NEAR EAST UNIVERSITY

GRADUATE SCHOOL OF APPLIED AND SOCIAL SCIENCES

TRANSPORT LAYER SECURITY WITH MD5 ALGORITHM USING VISUAL BASIC APPLICATION

Rami Raba

Master Thesis

Department Of Computer Engineering

Nicosia 2004

NEU **JURY REPORT**

DEPARTMENT OF COMPUTER ENGINEERING Academic Year: 2003-2004

STUDENT INFORMATION

Full Name	Rami Raba		
Undergraduate degree	BSc.	**Date Received**	**Fall** 2001-2002
Institution	Near East University	**CGPA**	2.07

THESIS

Title	**Transport Layer Security with MD5 Algorithm Using Visual Basic Application**
Description	The aim of this thesis is development of client server based internet security, a visual basic program which has been developed applies MDS algorithm.

Supervisor	Assist.Prof.Dr. Dogan Halctanir	**Department**	Engineering

JURY'S DECISION

The jury has decided to accept /~ the student's thesis.
The decision was taken unanimously /~

JURY MEMBERS

Number Attending	3	**Date**	23/01/2004

Name	**Signature**
Assoc. Prof. Dr. Doğan İbrahim, Chairman of the jury	
Assoc. Prof. Dr. Rahib Abiyev, Member	
Assist. Prof. Dr. Firudin Muradov, Member	

APPROVALS

Date
23/01/2004

Chairman of Department
Assoc. Prof. Dr. Doğan İbrahim

DEPARTMENT OF COMPUTER ENGINEERING
DEPARTMENTAL DECISION

~23/01/2004

Subject: Completion of M.Sc. Thesis

Participants: Assoc. Prof Dr. Doğan İbrahim, Assist.Prof. Dr. Doğan Haktanir, Assist.Prof. Dr. Firudin Muradov , Assoc. Prof. Rahib Abiyev, Amjad Alhaj-yaseen, Wael Abu Dagga, Muberra hudai, Ramiz Salama, Jalal Swelim, Qais Albeni.

DECISION

W~ certify that the, student whose.number and aame.are.given, below,..bas- fulfilled all. the requirements for a M .S. degree in Computer Engineering.

		CGPA
980065	RamiRaba	3.21

Assoc. Prof. Dr. Doğan İbrahim, Committee Chairman , Chairman of Computer Engineering Department, NEU

Assoc. Prof. Dr. Rahib Abiyejt(Committee Member, Computer Engineering Department, NEU

Assist. Prof. Dr. Fi~din Muradov, Committee Member, Computer Engineering Department, NEU

Assist. Prof. Dr. Doğan Haktanir;'Supe;;~r, Electrical and Electronic Engineering Department, NEU

~Department
Assoc. Prof. Dr. Doğan İbrahim

Rami Raba : Transport Layer Security with MD5 Algorithm Using Visual Basic Application

Approval of tbe Graduate School of Applied and Social Sciences

Prof. Dr. Fakhraddin Mamedov
Director

We certi(y this thesis is~s~ for the award of the Degree of Master of Science in Computer Engineering

Examining Committee in charge:

/J-~~

Assoc. Prof. Dr. Doğan İorarum, Committee Chairman , Chairman or ComputerBngineering Department, NEU

Assoc. Prof. Dr. Rahib AYyev, Committee Member, Computer Engineering Department, NEU

Assist. Prof. Dr. Ffrıdiıı~mrnfttee Member, Computer r~Dep~ent, NEU

Assist. Prof. Dr. Doğan Haktanir, Supervisor , Electrical and' Electronic Engineering Department, NEU

ACKNOWLEDGMENT

I would like to thank my supervisor Assist. Prof Dr Doğan Haktanır for his overwhelming and the limitless help he had shown.

Also I would like to ode this work to my family who have supported and sponsored me to reach this point of study. My dearest father, who encouraged and helped me to study in order to reach a higher degree of education. My sweat heart mother, you are the best woman and the best mother in the world

Here I have to dedicate this work to the Palestinian babies and children those are in danger and to thosefallen as martyrs.

Finally, " I dedicate this thesis to my friends who support me and for making many helpful suggestions to improve this thesis: God bless you all".

ABSTRACT

This thesis specifies the Transport Layer Security protocol, a security protocol that provides communications privacy over the Internet. The protocol allows Client/Server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

To fully appreciate how web services will drive a security growth, a new development platform such as Scalability and Control of Security Capabilities, Authentication and Authorization, Confidentiality, integrity and Security Mechanisms for Web Services should be developed.

The main objective of this thesis is to investigate the security of client/server based computer models. In this thesis the author has developed a Visual Basic based program, which was the MD5 encryption algorithm to provide secure communication between a client and a server.

It is shown that the MD5 algorithm suited to client/server based security applications.

TABLE OF CONTANTS

INTRODUCTION

Internet is an open system. The identity of people, companies and computers communicating on the Internet is not easy to determine and validate. Furthermore, the very communication path is inherently insecure all communications are potentially Open for an eavesdropper to read and modify as they pass between the communicating endpoints.

Internet commerce requires secure communications. To order goods, a customer typically sends credit card details. To order life insurance, the customer might have to supply confidential personal data. Internet users would like to know that such information is safe from eavesdropping or alteration.

Task of providing a secure method to communicate over the web has proven to be somewhat tricky. There is not such as a perfect standard. Several competing standards exist, but SSL/TSL seems to be emerging as a winner. SSL *is* already being supported *in* major browsers and web servers. Internet Engineering Task Force has developed SSL standard further and has released the first version of - revisioned and renamed SSL - TLS. Secure.

This thesis explains the work of Transport Security Layer within internet security also provides the basic principle of the MD5 data encryption algorithm and the methods of communication between a client and a server is described. The aim of the thesis is to investigate the client/server based internet security. A Visual Basic based program has been developed by the author, which applies MD5, based internet security.

The thesis consists of the introduction, six chapters, and appendix, which contain the program developed by the author.

Chapter 1 is an explanation of servers, web services and internet essentials are described in this chapter.

Cryptography, web security, problems and the various data encryption models are presented in chapter 2.

Chapter 3 discusses the Transport Layer Security technique, which is privately transfer information across the internet in details.

Chapter 4 presents Transport Layer Security protocols in details based security problems. TLS protocols essentials and the differences between TLS protocols and other security protocols are described here.

Chapter 5 is describes the Message-Digest security algorithm MD5 in detail.

Chapter six is the most important chapter of this thesis, which describes the stenography in details. The author developed a Visual Basic based security application program, using MD5 encryption. The details of this program, block diagram are given in this chapter.

Conclusion presents the results of the author's application and makes suggestions for future investigation.

Finally, the program developed by the author is given in an appendix at the end of the thesis.

1. SERVERS AND WEB SERVICES SECURITY

1.1. Overview

Web services are fast becoming a reality for both developers and enterprises. Web services make it easy for enterprises to integrate existing legacy applications. Protected behind firewalls, enterprise Web services exchange sales, inventory, and customer relationship data, even though the applications that manage that data reside on diverse software platforms.

Web Services security has been the most talked about thing in the Web Services arena for quite some time now. If there's one thing that has slowed the widespread acceptance and implementation of Web Services, it's their lack of security standards.

What this entire means, in practical terms, is that Web services can announce themselves across the Internet and expose their functionality to other applications ("I'm the Village Bank Online Loan Application Service"). In turn, they can search for and invoke other Web services, ("I want to check an applicant's credit rating with any one of my three credit bureau partners"), and they can interact with those services, exchanging requests for information ("Tell me John Smith's credit rating") and receiving responses ("A-plus").

1.2. Server Security

The servers have now been physically secured and isolated on the network. The next step is to secure access to the servers. Because the most damaging network attacks require gaining access to a server, it is important to restrict access as much as possible to all servers.

Start by limiting servers to single-use machines. The web server should be used only to serve websites; the mail server should only be used for mail, and so on. It is easier to manage security on individual servers if an administrator can limit the number of services running on them.

A good example of this is the X-Windows management system that is installed on many Unix systems by default. Because Unix servers are generally managed through the command line, keeping the X-Windows system installed only leaves unnecessary accounts installed and potential' security holes open. Of course in order for this strategy to work, it is also important to foJlow through and disable any services that are unused. Not only should unnecessary services be disabled, but whenever possible they should be removed, or not installed in the first place.

Services should never be run as the administrative user. It may be necessary to start the service using the administrative user account to *bind it* to the port, but the service should continue to run as a nonprivileged user. If the service is run as a nonprivileged user then if the service is compromised it is less likely that an attacker will be able to cause any further damage to the network. In addition to running services as nonprivileged users, unnecessary accounts should be removed from servers, or at a minimum, renamed. The Windows guest account is an example of an unnecessary account that should be deleted The same goes for the Unix games, bin, and sys accounts as well as other well-known accounts. If the accounts cannot be removed entirely they should be configured with no login capabilities and a very restrictive password. *All* accounts on all network devices, but especially on servers, should have passwords. Accounts created on the server should be subject to the same password policy as the rest of the network. This means that the passwords should be changed at regular intervals, and they should be sufficiently difficult to crack using password cracking tools.

Whenever possible information about user accounts should be stored in a centralized database. This helps decrease the likelihood that a wayward account will be created on a server. It also makes it easier for employees to hop from server to server, as Jong as they have permission to access the server. In cases where a centralized user database is in use, the authentication from the servers to the user database should use some sort of encryption. Kerberos is the type of authentication most often associated with *this* type of server management. Kerberos is also nice, because Unix operating systems, and Windows 2000, support it, making it possible for a user to authenticate against both types of servers, if

necessary. This does not mean that users should have the run of the network; servers should be configured so that only specific groups have access to specific servers. Users in the KCOunting group should not need access to the sales server, and so on. Again, limiting who has access to a server will limit the amount of damage an attacker who gains access to one of the servers can do. Files on the system should be restricted as well. Acommon way of enforcing file security is to create separate partitions on the server. One partition should be used for system files, while a separate partition can be used for user files. This İs a common practice on Unix servers, which are generally broken into /, /usr, kete, /opt, and others, depending on the needs of the server administrator. This is a less common practice on Window-based servers, but one that should be followed, even if it is as simple as putting the system files on a C:\ and user files on an F:\ partition. Partitioning servers has two effects:

- It separates system information in a logical manner. If something happens to one of the partitions, data on the second partition is usually safe.
- It creates a separate root directory for users. Users on the F:\ partition, or in the /usr partition see F:\ and /usr, respectively, as their root directories, and are unable to access the system files on the other partitions. While this is not perfect security, it does make the job of an attacker more difficult Ref [l].

1.3.Web Server Security

An organization's web sewer is an attacker's most common point of entry into the network. A web server is, almost by definition, a public server, so it makes an attractive target to attackers. In addition, depending on the nature of the website, breaking *into* the web server may give an attacker access to valuable customer information. Because web servers are such attractive targets, special steps need to be taken to secure the web server against attackers. The web server should be a single-use server, and should have a very restricted access policy; only personnel who absolutely need access should have it. In fact, a staging server *is* commonly used as a means of further restricting access to the actual web server.

A staging server is a replica of the web server. It should have the same operating system, same patches, same file structure, and all of the same software as the web server. Content [illegible] for the web server is loaded to the staging server and the pushed to the web server using software like RedDot Solution's Content Management Server (CMS). Different users, or departments, are given accounts on the staging server; the accounts are used to upload [illegible] to the staging server. The content is pushed from the staging server to the actual web server using a separate account to which the users do not have access. The web server is configured to only allow the staging account access from the JP address of the staging server Ref (12].

1.3. I. SSL Encryption

A common way to enhance the security of transactions conducted through a website is to use Secure Sockets Layer (SSL) encryption. Netscape developed SSL as a means of securing web-based transactions. The IETF has recently adopted most of the SSL specification to create a new protocol, designed to secure more than web-based transactions, called Transport Layer Security (TLS). SSL uses public key cryptography developed by RSA Data Securers loc. to secure transactions. Public key cryptography uses a public/private key pair to validate the information being sent between parties.

The default SSL port is 443 TCP; however, SSL can run over other ports, and many administrative applications now require SSL connections over different ports.

Traditionally, to make an SSL connection to a web server to Port 443, a user simply typed in https://www.example.com;if the connection were to be made on a different port, the user would type httlls://www.example.com: [port numbers]. The 'effect is the same: The transactions between the user and the web server will be encrypted.

The process of setting up an SSL on a web sewer varies depending on the type of web server being used, but the backend functionality is the same across all web servers. The public/private key pair is generated on the web server. The private key is stored separately from the public key, which is sent to a Certificate Authority (CA), such as VeriSign or

llawte. The CA verifies the identity of the organization that sent the key and issues an x.509v3 certificate. The x.509v3 certificate contains the organization's public key and is signed by the CA. The newly generated certificate is *installed* on the web server, and *SSL* sessions can begin.

When a user attempts to connect to a web server using SSL, several things happen. The first is that the user initiates an SSL connection with the web server, and tells the web server the minimum SSL version it will accept (Version 3.0 is the current standard). The web server responds with a copy of the x.509v3 key. The web browser chooses a random symmetric key, encrypts it with the server's x.509v3 key, and sends it back to the web **1aver.** The web server decrypts the symmetric key using the server's private key.

