File Transfer Application" centered at the top. Below this, there is a label "Select IP" followed by a dropdown menu showing "10.10.23.11". At the bottom, there are three rounded buttons: "Logout", "Browse", and "Send".

Window to select receiver, send file and logout



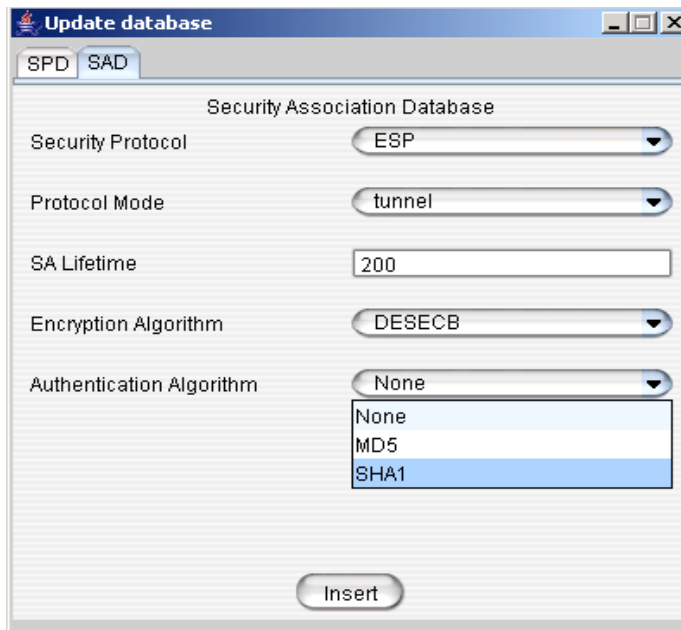
Window to select file

Database Updation



The screenshot shows a window titled "Update database" with a tab labeled "SPD". The main area is titled "Security Policy Database" and contains seven input fields: "Source IP", "Destination IP", "Protocol", "Source Port", "Destination Port", "Action", and "Policy". Each field is an empty text box. At the bottom center of the window is an "Insert" button.

Updating SPD



The screenshot shows the same "Update database" window, but with the "SAD" tab selected. The main area is titled "Security Association Database" and contains five input fields: "Security Protocol" (set to "ESP"), "Protocol Mode" (set to "tunnel"), "SA Lifetime" (set to "200"), "Encryption Algorithm" (set to "DESECB"), and "Authentication Algorithm" (set to "None"). The "Authentication Algorithm" dropdown menu is open, showing options: "None", "MD5", and "SHA1". At the bottom center is an "Insert" button.

Updating SAD

Appendix C--- Test Cases

AES encryption and decryption test

Identifier	AES	
Category	Unit Test	
Description	<p><i>Advanced encryption standard is a new algorithm made after triple DES. It is used in the API APsec to provide confidentiality of information.</i></p> <p><i>AES takes 128 bits block as input and gives 128 bits block as output. The key size is also 128 bits. It is not provided by java so I implemented to provide confidentiality of data using IPsec.</i></p> <p><i>This test case checks the problems while encrypting and decrypting data using AES.</i></p>	
Setup	<p>We want to encrypt and decrypt a string using the AES. On the system AES.java and CallAES.java should be present. The CallAES class should be running which would then ask for input.</p>	
Test Procedures	<ol style="list-style-type: none"> a. Run CallAES class b. It would then ask for 128 bits key and 128 bits string to be encrypted and then decrypted c. The key and the string should be in hexadecimal form d. The string after encryption would be given as input to the decryption method 	
Expect Results	On Success	<ul style="list-style-type: none"> • System will first print the encrypted form of the string given as input • System would then print the decrypted form of the input • Both the string given as input and the decrypted string would be same
	On Failure	<ul style="list-style-type: none"> • The string given as input for encryption and the decrypted string would not match • <i>Can throw following exceptions</i> <ul style="list-style-type: none"> ○ Illegal block size ○ Index Array out of bound ○ Class not found
Related Test Cases	None	
Notes	None	
Issues	None	

Sending packet test

Identifier	SendPacket	
Category	Unit Test	
Description	<p><i>Java does not provide raw socket support so I had to work in C to generate packets and send them on the network. Also for sending the packets I used WinPcap library for windows.</i></p> <p><i>The data is passed though Java to C language when the packet is made and sent. Data is passed through JNI and sent on network using WinPcap.</i></p> <p><i>This test case check whether the data passed to C through JNI is sent to the destination using WinpCap or not.</i></p>	
Setup	We want to send the packet form source to destination. Source and destination machine should fulfill following conditions.	
	Source	<ul style="list-style-type: none"> • WinPcap installed • Set visual studio environment • Include jni.h and the header file for native code in visual studio include field • Include winpcap.lib in library files
	Destination	<ul style="list-style-type: none"> • WinPcap installed • Iris installed
Test Procedures	<ol style="list-style-type: none"> a. Run Iris on the destination machine b. Run the Test.java class on sender which passes data to C code for sending 	
Expect Results	On Success	<ul style="list-style-type: none"> • Packet received on the destination machine and shown in Iris • The data in the packet is same as sent by the sender
	On Failure	<ul style="list-style-type: none"> • Packet would not be received on the destination machine • Can throw following exceptions on sender machine <ul style="list-style-type: none"> ○ Error opening adapter ○ RemoteException ○ LinkingError Exception ○ ClassNotFoundException ○ HeaderFileNotFound
Related Test Cases	ReceivePacket	
Notes	None	
Issues	None	