1be web server and the client use this symmetric key to encrypt traffic during the SSL session. The web server also assigns a unique ID to each encrypted session, called an SSL Session ID. The SSL Session ID is unencrypted and sent between the user and the server **with** each request

SSL is a great tool for encrypting data between users and a web server. It should not be thought of as a tool for securing a web server. SSL does not assist with securing data once it is on a server. This is an important point: Many administrators feel that if they have SSL encryption their web server is automatically secure. That is not the case. SSL encryption should be used with other security measures as part of an overall web server security solution.

1.3.2. Load Balancing

So far in this section, the discussion has revolved around attackers attempting to deface a website, and attempting to get customer information. Those attacks are common, but another common attack faced by web server administrators is a DOS attack. DOS attacks are different in that they often serve no other purpose than to see if the attacker can take a website offiine. The attacker may have a grudge against an organization, or may have been

frustrated in attempts to attack a server in other ways. Whatever the reason, a DOS attack against a web server is often difficult to defend against, especially if it is a DDOS attack. A typical website consists of a single server, with a database server sitting behind it. Even if a DOS attack does not saturate an Internet connection, it is not difficult to generate enough requests to knock a server offiine. Firewalls can stop many, but not all, DOS attacks, especially if the attacks appear to be a great number of legitimate requests-which they often are. A common method used to protect a website against DOS, and other types of attacks, is to load balance the site across multiple servers. Load balancing has traditionally been used to distribute traffic across multiple servers as a means of improving performance and customer experience. *As* some websites increased in popularity, a single server was not enough to handle the number of requests received.

Two other farms of load balancing have become popular for use with web servers: clustering and network load balancing. Clustering has been used with their types of servers for many years, but it has only recently become common pace for web servers. A cluster is a series of servers that act as a single server. The servers either communicate with each other or with a cluster controller to process requests as they are made. The cluster is assigned an IP dress, and each individual server can learn that IP via ARP to requests for that P address. The individual servers are also assigned unique IP addresses, which allow for server management and communication between the servers.

When a request to the web server is made, the clustered servers determine which is going to accept it based on a preprogrammed set of rules, known as metrics.

Depending on the metrics used, a single server may handle all requests from one source IP address, or the requests may be distributed between the servers. If one of the servers fails, the other servers in the cluster pick up its requests and service goes uninterrupted Ref [1].

1.4. Mail Server Security

There are two types of mail server security that need to be considered: the message transfer agent (MTA) and the user mailboxes. Most security resources are focused on the Mf A,

because that is the program responsible for routing mail to and from network users. It has also, traditionally, been one of the weakest security points on the network. Weak MTA security cannot only lead to attacks on a network, it can also result in mail from an organization being blacklisted. An organization that winds up on an MTA blacklist will be unable to send mail to sites that subscribe to the blacklist Ref [11].

1.4.1. Mail Scanning

Mail scanning is a common practice on enterprise networks. E-mail messages should be scanned for three things; viruses, UBE, and employee theft or violation of e-mail policy. As with any other security policy when an e-mil security policy is implemented it should be dearly explained to network users. The explanation of the security policy should detail what is being scanned for, as well as why it is being implemented.

E-mail scanning is sometimes controversial because users think of their e-mail account as a private communication tool as opposed to a communication tool owned by the company. A detailed explanation, coupled with a gentle reminder that the mail server and its contents are property of the organization can help ease any tension a new policy may cause. Prior to presenting the scanning policy to employees it should be reviewed by an organization's legal counsel. Privacy laws can vary from area to area. *It* is important that the policy be presented in such a way that an organization cannot be sued for scanning activities, or worse, sued because the scanning missed something.

Virus scanning is the most common form of mail server scanning and is something that should be implemented on every enterprise network.
No matter how careful network users are when it comes to viruses, worms, and opening of attachments, users from remote networks may not show the same level of care.

A virus or worm that is introduced into a network can cause tremendous damage before it is stopped.

Incoming and outgoing mail messages should all be scanned for viruses and worms. Many

companies, like Trend Micro and Sophos, make virus scanners designed specifically for mail gateways. The products are fast and able to keep up with even a heavily loaded mail server.

Virus scanning on the mail server does not mean that virus scanning software should not be used on end user workstations. Instead, the two should be used in tandem, preferably running software from different vendors on the server and the workstations. Running two different virus detection software products decreases the chances that a new virus will slip through an organization's defenses.

Virus definitions on the mail server should be updated frequently, preferably daily. The mail server is the first line of defense against viruses and should have the most up-to-date information to prevent the viruses from spreading.

In addition to virus scanning, many companies have begun scanning mail messages for UBE. UBE has proliferated to such a point that some organizations have found that 20 percent or more of their incoming mail is UBE. There are many programs on the market that can help an organization eliminate UBE at the mail server. Messages are automatically deleted or filed away for review.

These programs help shift the burden of deleting unwanted messages from employees to the server, where the process is automated Ref [l].

1.5. **Outsourcing**

Mail servers and web servers are unique in that they are intended to be public. People from all over the world have to be able to reach an organization's web and mail servers to find out information and communicate. The uniqueness of these servers requires special security consideration, and the security monitoring and maintenance takes up an inordinate amount of time, compared to other servers.

Many organizations may choose to outsource the management of mail and web services to an ISP or Application Service Provider (ASP). Outsourcing mail and web service needs has several advantages, the primary one being that it can increase the security of the corporate network. Web server attacks are the most common type of external network attack. If the web server is removed to a remote location, even if it is successfully compromised, the rest of the network will most likely remain intact. In addition, outsmarting allows security administrators to tighten network security and further restrict the type of access allowed into the network, increasing overall network security Managed hosting and mail providers also usually have staff onside 24x7 monitoring for network anomalies and looking for security breaches. Non-technical organizations may not have the staff or the resources, to monitor systems around the clock in this manner.

Some companies specialize in secure web and mail server hosting. These companies generally use secured versions of operating systems and provide very restrictive access to Servers. Setting up these services can be outsourced as well. A Managed Service Provider (MSP) will help an organization create secure web and mail solutions, either at the customer site or in a remote data center. These MSPs build the web infrastructure and handle the day-to-day maintenance of the servers. Some will also handle any security monitoring needs for an organization.

An organization that does not have a lot of experience with mail and web server. Maintenance should consider the possibility of outsourcing.
The monthly costs involved are minimal compared to the costs of hiring competent staff, and certainly are nothing compared to the potential damage that can be caused if a web server is compromised.

1.6. Web Services Value Proposition

With new development platforms - Web services mechanisms, such as XML formatting and SOAP messaging, are an organic part of the application development environment. This makes it faster, easier and more cost-effective to integrate heterogeneous resources,

both within enterprises and between trading partners. In tum, by reducing integration costs, enterpiises can undertake a wider range of integration efforts, including initiatives that could not previously be cost-justified.

A Web services application can be designed to aggregate information and services from multiple back-end systems. This makes it possible to perform routine tasks more efficiently, for example, accessing data from multiple systems in order to calculate a business unit's profit and loss.

Because Web services are self-contained and modular, they can be reused for multiple purposes. For example, an enterprise's shipping status service can be invoked by the sales order system and the customer care system Ref[12].

Support for Federated Identity Management:

Web services play a key role in solutions for federated identity management. A federated approach to identity management enables enterprises to enhance the user experience by providing single sign-on (SSO) and seamless navigation across multiple domains. The business benefits of Web services are well understood. These include faster time-to-market for new applications and services, reduced business process costs and reduced partnering costs. Gartner Research predicts that the delivery of software through a subscription-based hosting model will eventually eliminate the need for most software to be licensed and sold in physical packaging.

In tum, this will greatly reduce IT costs related to software distribution and maintenance. In addition to these gains, enterprises will implement new business models and innovative offerings that attract new customers, enhance revenue and provide a competitive advantage Ref [2].

1.7. Web Services and Network Computing

There is a strong momentum on several fronts to bring Web services into the mainstream of network computing. There are many proposals that have been put forth and are inVarious stages of review by two key standards bodies: the WorJd Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS).

Many of these proposals address the requirements of specific industries. In fact, the outpouring has been so great that W3C is looking at mechanisms for determining which of these proposals are truly foundational and which are more suitably developed by communities of interest. Web services have also inspired a high level of cooperation among technology vendors. For example, Microsoft and IBM have developed a proposal for a Global XML Web Services Architecture (GXA).

Support for Web services is further reflected in vendors' product strategies. Virtually all-maüor hardware and software vendors have committed to introducing Web services technoJogy into their offerings.

While enterprises are carefully weighing all their technology investments, the pace of Web services implementations is beginning to gain speed. As Forrester Research notes, "early adopters", such as Ford are using Web services today to solve problems that traditional EAi and B2Bi products can't tackle cost-effectively: lower-volume integration between departmental apps and automated exchanges with small suppliers Ref [2].

1.8. Scalability and Control of Security Capabilities

Today, most large enterprises support dozens of major applications, each with its own unique security implementation. Typically, this results in multiple schemes for authentication and Web access management. In the future, as Web services gain wide accep,tanceadding hundreds or even thousands of discrete services to the mix this already unwieldy approach will be unsustainable.

In addition, the distributed nature of Web services makes it more difficult than ever to enforce security policies. As Forrester Research points out, "With CORBA and DCE, IT bosses could ensure security by exercising control: Only a small set of security-savvy developers could touch the code. But with [new tools], novices are easily building and deploying Web services interfaces to critical data and unknowingly exposing their firms to security risks."

Although it won't happen tomorrow, Web services will make it possible to create a unified security infrastructure that functions much like a utility, delivering security services to be consumed by other Web services across the enterprise and between trading partners. As a result, proven capabilities such as authentication, authorization and encryption services will be applied in new ways to protect a much wider set of resources than is possible today Ref [2].

1.8.1. Authentication and Authorization

In the current e-business environment, authentication and authorization solutions focus largely on human-machine interactions, addressing the issue of who is at the other end of a network connection and determining what resources an authenticated user can access. The growth of machine-to-machine interactions requires new mechanisms for Web services to establish trust with each other, most probably through public key authentication.

In addition, demand for federated identity and single sign-on (SSO) solutions create a whole new set of challenges.

For example, how is user identity and context information shared among multiple enterprises while still ensuring the confidentiality needs of the end user? What are the conditions under which a Web service accepts a security assertion, for example about authentication or authorization, from an external party? Ref [10].

1.8.2. Confidentiality and Integrity

Current technologies, most notably cryptography and certificate management, offer well-developed mechanisms for protecting whole documents as they cross public and private networks. In addition to safeguarding confidentiality through encryption, current solutions enable recipients to authenticate the sender, via digital signatures, and verify text integrity to ensure a document has not been tampered with.

These proven capabilities can be applied to ensure the privacy and integrity of communications in a Web services environment. New Web services standards make it possible to selectively apply encryption and digital signatures to different parts of a document. This will support more complex business scenarios, for example, enabling multiple parties to digitally sign complex contracts.

1.8.3. Security is Paramount

With the first generation of Web and e-business technologies, many enterprises made the decision to "implement now and secure later." Now they realize that addressing these fundamental security challenges is one of the prerequisites for the long-term success of Web services - and e-business. For this reason, they are urging vendors to accelerate the development of security standards into the Web services framework, and vendors are responding.

1.8.4. Scalability

Scalability - An enhanced security infrastructure should scale to support thousands, or even millions of users. Entrust solutions scale to the required levels through high capacity, high availability, advanced network bandwidth management and reduced deployment costs:

- High capacity - Support for unlimited administrators and up to 10 million users per CA. A single UNIX server can now support up to 25 separate CAs, an attractive capability for ASPs and ISPs.

High availability - Automatic key and certificate update enables users to continually work in a secure environment even after original keys and certificates expire. Support for disaster recovery with redundant servers provides high availability to meet your requirements.

Advanced network bandwidth management - Users can cache certificates and certificate revocation lists (CRLs), reducing the need for directory communication over the network Ref [3].

, Web Services and Security Mechanisms

ditional security technologies used on the Internet often aren't sufficient for Web vices, The major problem is their transport dependence. For example, the most widely- d security technology, SSL (Secure Socket Layer), is tied to the network communication point, More precisely, the identity can be assigned only with the communication point that is usually shared by many Web Services.

er security technologies, namely GSS-API (Generic Security Service Application grammers Interface) and security mechanisms based on GSS-API like SPKM (Simple die Key Mechanism) and Kerberos, are designed specifically for use in loosely coupled iitectures. The GSS-API is independent of both the transportation and security :hanisms(security mechanism independence means that the underlying technologies for ıtography, identity representation, and data signing are fully encapsulated). The most ely used security mechanism in Web Services today is still SSL, but adoption ofSPKM Kerberos is increasing. SPKM is based on the asymmetric cryptography elements lained above, so is well suited to the widely dispersed environment of Web Services; beros uses symmetric cryptography.If you're concerned about security issues, then Web rices products and technologies that support these more advanced security options are ınswer Ref [3].

1.10. Summary

Many people feel the Internet is not secure because it is a public network. It is true that connecting an internal network to the Internet can expose the internal systems of a company to very considerable risks. The transmission of a document between two parties over the Internet may result in the document passing through half a dozen networks, each managed by different organizations.

Server security is important because servers are the last line of defense against an attacker. While router and firewall security breaches are on the rise, servers are still the number-one targets of attackers, and all servers should be configured to be as secure as possible.
The best way to ensure server security is to limit that has access to the server, limit which interface the server can be accessed on, and enforce a strong password policy. If these steps are combined with regular software patch updates, most servers will be relatively secure.
Public servers, such as web and mail servers are a different story and these servers have special security considerations. These servers have to allow access to anyone, but they can be configured to restrict direct access, except through the required ports, and they can be made to be more secure.

Web and mail server services can also be outsourced to one of many companies that provide this type of service. If an organization does not have the in-house expertise to manage these servers in a secure manner, outsourcing may be a viable option.
Information and transaction security schemes such as secret key encryption and public key encryption are used to ensure the privacy, integrity and confidentiality of business
Transactions and messages.

2. WEB SECURITIES AND CRYPTOGRAPHY

2.1. Overview

Virtually all businesses, most government agencies, and many individuals now have Web sites. The number of individuals and companies Internet access is expanding rapidly, and all of these have browsers. As a result, businesses are enthusiastic about setting up facilities on the Web for electronic commerce. But the reality is that the Internet and the Web are extremely vulnerable to compromises of various sorts. As businesses wake up to this reality, the demand for secure Web services grows.

Cryptography is a very old study, but since the advent of widespread use of computers *in* the last few decades, it has advanced at an enormous rate. Computers make trivial the cryptanalysis of all but one of the cryptographic algorithms that predate them (the One-time Pad). They also make practical the use of algorithms that are able to resist computer-assisted cryptanalysis effectively. However, inventing such algorithms is not a sport for amateurs; common sense can only be brought to bear on cryptography by people with the training to properly understand all the most advanced techniques known for computer-aided cryptanalysis.

What this means is that when we as engineers wish to apply cryptography to creating secure systems, we must confine our usage to well-known, well-analyzed algorithms and protocols.

2.2. **Web Security**

Secure Web Server provides security using a combination of the Secure Sockets Layer (SSL) protocol and (in most cases) CA-approved digital certificates. SSL handles the encrypted communications and the *mutual* authentication between browsers and Secure Web Server. The CA-approved digital certificate provides authentication for Secure Web Server (the CA puts its reputation behind its certification of our organization's identity).

The topic of Web security is a broad one and can easily fill a book (several are recommended at the end of this chapter). In this chapter, we begin with a discussion of the general requirements for Web security and then focus on two standardized schemes that are becoming increasingly important as part of Web commerce: SSL/TLS and SET Ref [4].

2.2.1. Architecture of the Web

- Hyper text information model (linking of document)
- Cliriet consultation protocol

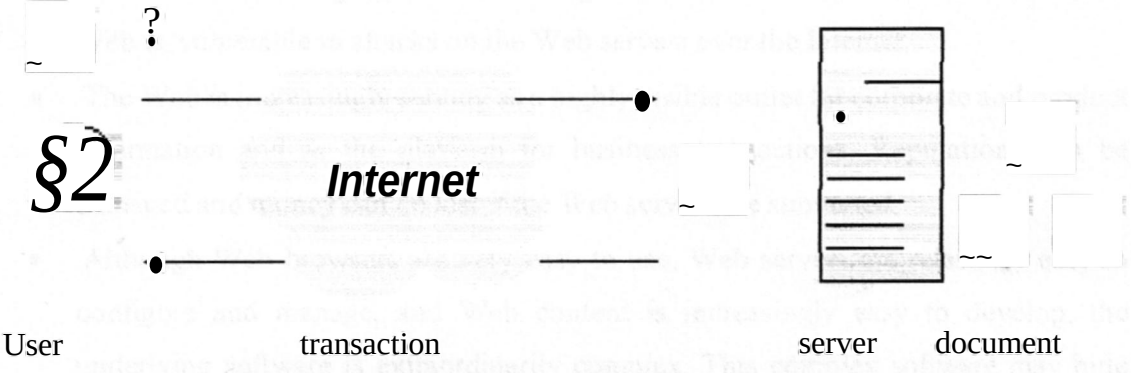


Figure 2.1. Architecture of the Web

2.2.2. Security Measures

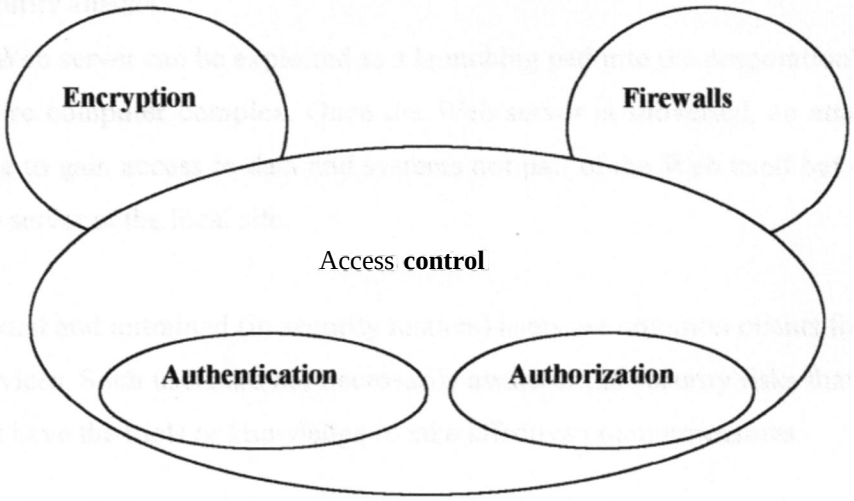


Figure 2.2. Security Measures

2.2.3. Web Security Considerations

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. As such, the security tools and approaches discussed so far in this book are relevant to the issue of Web security. But, as pointed out in [GA-7], the Web presents new challenges not generally appreciated in the context of computer and network security:

- The Internet is two ways. Unlike traditional publishing environments, even electronic publishing Systems involving teletext, voice response, or fax-back, the Web is 'vulnerable to attacks on the Web servers over the Internet.
- The Web is increasingly serving as a highly visible outlet for corporate and product information and as the platform for business transactions. Reputations can be damaged and money can be lost if the Web seıvers are subverted.
- Although Web browsers are very easy to use, Web servers are relatively easy to configure and manage, and Web content is increasingly easy to develop, the underlying software is extraordinarily complex. This complex software may hide many potential security flaws. The short history of the Web is filled with examples of new and upgraded systems, properly installed, that are vulnera-ble to a variety of security attacks.
- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the 16cal site.

- Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

2.2.4. Web Traffic Security Approaches

A number of approaches to providing Web security are possible. The various approaches that have been considered are similar in the services they provide and. to some extent, in the mechanisms that they use, but they differ with respect to the. Scope of applicability and their relative location within the TCP/IP protocol stack. One way to provide Web security is I use IP Security. The advantage of using IPSec is that it is transpire to end users and applications and provides a general-purpose solution. Further IPSec includes a filtering capability so that only selected traffic need incur the overhead ofIPSec processing.

2.2.5. Web Security Threats

Table 2.1 provides a summary of the types of security threats faced in using the Web. One way to group these threats is in terms of passive and active attacks. Pas-sive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a W eh site that is supposed to be restricted. Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site.

Another way to classify Web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server. Issues of server and browser security faJJ into the category of computer system secu-rity; Part Four of this book addresses the issue of system security in general but in also applicable to Web system security. Issues of traffic security fall into the category of network security and are addressed in this chapter.

	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan house browser • Modification of • Memory • modification of message traffic in transit	• Loss of information • Compromise of machine ' Vulnerability to all other threats	Cryptographic Checksums
Confidentiality	• Eavesdropping on the net • Theft of info from sever • Theft of info from client • Info about network Configuration • Info about which client	• Loss of information • Loss of privacy	Encryption , web proxies
Denial of service	• Killing of user threads • Flooding machine with bogus threats • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	' Misrepreseıtatio n of user • Belief that false information is valid	Cryptographic techniques

Table 2.1. a comparison of threats on the web

2.2.6. Secure Socket Layer and Transport Layer Security

Netscape originated SSL. Version 3 of the protocol was designed with pub-lie review and input from industry and was published as an Internet draft document. Subsequently, when a consensus was reached to submit the protocol for Internet standardization, the TLS working group was formed within IETF to develop a com-mon standard.

The current work on TLS is aimed at producing an initial version as an Internet Standard. This first version of *TLS* can be viewed as essentially an SSLv3.1and is very close to and backward compatible with SSLv3.

The bulk of this section is devoted to a discussion of SSLv3. At the end of the section, the key differences between -ssLv3 and TLS are described.

2.2.7. SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure 2.3. The SSL Record Protocol provides basic security services to various higher layer protocols. In particular, the hypertext transfer protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, the Change Cipher Spec Protocol, and the Alert Protocol. This SSL Figure 2.3 shows SET on top of HTYP; this is a common implementation in some implementations, SET makes use of TCP directly. Specific protocols are used in the management of SSL exchanges and are examined later in this section.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:

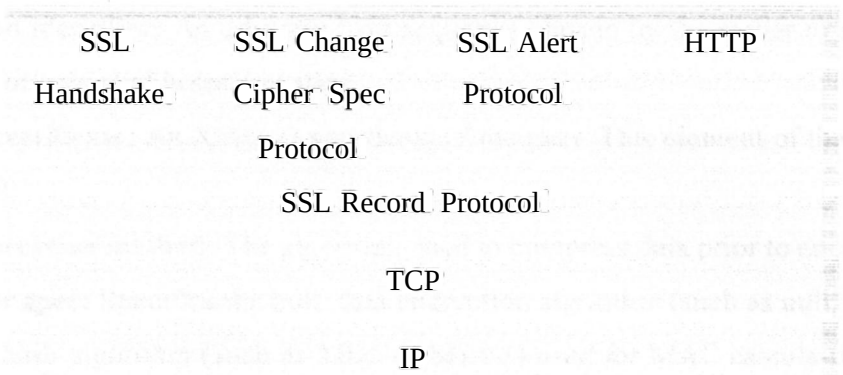


Figure 2.3 SSL Protocol Stack

- Connection: A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is asso-ciated with one session.
- Session: An SSL session is an association between a client and a server. The Handshake Protocol creates sessions. Sessions define a set of cryptographic security parameter's, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security para.me-ters for each connection.

Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice.

There are actually a number of states associated with each session. Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and writes states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states.

A session state is defined by the following parameters (definitions from the SSL specification):

- Session identifier: An arbitrary byte sequence chosen by the server to identify an active or presumable session state.
- Peer certificate: An X509 .v3 certificate of the peer. This element of the state may be null.
- Compression method: The algorithm used to compress data prior to encryption.
- Cipher spec: Specifies the bulk data encryption algorithm (such as null, DES, etc.) and a hash algorithm (such as l\.ID5 or SHA-I) used for MAC calcula-tion. It also defines cryptographic attributes such as the hash size.
- Master secret: Forty-eight-byte secret shared between the client and server.
- J.s presumable: A flag indicating whether the session can be used to initiate new connections.

A connection state is defined by the following parameters:

- Server and client random: Byte sequences that are chosen by the server and client for each connection.
- Server write MAC secret: The secret key used in MAC operations on data sent by the server.
- Client write MAC secret: The secret key used in MAC operations on data sent by the client.
- Server write key: The conventional encryption key for data encrypted by the server and decrypted by the client.
- Client write key: The conventional encryption key for data encrypted by the client and decrypted by the server.
- Initialization vectors: When a block cipher in CBC mode is used, an initialization vector (N) is maintained for each key. The SSL Handshake Protocol first initializes

this field. Thereafter the final cipher text block form each record is preserved for use as the N with the following record.

Sequence numbers: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed $2^{64} - 1$ Ref [5].

2.3. Secure Electronic Transaction

SET is an open encryption and security specification designed to protect credit card transactions on the Internet. The current version, SETvl, emerged from a call for security standards by MasterCard and Visa in February 1996. A wide range of companies were involved in developing the initial specification, including IBM, Microsoft, Netscape, RSA, Terisa, and VeriSign. Beginning in 1996, there have been numerous tests of the concept, and by 1998 the first wave of SET-compliant prod-ucts was available.

SET is not itself a payment system. Rather it is a set of security protocols and formats that enables users to employ the existing credit card payment infrastructure on an open network, such as the Internet, in a secure fashion. In essence, SET pro-vides three services:

- Provides a secure communications channel among all parties involved in a transaction
- Provides trust by the use ofX.509v3 digital certificates
- Ensures privacy because the information is only available to parties in a transaction when and where necessary

2.4. Terminology of Cryptography

Cryptography has developed an extensive vocabulary. More complex terms will be introduced gradually throughout the thesis. However a collection of basic terms is presented below.

There is a list of basic security requirements. This list includes: secrecy *(or* confidentiality) authenticity integrity, and nonrepudiation. Secrecy ensures that information flow between the sender and the receiver is unintelligible to outsiders. It protects information against threats based on eavesdropping. *Authenticity* allows the receiver of messages to determine the true identity of the sender. It guards messages against impersonation. Substitution or spoofing integrity enables the receiver to verify whether outsiders while in transit via an insecure channel have tampered *with* the message. It ensures that any modification of the stream of messages will be detected. Any modification that results from changing the order of transmitted messages deleting some parts of messages or replaying old messages will be detected. *Nonrepudiation* prevents the sender of a message from claiming that they have not sent the message.

Encryption was the first cryptographic operation used to ensure secrecy or confidentiality of information transmitted across an insecure communication channel. The encryption operation takes a piece of information (also called the message, message block, or plaintext) and translates it into a cryptogram (ciphertext or codeword) using a secret cryptographic key. Decryption is the reverse operation to encryption. The receiver who holds the correct secret key can recover the message (plaintext) from the cryptogram (ciphertext).