Receiving packet test

Identifier	ReceivePacket	
Category	Unit Test	
Description	<p><i>Java does not provide raw socket support so I had to work in C to send and receive packets on the network. I used WinPcap library for sending and capturing packets in windows environment.</i></p> <p><i>The data is passed though Java to C language when the packet is made and sent. Data is passed through JNI and sent to the receiver using WinPcap. At receiver an executable is made in visual studio using WinPcap which is called by a java programme.</i></p> <p><i>This test case checks whether the data sent by the sender is received on the destination or not.</i></p>	
Setup	Source	<ul style="list-style-type: none"> • WinPcap installed • Set visual studio environment • Include jni.h and the header file for native code in visual studio include field • Include winpcap.lib in library files
	Destination	<ul style="list-style-type: none"> • WinPcap installed • Set visual studio environment • Include jni.h and the header file for native code in visual studio include field • Include winpcap.lib in library files
Test Procedures	<ol style="list-style-type: none"> a. Run Receiver.java on the destination machine to call capture.exe b. Run the Test.java class on sender to send packet 	
Expect Results	On Success	<ul style="list-style-type: none"> • Packet received on the destination machine • The data in the packet is same as sent by the sender
	On Failure	<ul style="list-style-type: none"> • Packet would not be received on the destination machine • Packet received but not same • Can throw following exceptions on sender machine <ul style="list-style-type: none"> ○ LinkingError Exception ○ ClassNotFoundException ○ HeaderFileNotFound ○ Error opening adapter

		<ul style="list-style-type: none"> • Can throw following exceptions on destination machine <ul style="list-style-type: none"> ○ No such file as capture.exe ○ Error opening adapter ○ HeaderFileNotFound
Related Test Cases	SendPacket	
Notes	None	
Issues	None	

AES and IPsec integration test

Identifier	AESandIPsec	
Category	Integration Test	
Description	<i>I have implemented my own AES algorithm to include it in IPsec to provide confidentiality. This test case emphasizes on successful integration of AES module with IPsec.</i>	
Setup	<p>We want to protect the packet using IPsec in which AES would be used as encryption algorithm. On the system WinPcap should be installed and visual studio environment set. System should have the following classes:</p> <ul style="list-style-type: none"> • AES.java • PolicyChk.java • MakePacket.java • Receiver.java • And for capturing capture.exe should be present. 	
Test Procedures	<ol style="list-style-type: none"> a. Run Receiver.java on the machine to call capture.exe b. Run PolicyChk.java on the machine which calls makePacket.java to make packet. c. MakePacket.java calls AES.java to encrypt the data while applying IPsec on the packet d. The packet after IPsec processing is sent by native code e. The packet is captured by capture.exe and decoded, and the encrypted portion is decrypted 	
Expect Results	On Success	<ul style="list-style-type: none"> • The decrypted data is same as the data encrypted by AES that was put in the packet by IPsec

	On Failure	<ul style="list-style-type: none"> • The decrypted data is not same as the data encrypted by AES that was put in the packet by IPSec • Can throw following exceptions <ul style="list-style-type: none"> ○ ClassNotFoundException ○ No such file as capture.exe ○ Error opening adapter ○ HeaderFileNotFound
Related Test Cases	SenderPacket, ReceiverPacket and AES	
Notes	None	
Issues	None	

Sending and receiving through IPSec test

Identifier	SendReceiveIPSec	
Category	Integration Test	
Description	<p><i>IPSec is a framework of open standards developed by the Internet Engineering Task Force (IETF). IPsec provides security for transmission of sensitive information over unprotected networks like internet.</i></p> <p><i>This test case emphasizes on successful transmission of data from sender to receiver using IPSec.</i></p>	
Setup	We want to make packet and send it to receiver using IPsec and check whether packet received is the real one or not. Source and destination machine should fulfil following conditions.	
	Source	<ul style="list-style-type: none"> • WinPcap installed • Set visual studio environment • Include jni.h and the header file for native code in visual studio include field • Include winpcap.lib in library files • Source should have following files <ul style="list-style-type: none"> ○ AES.java ○ PolicyChk.java ○ MakePacket.java
	Destination	<ul style="list-style-type: none"> • WinPcap installed • Set visual studio environment • Include jni.h and the header file for native code in visual studio include field • Include winpcap.lib in library files

		<ul style="list-style-type: none"> • Receiver should have following files <ul style="list-style-type: none"> ○ Receiver.java ○ And for capturing capture.exe should be present.
Test Procedures		<ol style="list-style-type: none"> a. Run Receiver.java on the destination machine to call capture.exe b. Run PolicyChk.java on the machine which calls makePacket.java to make packet. c. MakePacket.java calls AES.java to encrypt the data while applying IPSec on the packet d. The packet after IPSec processing is sent by native code e. On receiver packet is captured and processed
Expect Results	On Success	<ul style="list-style-type: none"> • On destination machine it will print the data got. • The data got would be the same as sent by the sender.
	On Failure	<ul style="list-style-type: none"> • Packet would not be received on the destination machine • Packet received but not same • Can throw following exceptions on sender machine <ul style="list-style-type: none"> ○ LinkingError Exception ○ ClassNotFoundException ○ HeaderFileNotFound ○ Error opening adapter • Can throw following exceptions on destination machine <ul style="list-style-type: none"> ○ No such file as capture.exe ○ Error opening adapter
Related Test Cases	SenderPacket, ReceiverPacket and AES	
Notes	None	
Issues	None	