The step-by-step description of encryption (or decryption) is called the *encryption algorithm (*or decryption *algorithm).* If there is no need to distinguish encryption from decryption, we are *going* to cal] them coJJectiveJy ciphers, *Crypto algorithm* or *cryptosystems.*

Private-key or *symmetric* cryptosystems use the same secret key for encryption and decryption. More precisely, the encryption and decryption keys do not need to be identical the knowledge of one of them suffices to find the other (both keys must be kept secret).
Pnb l ic-key or asymmetric cryptosystems use different keys for encryption and decryption. The knowledge of one key does not compromise the other.

Hashing is a cryptographic operation that generates a relatively short *digest* for a message of arbitrary length. Hashing algorithms are required to be collision-resistant, i.e. it is "difficult" to find two different messages with the same digest Ref [4].

2.5. Technologies Enabled by Cryptography

- Certificates and Digital Ids
- Digital Signatures
- Secure communications channels

2.5.1. How Does It Work?

- Algorithm = the computational procedure used to hide and unhide information
- Key = a value that causes the algorithm to run in a specific way and produce specific ciphertext.

2.6. Introduction to Cryptosystems

Make any enquiry about computer security, and you will almost immediately fall over the terms cryptography and encryption (and also decryption), but what exactly is meant by this?

The dictionary (in my case the Oxford English), defines cryptography as hidden writing. It has been around for a very long time. The Ancient Egyptians, the Arabs and the Romans developed their own systems. An encapsulation of cryptology presents the ideas that are the foundation of cryptology. In this chapter I also introduce the concept of a cryptosystem, as well as the building blocks of a cryptosystem, namely cryptography and cryptanalysis Ref (15].

2.6.1. What is it used for?

Cryptography is used whenever someone wants to send a secret message to someone else, in a situation where anyone might be able to get hold of the message and read it. Generals to send orders to their armies or to send messages between lovers often used it. The most famous encryption machine invented was the Enigma, used in the Second World War to send military messages.

2.6.2. The Science of Cryptology

This is an extensive discussion of the science of cryptology. One can find introductions to various concepts in cryptology. Sections are dedicated to cryptography (the science of code-making) as well as cryptanalysis (the science of code-breaking). At the beginning of chapter, we have included any mathematical topics that may be necessary to understand the material for the given. Also, there is a cross-reference of other crypto logical ideas that relate.

2.6.3. Architecture of Cryptography

Cryptography is divided into two parts: private-key encryption and public-key encryption. A private-key cryptosystem is one that is intended to be used among a small group of people. This type of cryptosystem is easily controlled. A public-key cryptosystem is encryption at a much larger scale. There are many participants as well as many attempting to break the codes. This cryptosystem is not easily managed and requires sophisticated tools.

2.6.3.1. Private-key Encryption

Private-key encryption utilizes substitution ciphers. While these are the easiest to understand, they are also the easiest to break. The level of understanding of these ciphers is suitable for the general public. The following are common private-key encryption schemes:

- Caesar Cipher: the Caesar Cipher involves replacing each letter of the alphabet with the letters standing three places further down the alphabet..

- Affine Cipher: As there are only 25 distinct shift transformations, the task of deciphering an intercepted message encoded with such a transformation is quite simple. In other words, a shift transformation is not a secure encoding device. A more secure system that is as easy to use as a shift cipher is an affine transformation. An affined transformation is a function of the form is of the form

$$C = f(P) = (aP + b) \bmod N,$$

Where P is the numerical value assigned to the plaintext character to be encoded, a and b are integers, and N is the number of characters in the plaintex:talphabet Ref [22].

- Random Cipher: A random substitution cipher encodes a message using a one-to-one function that randomly maps each letter A, B, .., Z to a letter in the same alphabet. The image of the function, then, is the encoded text of the message.

- Vigenere Cipher: The Vigenere cipher has been widely used to develop cryptosystems dating back to the 16th century. Its basic construction is a combination of a Caesar shift combined with a keyword. To generate the substitution, under the written alphabet the code word is written with the remaining letters following the word itself:

- Transposition Cipher: The transposition cipher is actually a very simple idea. Instead of replacing the characters with other character, this cipher simply changes the order of the characters. For decoding purposes, because we are not substituting anything, this cipher is not affected at all by a frequency analysis. The key for this cipher is also not standard. Instead of a list of alphabetic substitutions, it is a mapping order.
- Digital Encryption Standard (DES)

2.6.3.2. Public-key Encryption

Public-key encryption is a horse of a different color. The encryption schemes used are more difficult to implement; however, they are far more secure than the above ciphers. An introduction to the theory of public-key cryptosystems is provided. The material on the following pages is intended for a more mathematically sophisticated audience. These are popular (and used) public-key encryption schemes:

- Rivest, Shamir, Adelman (RSA)
- Pretty Good Privacy (PGP)

2.6.4. Cryptanalysis

While, cryptography is dedicated to encrypting messages, the focus of cryptanalysis is the break those codes. This discipline is the study of decoding information without the use of a known key. In this section, we present various algorithms to decipher encrypted messages Ref [16].

- Exhaustive Key Search
- Frequency Analysis

2.7. Change Cipher Specification Protocol

The change cipher specification protocol exists to signal transitions in ciphering strategies. The protocol consists of a single message, which is encrypted and compressed under the current (not the pending) CiperSpec. The message consists of a single byte of value 1. The change cipher specification message is sent by both the client and server to notify the receiving party that subsequent records will be protected under the just-negotiated CipherSpec and keys.

2.8. **Alert Protocol**

One of the content types supported by the SSL Record layer is the a..ieinJi1 convey the severity of the message and a description of the alert. Alert messages level of "fatal" result in the immediate termination of the connection. In this case, other connection corresponding to the session may continue, but the session identifier must be invalidated, preventing the failed session from being used to establish new connections. Like other messages, alert messages are encrypted and compressed, as specified by the current connection state.

2.9. Ciphers Used with SSL/TLS

The SSL/TLS protocol supports the use of a variety of different cryptographic algorithms, or ciphers, for use in operations such as authenticating the server and client to each other, transmitting certificates, and establishing session keys. Clients and servers may support different cipher suites, or sets of ciphers, depending on factors such as the version of SSL they support, company policies regarding acceptable encryption strength, and government restrictions on export of SSL enabled software. Among its other functions, the SSL handshake protocol determines how the server and client negotiate which cipher suites they will use to authenticate each other, to transmit certificates.

The cipher suite descriptions that follow refer to these algorithms:

1. **DES.** Data Encryption Standard, an encryption algorithm used by the U.S. Government. DES is a symmetric cryptosystem. When used for communication, both sender and receiver must know the same secret key, which is used both to encrypt and decrypt the message. DES can also be used for single-user encryption, such as to store files on a hard disk in encrypted form. In a multi-user environment, secure key distribution may be difficult; public-key cryptography provides an ideal solution to this problem. DES has a 64-bit block size and uses a 56-bit key during encryption. It is a 16-round Feistel cipher and was originally designed for implementation in hardware Ref [6].

..;. DES encryption includes the following components;

- Encıyption and decryption algorithms
- Matching shared secret keys on each peer
- Input cleartext data to be encrypted

2. **DSA.** Digital Signature Algorithm, part of the digital authentication standard used by the U.S. Government. DSA is based on the discrete logarithm problem and derives from cryptosystems proposed by Schnorr and ElGamal. It is for authentication only. In DSA, signature generation is faster than signature verification, whereas in ~~A signature verification is faster than signature generation (if the public and private exponents, respectively, are chosen for this property, which is the usual case).

3. **KEA.** Key Exchange Algorithm, an algorithm used for key exchange by the U.S. Government.

4. **MD5.** Message Digest algorithm. MD2, MD4 and MD5 are message-digest algorithms developed by Rivest. They are meant for digital signature applications where a large message has to be "compressed" in a secure manner before being signed with the private key. All three algorithms take a message of arbitraıy length and produce a 128-bit message digest. MD5 was developed by Rivest in 1991. It is basically MD4 with "safety-belts" and while it is slightly slower than MD4, it is more secure. The algorithm consists of four distinct rounds, which have a slightly different design from that of MD4. Message-digest size, as well as padding requirements, remains the same. *Den Boer* and *Bosselaers* have found pseudo-collisions for MD5, but there are no other known cryptanalytic results.

5. **RC2 and RC4.** Rivest encryption ciphers developed for RSA Data Security. RC2 is a variable key-size block cipher designed by *Rivest* for RSA Data Security. "RC" stands for "Ron's Code" or "Rivest's Cipher." It is faster than DES and is designed as a "drop-in" replacement for DES. It can be made more secure or less secure than DES against

exhaustive key search by using appropriate key sizes. It has a block size of 64 bits and is about two to three times raster than DES in software. The algorithm is confidential and proprietary to RSA Data Security. RC2 can be used in the same modes as DES. An agreement between the Software Publishers Association (SPA) and the United States government gives RC2 and RC4 special status by means of which the export approval process is simpler and quicker than the usual cryptographic export process. However, to qualify for quick export approval a product must limit the RC2 and RC4 key sizes to 40 bits; 56 bits is allowed for foreign subsidiaries and overseas offices of United States companies. An additional string (40 to 88 bits long) called a *salt* can be used to thwart attackers who try to precompute a large look-up table of possible encryptions. The salt is appended to the encryption key, and this lengthened key is used to encrypt the message; the salt is then sent, unencrypted, with the message. RC2 and RC4 have been widely used by developers who want to export their products; DES is almost never approved for export.

6. **RSA.** The pioneering paper by *Diffee* and *Hellman* introduced a new approach to cryptography and, in effect, challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems. One of the responses to the cha Secure Web Server provides security using a combination of the Secure Sockets Layer (SSL) protocol and (in most cases) CA-approved digital certificates. SSL handles the encrypted communications and the mutual authentication between browsers and Secure Web Server. The CA-approved digital certificate provides authentication for Secure Web Server (the CA puts its reputation behind its certification of our organization's identity).*Rivest, Shamir, and Adleman* developed it in 1977 Ref [2].

7. **RSA key exchange.** A key-exchange algorithm for SSL based on the RSA algorithm. Cipher suites supported by the SSL/TLS protocol that use the RSA key-exchange algorithm *strongest cipher suite:* This cipher suite is appropriate for banks and other institutions that handle highly sensitive data.

8. **SHA-1.** Secure Hash Algorithm, a hash function used by the U.S. Government. SHA-I was developed by NIST and is specified in the Secure Hash Standard (SHS, FIPS 180). SHA-1 was published in 1994. SHA-1 produces a 160-bit (20 byte) message digest. Although slower than MD5, this larger digest size makes it stronger against brute force attacks.

9. **SKIPJACK.** A classified symmetric-key algorithm implemented in FORTEZZA-compliant hardware used by the U.S. Government.

10. **Triple-DES.** DES applied three times. Key-exchange algorithms like KEA and RSA key exchange govern the way in which the server and client determine the symmetric keys they will both use during an SSL session. The most commonly used SSL cipher suites use RSA key exchange.

The SSL 2.0 and SSL 3.0 protocols support overlapping sets of cipher suites. Administrators can enable or disable any of the supported cipher suites for both clients and seıvers. When a particular client and server exchange information during the SSL handshake, they identify the strongest enabled cipher suites they have in common and use those for the SSL session.

Decisions about which cipher suites a particular organization decides to enable depend on trade-offs among the sensitivity of the data involved, the speed of the cipher, and the applicability of export rules.

Organizations want to disable the weaker ciphers to prevent SSL connections with weaker encryption. However, due to government restrictions on products that support anything stronger than 40-bit encryption, disabling support for all 40-bit ciphers effectively restricts access to network browsers that are available only in the United States (unless the server involved has a special Global Seıver ID that permits the international client to "step up" to stronger encryption).

To serve the largest possible range of users, it's administrators may wish to enable as broad a range of SSL cipher suites as possible. That way, when a domestic client or server is dealing with another domestic server or client, respectively, it will negotiate the use of the strongest ciphers available. And when an domestic client or server is dealing with an international server or client, it will negotiate the use of those ciphers that are permitted under U.S. export regulations.

2.10. **Summary**

In my discussion in this chapter I presented Cryptography and Web security which are provides our engineers with tools to implement the information protection requested or in other words; to achieve the security goals expect The collection of basic tools includes encryption algorithms, authentication codes one-way functions hashing functions, secret sharing schemes, signature schemes pseudorandom bit generators zero-knowledge proof systems, etc, For these elementary tools, it is possible to create more complex tools and services such as threshold encryption algorithms, authentication protocols, key establishment protocols, and a variety of application-oriented protocols, including electronic payment systems, electronic election, and electronic commerce protocols. Each tool is characterized by its security specification, which usually indicates the recommended configuration and its strength against specific threats, such as eavesdropping and illegal modification of information. The engineer can use all the tools provided by cryptography to combine them into a single solution.

3. TRANSPORT LAYER SECURITY

3.1. **Overview**

Internet commerce requires secure communications. To order goods, a customer typically sends credit card details. To order life insurance, the customer might have to supply confidential personal data. Internet users would like to know that such information is safe from eavesdropping or alteration.

Many Web browsers protect transmissions using the protocol SSL (Secure Sockets Layer). The client and server machines exchange nonces and compute session keys from them. The latest version of the protocol is called TLS (Transport Layer Security) it closely resembles SSL 3.0. Is TLS really secured? My proofs suggest that it is, I have analyzed a much simplified form ofTLS; I assume hashing and encryption to be secure.
TLS is the successor to the Secure Sockets Layer, and is nearly identical to SSL except that it implements.

- An open, standards-based solution,
- More non-proprietary ciphers,
- Better error reporting, and
- HM.AC digests instead ofl\,ID5.

TLS and SSL are not interoperable. The TLS protocol does contain a mechanism that allows TLS implementation to back down to SSL. The most recent browser versions support TLS. The TLS Working Group continues to work on the TLS protocol and related applications.

Internet browsers use security protocols to protect confidential messages. An inductive analysis of TLS, has been performed using the theorem prove Isabelle. Proofs are based on higher-order logic and make no assumptions concerning beliefs or finiteness. All the

obvious security goals can be proved; session resumption appears to be secure even if old session keys have been compromised. The proofs suggest minor changes to simplify the analysis.

TLS, even at an abstract level, is much more complicated than most protocols that researchers have verified. Session keys are negotiated rather than distributed, and the protocol has many optional parts. Nevertheless, the resources needed to verify TLS are modest: six man-weeks of effort and three minutes of processor time.

3.2. What Is TLS?

The really simple answer: it's SSL. TLS includes SSL. Think of it as "SSL Plus." Transport Layer Security (TLS) is a proposed IETF standard that supercedes Netscape's Secure Socket Layer (SSL).

As the name suggests, the purpose of the protocol is to provide encryption and certification at the transport layer (in this case, TCP), so that data can flow through a secure channel without requiring significant changes to the client and server applications.

One widespread use for this technology is in "HTTPS" or secures HTTP communications.

Macro byte's TLS provides client side tools for making secure HTTPS requests, and server side tools for running a secure web server. It also includes tools for creating and managing "keys," self-signed certificates, and "certificate signing requests" (for sending to companies like Verisign, Thawte, or Komodo, to receive a valid server certificate).

3.3. Comparison Between SSL, S-HITP and TLS

Actually as I have described in first chapter the World Wide Web (WWW) is a client-server technology, where information is available on servers and is accessed by the clients. The communication between these two parties is defined in the Hyper Text Transfer

Protocol (HTTP). With the increase in the use of this technology services such as entity authentication, data authentication, data confidentiality and non-repudiation are essential.

In this topics *Rajiv Papneja, Ravi Kiran Bhaskar, Sumeet Bhatia* discussed and compared three protocols that are used for providing security services for the WWW, but the thing which I analyzed from that three researchers written that SSL, S-HTTP and TLS as following:

- **SSL:** The Secure Sockets Layer (SSL) protocol, which was originally developed by Netscape, is a set of rules governing server authentication, client authentication, and encrypted communication between servers and clients. SSL *is* a type of sockets communication that runs above TCP/IP and below higher-level protocols such as HTTP or IMAP and requires no changes to the application layer. It uses the transport layer on behalf of the higher-layer protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, vice-versa enabling both the machines to establish an encrypted connection.

- **S-HTTP:** S-HTTP is a security-enhanced variant of HTTP proposed by Enterprise Integration Technologies (EIT) it is a message-based protocol that allows you to choose the security specifications you want for each transaction between the client and server. It adds the security enhancements directly to the application. S-HTTP provides the security through the protection mechanisms: Digital Signature, Message Authentication and Message Encryption.

- **TLS:** The Transport Layer Security (TLS) protocol, a product of the Internet Engineering Task Force (IETF), is a two-layer protocol that resides above the transport layer in the protocol stack. TLS provides two main security features: confidentiality and integrity. It is a protocol that is based on the SSL 3.0 protocol Ref [7].

The comparisons of the above stated Protocols that I analyzed based on the following two criteria's:

SSL	S-BTTP	TLS
SSL Record Protocol, SSL Handshake Protocol, URL Protocol	URL Protocol, Hypertext Markup Language (HTML) Protocol	TLS Record Protocol, TLS Handshake Protocol

Table 3.1. Protocols Used Ref [7].

SSL	TLS	S-HITP
RS~ FORTEZZ~ Triple-DES, RC4, DES, SK.lPJACK, MD5, SHA-I,	Triple DES, DES, RS~ DSS, RC4, MD5, SHA-I	RS~ MD5, DES, RC2, IDEA, PKCS-7

Table 3.2. Cryptographic Algorithms Supported

3.4. Wireless Transport Layer Security (WTLS)

Wireless Local Area Networks (WLANs) are expected to have a reassuring effect on e_cornmerce, but not until we have established that both the sender and receiver can be trusted and are who they say they are. It follows that there must also be a way to guarantee that the information has not been intercepted or modified in route. Wireless Transport

Layer Security (WTLS) was developed for this purpose as a replacement for the flawed WEP802. l Ib security. It works by performing client and server authentication to confirm the identity of the sender and his or her message.
It also encrypts the data in transit to keep the information secret and checks the integrity of the data after it arrives.

WTLS is a useful wireless security standard on the IETF's transport layer security (TLS) protocol, which in turn was expressly developed as an Internet standard version of Secure Socket Layer (SSL) Ref [8].

3.5. **Transport Layer Security**

TLS was introduced by the IETF to provide privacy and data integrity between two applications communicating over the Internet As a variant of SSL version 3, TLS offers several enhancements. SSL opens and closes a new socket for each message, whereas TLS establishes a secure connection, and can pass numerous messages through the same socket, providing much better throughput. TLS also provides greater separation between the handshake process and the record layer, allowing for implementation of new authentication methods in the future. *As* TLS was designed to minimize network activity, an optional session caching scheme was introduced to improve performance. Other differences between SSL and TLS are addressed in *table 3.3.*

3.6. **Advantages and Disadvantages of SSL/fLS**

SSL and TLS each provide a method of securing data to ensure authentication, confidentiality, and integrity, and an effective means of encrypting data to defend against tampering and spoofing. They were designed to operate in conjunction with other protocols, allowing for future protocols or changes to encryption schemes. TLS is also backwards compatible with earlier versions of SSL.

The effectiveness of SSL/TLS depends on the size of the encryption key. SSL has been cracked in less than a week using relatively short keys, such as 40- or 56-bit. 128-bit cipher keys are recommended for securing e-commerce or business applications.

SSL	TLS
• Minorversion0 • Uses a preliminaryHMACalgorithm • Does not applyMAC to version information • Doesnot specify a paddingvalue • Limitedset of alerts andwarning • Fortezzais an optionforkey exchange and encryption	• Minorversion1 • UsesHMAC • AppliesMac to versioninformation • Initializespaddingto a specific value • More alerts and warningsfor detailed informationan troubleshooting • Fortezzanot available

Table 3.3. Advantages and Disadvantages ofSSL/fLS

Sometimes SSL can be fooled during the handshake process. The protocol compatibility check is unprotected, and may be tampered with to institute a weaker encryption level.
An external drawback is that U.S. Jaw presently inhibits exportation of encryption keys by requiring a license for certain cryptographic products. In the interest of national security, this law is enforced to prohibit terrorist information from being exchanged using electronic means. It can also inhibit commercial transactions that otherwise would be considered reasonable and routine Ref [8].

Three common commercial of TLS are Netscape, Microsoft, and Entrust.

3.6.1. Netscape

Netscape Corporation created SSL to secure connections between Internet browsers and Internet servers. As SSL became an accepted standard, Netscape received the usual requests

to move SSL under a standard body. The new protocol was designated Transport Layer Security (TLS) and was submitted to the Internet Engineering Task Force (IETF's) TLS Working Group in early 1997. With SSL 3.0 as their starting point, the Working Group published TLS protocol 1.0 into the standards track. While Netscape browsers still support SSL, the company officially supports TLS as of Netscape Version 6.0.

3.6.2. Microsoft

Ever since SSL has emerged as a de facto standard, Internet Explorer has supported its own various versions. Microsoft also developed its own proprietary transport layer protocol, Private Communications Technology (PCT), to resolve problems associated with SSL v2.0. The company supported PCT in versions of Internet Explorer through 4.x. After SSL v3.0 was released, however, and the movement to make TLS an Internet standard was initiated, Microsoft dropped support for PCT in favor ofjoining forces with the TL_S working group. Some Microsoft products such as S-channel to support PCT for backwards compatibility only, but the company specifically discourage use of PCT for any future development.

3.6.3. Entrust

Entrust provides support for TLS vl .0 in its Entrust Toolkit for SSL/TLS. This toolkit lets developers incorporate SSL/TLS security into server-oriented C++ application. Developers can also use the TLS protocol to create secure a communications channel between a server and a client. Entrust has also implemented support for wireless solutions that use WTLS with WAP.

3.7. Pros and Cons of WTLS

Simply put, wireless networks provide less bandwidth, stability, and reliability than their wired counterparts. Latancy, or dropped packets, contributes to their inefficiency. In addition, wireless devices uniformly have Jess CPU power, memory, and battery Jife. Encryption techniques normally consume a great deal of CPU usage, memory, and bandwidth. All these limitations make wireless hard to secure.

WTLS has been instrumental in securing wireless devices such as cellular phones, PDAs, and laptops. It compensates for device shortcomings with respect to power and memory by minimizing protocol overhead, using better compression strategies and more efficient cryptographic algorithms. Specifically, WTLS uses RSA's RC5, or Elliptical Curve Cryptography (ECC) to encrypt data, and transport it over wireless links.
The protocol works in conjunction with PKI and wireless cookies to provide a wireless security solution. PK.I uses digital certificates to secure application platforms and browsers. Wireless cookies help maintain session management in the same manner as Internet cookies. The combination offers a fairly good wireless security solution.

Although it is not the whole solution, WTLS is essential to the security of wireless communications for the following reasons: it provides data integrity, pnvacy, authentication; and it allows developers to implement new technologies without compromising security functionality.

3.8. **WTLS Vulnerabilities**

As reasonably secure solution, WTLS still must be regarded as vulnerable, particularly in protocol translation. When a packet using WTLS moves through the air and hits the WAP gateway, it cannot traverse the wire until it has been converted to SSL or TLS. During conversion, there is a short period (fractions of a second) when the message is not encrypted, and therefore susceptible to eavesdropping or packet shifting. Critics view this as a major vulnerability. Some companies outsource this service to ensure that the WAP gateway is external.

Others downplay this vulnerability by pointing out that a hacker would have to have access to the carriers' gateway, have root access to that machine, and do a core dump at exactly the right millisecond. The argument for improbability is perhaps not sufficient, so Jong as the possibility exists unchallenged and the potential consequences are substantial.

WTLS also has some exposure in that wireless devices lack robust processing power. Methods of encryption must stay relatively low-level *to* accommodate the power and speed constraints imposed by devices. When cellular phones and PDA devices are able to use more advanced chip technology and increase battery and performance, the level of security will automatically increase. Today if a cellular phone has to encrypt and decrypt a 128-bit message, the user will notice a substantial degradation in performance, and may be unable to complete the transaction Ref [8).

3.9. Comparing the Wireless Application Protocol to the Wired TCP/IP Stack

When the WAP architecture is compared with architecture deployed in the wired environment, there are a number of items that do not logically map to the other, *Hudson* points out.

Wired	Wireless
• Application Protocols (IITTP, etc) • Application Environment (IITML, Javascript, Java) • Security Layer (SSL/TLS) • Transport Layer (TCP/IP)	• Application Environment (WAE) • Session Layer (WSP) • Transaction Layer (WTP) • Security Layer (WTLS) • Bearers: UDP SMS USSD CSD CDMA IS-136 CDPD PDC-P (etc)

Table 3.4. Wired Compared with Wireless Ref[9].

Hudson said, "Ironically, the security models are about the closest things to compare between the WAP and wired architectures," but I analyzed from the previous table that wired environment support for building applications (in the context of the Web) is a fairly

mix of strategies using HTML, but in the WAP standards, there is very little in common between the WAP and Internet models,

3.10. Comparing the Security Layers WTLS and SSL/fLS

WTLS as a protocol was closely modeled on the Internet standard TLS (Transport Layer Security). TLS is simply the next version of SSL, the de-facto security standard used today on the Internet. *(Deborah Durham, Kimberly Getgen).*

I noted the difference with WTLS is that some additional features are supported in the protocols, which are oriented toward the challenges of transporting data in a wireless network.

3.11. Comparing WTLS and SSL/fLS Authentication

Like SSL/fLS, in WTLS you can choose what level of authentication takes place between the client and the gateway. However, unlike SSLffLS, there are three authentication options supported within the WTLS/WAP model:

- Class one supports anonymous interactions between wireless clients and WAP gateway servers.
- Class two supports server authentication through the issuance of WTLS certificates to WAP gateway servers.
- Class three supports client and server authentication between wireless clients and the WAP gateway. This type of authentication is heading towards supporting Smartcards (GSM uses Smartcards called SIMs for storing account identification details and implementing the authentication features, so this is a logical extension of the model into the WTLS space). The WAP Forum has also produced a Wireless Identity Module

(WIM) specification that builds on RSA Laboratories' PKCS 15 and describes the use of smart cards for PKI-based client authentication.

There is quite a difference here in comparison with authentication in SSL/TLS. In the Internet model, while both server and client authentication are supported in the protocol, server authentication is the "default." And although anonymous and client authentication modes are supported, it is unusual to see them used Ref [9].

3.12. TLS Extension

Extensions that may be used to add functionality to TLS. It provides both generic extension mechanisms for the TLS handshake client and server hellos, and specific extensions using these generic mechanisms.

TLS is now used in an increasing variety of operational environments - many of which were not envisioned when the original design criteria for TLS were determined. The extensions introduced in this document are designed to enable TLS to operate as effectively as possible in new environments like wireless networks.

Wireless environments often suffer from a number of constraints not commonly present in wired environments - these constraints may include bandwidth limitations, computational power limitations, memory limitations, and battery life limitations.

The extensions described here focus on extending the functionality provided by the TLS protocol message formats. Other issues, such as the addition of new cipher suites, are deferred.

Specifically, the extensions described to:

- Allow TLS clients to provide to the TLS server the name of the server they are contacting. This functionality is desirable to facilitate secure connections to servers that host multiple virtual' servers at a single underlying network address.

- Allow TLS clients and servers to negotiate the maximum fragment length to be sent. This functionality is desirable as a result of memory constraints among some clients, and bandwidth constraints among some access networks.

- Allow TLS clients and servers to negotiate the use of client certificate URLs. This functionality is desirable in order to conserve memory on constrained clients.

- Allow TLS clients to indicate to TLS servers which CA root keys they possess. This functionality is desirable in order to prevent multiple handshake failures involving TLS clients that are only abJe to store a smalJ number of CA root keys due to memory limitations.

- Allow TLS clients and servers to negotiate the use of truncated MACs. This functionality is desirable in order to conserve bandwidth in constrained access networks.

- Allow TLS clients and servers to negotiate that the server sends the client certificate status information (e.g. an OCSP response) during a TLS handshake. This functionality is desirable in order to avoid sending a CRL over a constrained access network and therefore save bandwidth.

3.12.1. General Extension Mechanisms

This section presents general extension mechanisms for the TLS handshake client hello and server hello messages.

These general extension mechanisms are necessary in order to enable clients and servers to negotiate whether to use specific extensions, and how to use specific extensions. The extension formats described are based on (MAILINGUST).

3.12.1.1. Extended Client Hello

Clients MAY request extended functionality from servers by sending the extended client hello message format in place of the client hello message format.

The extended client hello message format is:

```
struct {
   ProtocolVersion client_version;
   Random random;
   Session.IDsession_id;
   CipherSuite cipher_suites<2..2A16-1>;
   CompressionMethod compression_methods<l ..2"8-1>;
   Extension client_he1lo_extension_list<0.2AI6-1>;
} ClientHello;
```

Here the new "client_hello_extension_list" field contains a list of extensions. The actual "Extension" format is defined in *Section 3.12.1.3.*

In the event that the server does not supply a client requests additional functionality using the extended client hello, and this functionality, the client MAY abort the handshake.

A server that supports the extensions mechanism MUST accept only client hello messages in either the original or extended Client Hello format, and (as for all other messages) MUST check that the amount of data in the message precisely matches one of these formats; if not then it MUST send a fatal "decode_error" alert. This overrides the "Forward compatibility note" in TLS.

3.12.1.2. Extended Server Hello

The extended server hello message format MAY be sent in place of the server hello message when the client has requested extended functionality via the extended client hello message specified in *Section 3.12.1.1.*

The extended server hello message format is:

```
struct {
  ProtocolV ersion server_version;
  Random random;
  SessionID session _id;
  CipherSuite cipher_suite;
  CompressionMethod compression_ method;
  Extension server_hello_extension_list<0 ..2"16-1>;
} ServerHello;
```

Here the new "server_hello_extension_list" field contains a list of extensions. The actual "Extension" format is defined in *Section 3.12.1.3.*

Note that the extended server helJo message is only sent in response to an extended client hello message. This prevents the possibility that the extended server hello message could "break" existing TLS l.0 clients.

3.12.1.3. Hello Extensions

The extension format for extended client hellos and extended server hellos is:

```
struct {
  ExtensionType extension_type;
  opaque extension_ data<O..2" 16-1>;
} Extension;
```

Here:

- "Extension_type" identifies the particular extension type.
- "Extension data" contains information specific to the particular extension type.

The extension types defined in this document are:

```
enum {
    server_ name(O), max_ fragment_length(l ),
    client_ certificate_ url(2 ), trusted_ ca_ keys(3 ),
    truncated_hmac(4), statüs_request(5), (65535)
} ExtensionType;
```

We note that for all extension types (including those defined in future), the extension type MUST NOT appear in the extended server hello *unless* the same extension type appeared in the corresponding client hello. Thus clients MUST abort the handshake if they receive an extension type in the extended server hello that they did not request in the associated (extended) client hello.

Nonetheless "server initiated" extensions may be provided in the future within this framework by requiring the client to first send an empty extension to indicate that it supports a particular extension.
Also note that when multiple extensions of different types are present in the extended client hello or the extended server hello, the extensions may appear in any order. Their MUST NOT is more than one extension of the same type.

Finally we note that all the extensions defined are relevant only when a session is initiated. However, a client that requests resumption of a session does not in general know whether the server will accept this request, and therefore it SHOULD send an extended client hello if it would normally do so for a new session.

If the resumption request is denied, then a new set of extensions will be negotiated as normal. If, on the other hand, the older session is resumed, then the server MUST ignore extensions appearing in the client hello, and send a server hello containing no extensions; in this case the extension functionality negotiated during the original session initiation is applied to the resumed session.

3.12.2. Extensions to the Handshake Protocol

Here I will suggest the use of two new handshake messages, "CertificateURL" and "CertificateStatus".The new handshake message structure therefore becomes:

```
enum {
    hello_request(O),client_hello(l), server_hello(2),
    certificate(l1), server_key_exchange (12),
    certificate_request(13), server_hello_done(! 4),
    certificate_verify(15), client_key_exchange(16),
    finished(20), certificate_ur1(21), certificate_status(22),
    (255)
} HandshakeType;
struct {
    HandshakeType msg_type;    /* handshake type*/
    uint24 length;             /* bytes in message*/
    select (HandshakeType) {
        case hello_request:      HelloRequest;
        case client_hello:       ClientHello;
        case server_hello:       ServerHello;
        case certificate:        Certificate;
        case server_key_exchange: ServerKeyExchange;
        case certificate_request: CertificateRequest;
        case server_hello_done:   ServerHelloDone;
```

```
        case certificate_ verify:  Certificate Verify;
        case client_ key_ exchange:  ClientKeyExchange;
        case finished:          Finished;
        case certificate _url:     CertificateURL;
        case certificate _status:  CertificateStatus;
    } body;
} Handshake;
```

If an application negotiates a server name using an application protocol, then upgrades to TLS, and a server_ name extension is sent, then the extension SHOULD contain the same name that was negotiated in the application protocol. If the server_name is established in the TLS session handshake, the client SHOULD NOT attempt to request a different server name at the application layer.

3.12.3. Maximum Fragment Length Negotiation

TLS specifies a fixed maximum plaintext fragment length of 2^14 bytes. It may be desirable for constrained clients to negotiate a smaller maximum fragment length due to memory limitations or bandwidth limitations.

In order to negotiate smaller maximum fragment lengths, clients MAY include an extension of type "max_fragment_length" in the (extended) client hello. The "extension_data" field of this extension SHALL contain:

```
enum{
    2^9(1 ), 2^10(2),  2^11(3), 2^12(4), (255)
} MaxFragmentLength;
```

Whose value is the desired maximum fragment length. The allowed values for this field are: 2"9, 2"10, 2"11, and 2"12.

Servers that receive an extended client hello containing a "max_fragment_length" extension MAY accept the requested maximum fragment length by including an extension of type "max_fragment_length" in the (extended) server hello. The "extension data" field of this extension SHALL contain "MaxFragmentLength" whose value is the same as the requested maximum fragment length.

If a server receives a maximum fragment length negotiation request for a value other than the allowed values, it MUST abort the handshake with an "illegal_parameter" alert. Similarly, if a client receives a maximum fragment length negotiation response that differs from the length it requested, it MUST also abort the handshake with an "illegal parameter" alert.

Once a maximum fragment length other than 2^14 has been successfully negotiated, the client and server MUST immediately begin fragmenting messages (including handshake messages), to ensure that no fragment larger than the negotiated length is sent Note that TLS already requires clients and servers to support fragmentation of handshake messages.

The negotiated length applies for the duration of the session including session resumptions.

3.12.4. Client Certificate URLs

TLS specifies that when client authentication is performed, clients send client certificates to servers during the TLS handshake. It may be desirable for constrained clients to send certificate URLs in place of certificates, so that they do not need to store there certificates and can therefore save memory.

In order to negotiate to send certificate URLs to a server, clients MAY include an extension of type "client_certificate_url" in the (extended) client hello. The "extension_data" field of this extension SHALL be empty.

(Note that it is necessary to negotiate use of client certificate URLs in order to avoid "breaking" existing TLS I.O servers.)

Servers that receive an extended client hello containing a "client _certificate _url" extension, MAY indicate that they are willing to accept certificate URLs by including an extension of type "client_certificate_url" in the (extended) server hello. The "extension data" field of this extension SHALL be empty.

After negotiation of the use of client certificate URLs has been successfully completed (by exchanging hellos including "client_certificate_url" extensions), clients MAY send a "Certificate URL" message in place of a "Certificate" message:

```
enum {
   individual_ certs(O), pkipath(l ), (255)
} CertChainType;

enum {
   false(O), true(l)
} Boolean;

struct {
   CertChain Type type;
   URLAndOptionalHash  url_and_hash_List<l ..2/\16-1>;
} CertificateURL;
struct {
   opaque url<l..2/\.16-1>;
   Boolean hash_present;
   select (hash_present) {
      case false: struct {} ;
      case true: SHAIHash;
   } hash;
} URLAndOptionalHash;
```

opaque SHA1Hash[20];

Here "url_and_hash_list" contains a sequence ofURLs and optional hashes.
When X.509 certificates are used, there are two possibilities:

- If CertificateURL.type is "individual_certs", each URL refers to a single DER-encoded X.509v3 certificate, with the URL for the client's certificate first, or
- If CertificateURL.type is "pkipath", the list contains a single URL referring to a DER encoded certificate chain, using the type PkiPath.
- When any other certificate format is used, the specification that describes use of that format in TLS should define the encoding format of certificates or certificate chains, and any constraint on their ordering.

The hash corresponding to each URL at the client's discretion is either not present or is the SHA-1 hash of the certificate or certificate chain (in the case of X.509 certificates, the DER-encoded certificate or the DER-encoded PkiPath).

Servers receiving "Certificate URL" SHALL attempt to retrieve the client's certificate chain from the URLs, and then process the certificate chain as usual. A cached copy of the content of any URL in the chain MAY be used, provided that a SHA-I hash is present for that URL and it matches the hash of the cached copy. Servers that support this extension MUST support the http: URL scheme for certificate URLs, and MAY support other schemes.

We note that clients may choose to send either "Certificate" or "CertificateURL" after successfully negotiating the option to send certificate URLs. The option to send a certificate is included to provide flexibility to clients possessing multiple certificates.

If a server encounters an unreasonable delay in obtaining certificates in a given CertificateURL, it SHOULD time out and signals a "certificate unobtainable" error alert.

3.12.5. Trusted CA Indication

Constrained clients that, due to memory limitations, possess only a small number of CA root keys may wish to indicate to servers which root keys they possess, in order to avoid repeated handshake failures.

In order to indicate which CA root keys they possess, clients MA include an extension of type "trusted_ca_keys" in the (extended) client hello. The "extension_data" field of this extension SHAL contains "Trusted Authorities" where:

```
struct {
   TrustedAuthority trusted_authorities_list<0..2"16-1>;
} Trusted.Authorities;

struct {
   IdentifierType identifier_type;
   select (identifier_type) {
      case pre_agreed: struct {};
      case key_shal_hash: SHAlHash;
      case x509_name: DistinguishedName;
      case cert_shal_hash: SHA1Hash;
   } identifier;
} TrustedAuthority;

enum {
   pre_agreed(O),key_shal_hash(l), x509_name(2),
   cert_shal_hash(3), (255)
```

} IdentifierType;

opaque DistinguishedName<1 .. 2^16-1>;

Here "TrustedAuthorities" provides a list of CA root key identifiers that the client possesses. Each CA root key is identified via either:

- "Pre_agreed" - no CA root key identity supplied.

- "Key_shaI_hash" - contains the SHA-I hash of the CA root key. For DSA and ECDSA keys, this is the hash of the "subjectPublicKey" value. For RSA keys, the hash is of the big-endian byte string representation of the modulus without any initial O-valued bytes. (This copies the key hash formats deployed in other environments.)

- "X509_name" - contains the DER-encoded X.509 Distinguished Name of the CA

- "Cert_shaI_hash" - contains the SHA-I hash of a DER-encoded Certificate containing the CA root key.

We note that clients may include none, some, or all of the CA root keys they possess in this extension. The option to include no CA root keys is included to allow the client to indicate possession of some pre-defined set of CA root keys.

Servers that receive a client hello containing the "trusted_ca_keys" extension, MAY use the information contained in the extension to guide their selection of an appropriate certificate chain to return to the client. In this event, the server SHALL include an extension of type "trusted_ca_keys" in the (extended) server hello. The "extension data" field of this extension SHALL be empty.

3.12.6. Truncated HMAC

Currently defined TLS cipher suites use the MAC construction HMAC with either MD5 or SHA-1 (HMAC) to authenticate record layer communications. In TLS the entire output of the hash function is used as the MAC tag. However it may be desirable in constrained environments to save bandwidth by truncating the output of the hash function to 80 bits when forming MAC tags.

In order to negotiate the use of 80-bit truncated HMAC, clients MAY include an extension of type "truncated_hmac" in the extended client hello. The "extension_data" field of this extension SHALL be empty.

Servers that receive an extended hello containing a "truncated_hmac" extension, MAY agree to use a truncated HMAC by including an extension of type "truncated_hmac", with empty "extension_data", in the extended server hello.

If HMAC truncation has been successfully negotiated during a TLS handshake, and the negotiated cipher suite uses HMAC, both the client and the server pass this fact to the TLS record layer along with the other negotiated security parameters. Subsequently during the session, clients and servers MUST use truncated HMACs, calculated as specified in (HMAC). That is, CipherSpec.hash _size is 10 bytes, and only the first 10 bytes of the HMAC output are transmitted and checked. Note that this extension does not affect the calculation of the PRF as part of handshaking or key derivation.

The negotiated HMAC truncation size applies for the duration of the session including session resumptions.

3.12.7. Certificate Status Request

Constrained clients may wish to use a certificate-status protocol such as OCSP to check the validity of server certificates, in order to avoid transmission of CRLs and therefore save

bandwidth on constrained networks. This extension allows for such information to be sent in the TLS handshake, saving roundtrips and resources.

In order to indicate their desire to receive certificate status information, clients MAY include an extension of type "Status request" in the (extended) client hello.

The "extension data" field of this extension SHALL contain "CertificateStatusRequest" where:

```
struct {
    CertificateStatus Type status_ type;
    select (status_type) {
        case ocsp: OCSPStatusRequest;
    } request;
} CertificateStatusRequest;

enum { ocsp(l ), (255) } CertificateStatusType;

struct {
    ResponderID responder _id_ list<O..2" 16-1>;
    Extensions request_extensions;
} OCSPStatusRequest;

opaque ResponderID<l ..2" 16-1>;
opaque Extensions<0 ..2/\16-1>;
```

In the OCSPStatusRequest, the "ResponderIDs" provides a list of OCSP responders that the client trusts. A zero-length "responder_id_list" sequence has the special meaning that the responders are implicitly known to the server - e.g. by prior arrangement. "Extensions" is a DER encoding of OCSP request extensions.

Both "ResponderID" and "Extensions" are DER-encoded ASN.1. "Extensions" is imported from (PK.IX). A zero- length "request extensions" value means that there are no extensions (as opposed to a zero-length ASN.1 SEQUENCE, which is not valid for the "Extensions" type).

In the case of the "id-pkix-ocsp-nonce" OCSP extension, (OCSP) is unclear about its encoding; for clarification, the nonce MUST be a DER-encoded OCTET STRING, which is encapsulated as another OCTET STRING (note that implementations based on an existing OCSP client will need to be checked for conformance to this requirement).

Servers that receive a client hello containing the "status request" extension MAY return a suitable certificate status response to the client along with their certificate. If OCSP is requested, they SHOULD use the information contained in the extension when selecting an OCSP responder, and SHOULD include request_extensions in the OCSP request

Servers return a certificate response along with their certificate by sending a "CertificateStatus" message immediately after the "Certificate" message (and before any "ServerKeyExchange" or "CertificateRequest" messages). If a server returns a "CertificateStatus" message, then the server MUST have included an extension of type "status_request" with empty "extension_data" in the extended server hello.

```
struct {
   CertificateStatusType status_type;
   select (status_type) {
      case ocsp: OCSPResponse;
   } response;
} CertificateStatus;
opaque OCSPResponse<l..2/\24-1>;
```

An "ocsp_response" contains a complete, DER-encoded OCSP response. Note that only one OCSP response may be sent

The "Certificate_Status" message is conveyed using the handshake message type "certificate_status". Note that a server MAY also choose not to send a "Certificate Status" message, even if it receives a "status_request" extension in the client hello message.

3.12.8. Procedure for Defining New Extensions

Traditionally for Internet protocols, the Internet Assigned Numbers Authority (IANA) handles the allocation of new values for future expansion, and RFCs usually define the procedure to be used by the IANA. However, there are subtle (and not so subtle) interactions that may occur in this protocol between new features and existing features, which may result in a significant reduction in overall security.

The following considerations should be taken into account when designing new extensions:

- All of the extensions defined in this document follow the convention that for each extension that a client requests and that the server understands, the server replies with an extension of the same type.

- Some cases where a server does not agree to an extension are error conditions, and some simply a refusal to support a particular feature. In general error alerts should be used for the former, and a field in the server extension response for the latter.

- Extensions should as far as possible be designed to prevent any attack that forces use (or non-use) of a particular feature by manipulation of handshake messages.

This principle should be followed regardless of whether the feature is believed to cause a security problem.

Often the fact that the extension fields are included in the inputs to the finished message hashes *will* be sufficient, but extreme care is needed when the extension changes the meaning of messages sent in the handshake phase.

Designers and implementers should be aware of the fact that until the handshake has been authenticated, active attackers can modify messages and insert, remove, or replace extensions.

- It would be technically possible to use extensions to change major aspects of the design of TLS; for example the design of cipher suite negotiation. This is not recommended; it would be more appropriate to define a new version of TLS - particularly since the TLS handshake algorithms have specific protection against version rollback attacks based on the version number, and the possibility of version rollback should be a significant consideration in any major design change Ref[19].

3.13. Summary

To know how to transfer information privately and securely across the Internet, the Transport Layer Security was explained. TLS was introduced by the IETF to provide privacy and data integrity between two applications communicating over the Internet

The Advantages and disadvantages of SSL/fLS has discussed and we noted that SSL and TLS each provide a method of securing data to ensure authentication, confidentiality. and integrity, and an effective means of encrypting data to defend against tampering and spoofing.

WTLS has been instrumental in securing wireless devices such as cellular phones, PDAs, and laptops. It compensates for device shortcomings with respect to power and memory by minimizing protocol overhead, using better compression strategies and more efficient cryptographic algorithms.
WTLS also has some exposure in that wireless devices lack robust processing power

Also I discussed the extension, which can be used to add functionality to TLS. It provides both generic extension mechanisms for the TLS handshake client and server hellos, and specific extensions using these generic mechanisms.

4. TRANSPORT LAYER SECURITY PROTOCOLS

4.1. Overview

A security protocol is a communications protocol that encrypts and decrypts a message for online transmission. Security protocols generally also provide authentication. Major security protocols that have emerged on the Web are Netscape's SSL and NCSA's HTTPS. Web browsers and servers generally support all the popular security protocols.

The Transport Layer Security Protocol (TLS) is an industry standard designed to protect the privacy of information communicated over the Internet. TLS assumes that a connection-oriented transport, typically TCP, is in use. The TLS protocol allows client/server applications to detect the following security risks:

- Message tampering
- Message interception
- Message forgery

4.2. TLS over Stream Control Transmission Protocol

TLS is designed to run on top of a byte-stream oriented transport protocol providing a reliable, in-sequence delivery. Thus, TLS is currently mainly being used on top of the Transmission Control Protocol (TCP).

Comparing TCP and SCTP, the latter provides additional features and this document shows how TLS should be used with SCTP to provide some of these additional features to the TLS user.

Here we define:

- How to use the multiple streams feature ofSCTP.
- How to handle the message oriented nature of SCTP.

It should be noted that the TLS user could take advantage of the multi-homing support of SCTP. The dynamic reconfiguration of IP-addresses, as currently being discussed, can also be used with the described solution.

The method described in this document does not require any changes ofTLS or SCTP. It is only required that SCTP implementations support the optional feature of fragmentation of SCTP user messages.

4.2.1. Fragmentation of User Messages

To avoid the knowledge and handling of the MTU inside TLS, SCTP MUST provides fragmentation of user messages, which is an optional feature. Since SCTP is a message-oriented protocol, *it* must be able to transmit all TLS records as SCTP user messages. Thus the supported maximum length of SCTP user messages MUST be at least 2\14 + 2048 + 5 = 18437 bytes, which is the maximum length of a TLS Ciphertext Ref [17].

4.2.2. TLS Requirements

4.2.2.1. Supported Ciphersuites

TLS implementations for TLS over SCTP MUST support at least the ciphersuite TLS_RSA_WITH_AES_128_CBC_SHA

4.2.3. Connections and Bi-directional Streams

TLS makes use of a bi-directional stream by establishing a connection over it. This means that the number of bi-directional streams limits the number of connections for an association.

The TLS handshake protocol is used on each bi-directional stream separately.
Each handshake can be:

- A full handshake or
- An abbreviated handshake that resumes a TLS session with a session id from another connection (on the same or another association).

After completing the handshake for a connection, the bi-directional stream can be used for TLS-based user data transmission. It should also be noted that the handshakes for the different connections are independent and can be delayed until the bi-directional stream is used for user data transmission.

4.2.4. Usage of Bi-directional Streams

It is not required that all bi-directional streams are used for TLS- based user data transmission. IfTLS is not used, it is called SCTP-based user data transmission.

SCTP-based user data transmission

If a bi-directional stream is not used for TLS-based communication there are no restrictions on the features provided by *SCTP* for SCTP- based user data transmission.

TLS-based user data transmission

In general, the bi-directional stream will be used for TLS-based user data transmission and it SHOULD NOT be used for SCTP-based user data transmission. The exception to this rule is for protocols, which contain upgrade-to-Tl.S mechanisms.

TLS requires that the underlying transport deliver TLS records in strict sequence. Thus, the 'unordered delivery' feature of *SCTP MUST NOT* is used on streams, which are used for TLS based user data transmission. For the same reason, TLS records delivered to SCTP for transmission MUST NOT have limited lifetimes.

4.2.5. Usage of Uni-directional Streams

The uni-directional streams cannot be used for TLS-based user data transmission. Nevertheless, they can be used without any restrictions for SCTP-based communication.

Just after the association has been established, the client sends two ClientHello messages on the bi-directional streams 0 and 1. After a full handshake has been completed on each bi-directional stream, TLS-based user data transmission can take place on that stream. It *is* possible that on the bi-directional stream 0, the handshake has been completed, and user data transmission is ongoing, while on the bi-directional stream 1, the handshake has not been completed, or vice versa.

4.2.6. Security Considerations ofTLS over SCTP

It is possible to authenticate TLS endpoints based on IP-addresses in certificates. Unlike TCP, SCTP associations can use multiple addresses per SCTP endpoint. Therefore it is possible that TLS records will be sent from a different IP-address than that originally authenticated. This is not a problem provided that no security decisions are made based on that IP-address. This is a special case of a general rule: all decisions should be based on the peer's authenticated identity, not on its transport layer identity.

4.3. Organization of TLS

To use TLS for client/server communication, the following steps are involved:

1. Handshake and cipher suite negotiation
2. Authentication of parties
3. Key-related information exchange
4. Application data exchange

4.2.5. Usage of Uni-directional Streams

The uni-directional streams cannot be used for TLS-based user data transmission. Nevertheless, they can be used without any restrictions for SCTP-based communication.

Just after the association has been established, the client sends two ClientHello messages on the bi-directional streams 0 and 1. After a full handshake has been completed on each bi-directional stream, TLS-based user data transmission can take place on that stream. It is possible that on the bi-directional stream 0, the handshake has been completed, and user data transmission is ongoing, while on the bi-directional stream 1, the handshake has not been completed, or vice versa.

4.2.6. Security Considerations ofTLS over SCTP

It is possible to authenticate TLS endpoints based on IP-addresses in certificates. Unlike TCP, SCTP associations can use multiple addresses per SCTP endpoint.. Therefore it is possible that TLS records will be sent from a different IP-address than that originally authenticated. This is not a problem provided that no security decisions are made based on that IP-address. This is a special case of a general rule: all decisions should be based on the peer's authenticated identity, not on its transport layer identity.

4.3. Organization ofTLS

To use TLS for client/server communication, the following steps are involved:

1. Handshake and cipher suite negotiation
2. Authentication of parties
3. Key-related information exchange
4. Application data exchange

The steps that make up TLS are divided into two protocols that, together, provide connection security:

- TLS Handshake Protocol
- TLS Record Protocol

4.3.1. TLS Handshake Protocol

The *Transport Layer Security* (TLS) Handshake Protocol is responsible for the authentication and key exchange necessary to establish or resume secure sessions. When establishing a secure *session,* the Handshake Protocol manages the following:

- Cipher suite negotiation
- Authentication of the server and optionally, the client
- Session key information exchange.

A TLS handshake involves a client, such as a World Wide Web browser, and a Web server. Below, I refer to the client as *A* ('Alice') and the server as *B* ('Bob'), as is customary for authentication protocols, especially since *C* and *S* often have dedicated meanings in the literature.

At the start of a handshake, *A* contacts *B,* supplying a session identifier and nonce. In response, *B* sends another nonce and his public-key certificate (my *model* omits other possibilities). Then *A* generates a *pre-master-secret,* a 48-byte random string, and sends it to *B* encrypted with his public key. *A* optionally sends a signed message to authenticate herself Now, both parties calculate the *master-secret M* from the nonces and the pre-master-secret, using a secure pseudo-random-number function (PRF). They calculate session keys from the nonces and master-secret.

Each session involves a pair of symmetric keys; A encrypts using one and B encrypts using the other. Before sending application data, both parties exchange finished messages to confirm all details of the handshake and to check that clear text parts of messages have not been altered.

A full handshake is not always necessary. At some later time, A can resume a session by quotiı:ıgan old session identifier along with a fresh nonce. If B is willing to resume the designated session, then he replies with a fresh nonce. Both parties compute fresh session keys from these nonces and the stored master-secret, M. Both sides confirm this shorter run using finished messages.

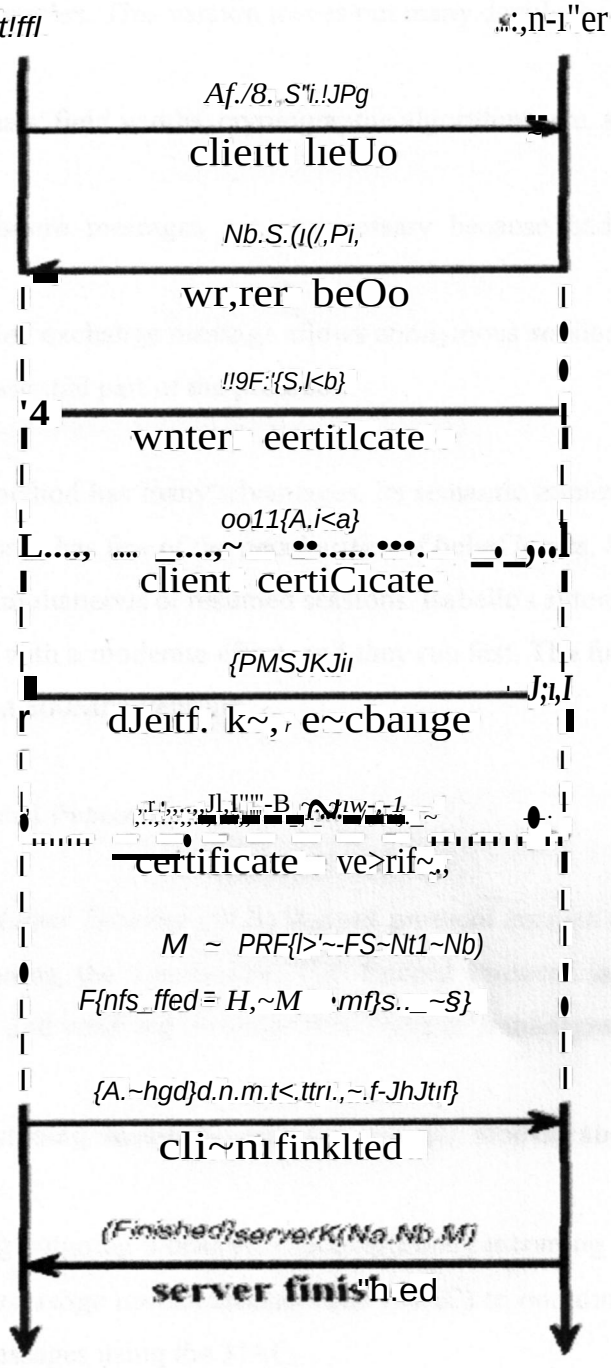


Figure 4.1. TI.$ Handshake Protocol as Modeled

TLS is highly complex. This version leaves out many details for the sake of simplicity:

1. Record formats; field widths, cryptographic algorithms, etc. are irrelevant in an abstract analysis.
2. Alert and failure messages are unnecessary because bad sessions can simply be abandoned.
3. The server key exchange message aJlows anonymous sessions among other things, but it is not an essential part of the protocol.

The inductive method has many advantages. Its semantic framework, based on the actions agents can perform, has few of the peculiarities of belieflogics. Proofs impose no limits on the number of simultaneous or resumed sessions. Isabelle's automatic tools allow the proofs to be generated with a moderate effort, and they run fast. The full TLS proof script runs in 150 seconds on a 300Mhz Pentium.

4.3.2. TLS Record Protocol

The *Transport Layer Security* (TLS) Record protocol secures application data using the keys created during the Handshake. The Record Protocol is responsible for securing application data and verifying its *integrity* and origin. It manages the foJJowing:

- Dividing outgoing messages into manageable blocks, and reassembling incoming messages.
- Compressing outgoing blocks and decompressing incoming blocks (optional).
- Applying a *message authentication code* (MAC) to outgoing messages, and verifying incoming messages using the MAC.
- Encrypting outgoing messages and decrypting incoming messages.
- When the Record Protocol is complete, the outgoing encrypted data is passed down to the Transmission Control Protocol (TCP) layer for transport.

The TLS Protocol is a layered protocol. The TLS Record Protocol is the lowest layer. It provides the low-level message fragmentation, encryption/decryption, etc. Layered on top of this protocol are four record protocol clients: the handshake protocol, the alert protocol, the change cipher spec protocol, and the application data protocol. The delegation protocol is a fifth record protocol client.

4.4. Lightweight Directory Access Protocol (v3):

Extension for Transport Layer Security

Here we define the "Start Transport Layer Security (TLS) Operation" for LDAP [LDAPv3, TLS]. This operation provides for TLS establishment in an LDAP association and İs defined in terms of an LDAP extended request.

4.4.1. The Start TLS Request

This section describes the Start TLS extended request and extended response themselves: how to form the request, the form of the response, and enumerates the various result codes the client MUST be prepared to handle. The section following this one then describes how to sequence an overall Start TLS Operation.

Requesting TLS Establishment

A client may perform a Start TLS operation by transmitting an LDAP PDU containing an ExtendedRequest [LDAPv3] specifying the OID for the
Start TLS operation:

1.3.6.1.4.1.1466.20037

An LDAP ExtendedRequest is defined as follows:

```
ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
        requestName      [0] LDAPOID,
        requestValue     [1] OCTET STRING OPTIONAL}
```

Setting the requestName field to the OID string given above forms a Start TLS extended request. The requestValue field is absent. The client MUST NOT sends any PDUs on this connection following this request until it receives a Start TLS extended response.
When a Start TLS extended request is made, the server MUST return anLOAP POU containing a Start TLS extended response. An LOAP Extended Response is defined as follows:

```
ExtendedResponse ::= [APPLICATION24] SEQUENCE {
        COMPONENTS OF LOAPResult,
        responseName     [1 0] LDAPOID OPTIONAL,
        response         (1 1] OCTET STRING OPTIONAL }
```

A Start TLS extended response MUST contain a responseName field, which MUST be set to the same string as that in the responseName field present in the Start TLS extended request. The response field is absent. The server MUST set the resultCode field to either success or one of the other values outlined in section 4.4.3.

4.4.2. "Success" Response

If the ExtendedResponse contains a resultCode of success, this indicates that the server is willing and able to negotiate *TLS.*

4.4.3. Response other than "Success"

If the ExtendedResponse contains a resultCode other than success, this indicates that the server is unwilling or unable to negotiate TLS.
If the Start TLS extended request was not successful, the resultCode will be one of
OperationsError (operations sequencing incorrect; e.g. TLS already established)
Protocol Error (TLS not supported or incorrect POU structure)
Referral (this server doesn't do *TLS,* try this one)
Unavailable (e.g. some major problem with TLS or server is shutting down)

The server MUST return operations Error if the client violates any of the Start TLS extended operation-sequencing requirements.

4.4.4. Sequencing of the StartTLS Operation

Here we describe the overall procedures clients and servers MUST follow for TLS establishment. These procedures take into consideration various aspects of the overall security of the LDAP association including discovery of resultant security level and assertion of the client's authorization identity.

We have to note that the precise effects, on a client's authorization identity, of establishing TLS on an LDAP association.

4.4.4.1. Requesting to Start TLS on an LDAP Association

The client MAY send the Start TLS extended request at any time after establishing an LDAP association, except that in the following cases:

The client MUST NOT send a Start TLS extended request:

- IfTLS is currently established on the connection, or
- During a multi-stage SASL negotiation, or
- If there are any LDAP operations outstanding on the connection.
- The result of violating any of these requirements is a resultCode of operationsError, as described above in section 4.4.3.

The client MAY have already performed a Bind operation when it sends a Start TLS request, or the client might have not yet bound.

If the client did not establish a TLS connection before sending any other requests, and the server requires the client to establish a TLS connection before performing a particular request, the server MUST reject that request with a confidentialityRequired or

strongAuthReq uired result. The client MAY send a Start TLS extended request, or it MAY choose to close the connection.

4.4.4.2. Starting TLS

The server will return an extended response with the resultCode of success if it is willing and able to negotiate TLS. It will return other resultCodes, documented above, if it is unable.

In the successful case, the client, which has ceased to transfer LDAP requests on the connection, MUST either begin a TLS negotiation or close the connection. The client will send PDUs in the TLS Record Protocol directly over the underlying transport connection to the server to initiate TLS negotiation

4.4.4.3. TLS Version Negotiation

Negotiating the version ofTLS or SSL to be used is a part of the TLS Handshake Protocol.

4.4.4.4. Discovery of Resultant Security Level

After a TLS connection is established on an LDAP association, both parties MUST individually decide whether or not to continue based on the privacy level achieved Ascertaining the TLS connection's privacy level is implementation dependent and accomplished by communicating with one's respective local TLS implementation.

If the client or server decides that the level of authentication or privacy is not high enough for it to continue, it SHOULD gracefully close the TLS connection immediately after the TLS negotiation has completed.

The client MAY attempt to Start TLS again, or MAY send an unbind request, or send any other LDAP request

4.4.4.5. Assertion of Client's Authorization Identity

The client MAY, upon receipt of a Start TLS extended response indicating success, assert that a specific authorization identity be utilized in determining the client's authorization status. The client accomplishes this via an LDAP Bind request specifying a SASL mechanism of "EXTERNAL" SASL.

4.4.4.6. Server Identity Check

The client MUST check its understanding of the server's hostname against the server's identity as presented in the server's Certificate message, in order to prevent man-in-the-middle attacks.

Matching is performed according to these rules:

- The client MUST use the server hostname it used to open the LDAP connection as the value to compare against the server name as expressed in the server's certificate. The cJient MUST NOT use the server's canonical DNS name or any other derived form of name.
- If a subjectAltName extension of type dNSName is present in the certificate, it SHOULD be used as the source of the server's identity.
- Matching is case-insensitive.
- The "*" wildcard character is allowed. If present, it applies only to the left-most name component.

If the hostname does not match the dNSName-based identity in the certificate per the above check, user-oriented clients SHOULD either notify the user (clients MAY give the user the opportunity to continue with the connection in any case) or terminate the connection and indicate that the server's identity is suspect. Automated clients SHOULD close the connection, returning and/or logging an error indicating that the server's identity is suspect.

Beyond the server identity checks described in this section, clients SHOULD be prepared to do further checking to ensure that the server is authorized to provide the service it is observed to provide. The client MAY need to make use of local policy information Ref (18].

4.4.4. 7. Refresh of Server Capabilities Information

The client MUST refresh any cached server capabilities information (e.g. from the server's root DSE). This is necessary to protect against active-intermediary attacks, which may have altered any server capabilities information retrieved prior to TLS establishment The server MAY advertise different capabilities after TLS establishment Ref [14].

4.4.5. Closing a TLS Connection

4.4.5.1. Graceful Closure

Either the client or server MAY terminate the TLS connection on an LDAP association by sending a TLS closure alert. This will leave the LDAP association intact.

Before closing a TLS connection, the client MUST either wait for any outstanding LDAP operations to complete, or explicitly abandon them LDAPv3.

After the initiator of a close has sent a closure alert, it MUST discard any TLS messages until it has received an alert from the other party. It will cease to send TLS Record Protocol PDUs, and following the receipt of the alert, MAY send and receive LDAP PDUs.

The other party, if it receives a closure alert, MUST immediately transmit a TLS closure alert. It will subsequently cease to send TLS Record Protocol PDUs, and MAY send and receive LDAP PDUs.

4.4.5.2. Abrupt Closure

Either the client or server MAY abruptly close the entire LDAP association and any TLS connection established on it by dropping the underlying TCP connection.

4.4.6. Effects of TLS on a Client's Authorization Identity

This section describes the effects on a client's authorization identity brought about by establishing TLS on an LDAP association. The default effects are described first, and next the facilities for client assertion of authorization identity are discussed, including error conditions. Lastly, the effects of closing the TLS connection are described.

4.4.6.1. TLS Connection Establishment Effects

- Default Effects

Upon establishment of the TLS connection onto the LDAP association, any previously established authentication and authorization identities MUST remain in force, including anonymous state. This holds even in the case where the server requests client authentication via TLS -- e.g. requests the client to supply its certificate during TLS negotiation.

- Client Assertion of Authorization Identity

A client MAY either implicitly request that its LDAP authorization identity be derived from its authenticated TLS credentials or it MAY explicitly provide an authorization identity and assert that it be used *in* combination *with* its authenticated TLS credentials.

The former is known as an implicit assertion, and the latter as an explicit assertion.

- Implicit Assertion

An implicit authorization identity assertion is accomplished after TLS establishment by invoking a Bind request of the SASL form using the "EXTERNAL" mechanism name.

(SASL, LDAPv3) that SHALL NOT include the optional credentials octet string (found within the SaslCredentials sequence in the Bind Request). The server will derive the client's authorization identity from the authentication identity supplied in the client's TLS credentials (typically a public key certificate) according to local policy. The underlying mechanics of how this is accomplished are implementation specific.

- Explicit Assertion

An explicit authorization identity assertion is accomplished after TLS establishment by invoking a Bind request of the SASL form using the "EXTERNAL" mechanism name (SASL, LDAPv3) that SHALL incJude the credentials octet string.

- Error Conditions

For either form of assertion, the server MUST verify that the client's authentication identity as supplied in its TLS credentials is permitted to be mapped to the asserted authorization identity. The server MUST reject the Bind operation with an invalidCredentials resuJtCode in the Bind response if the client is not so authorized.

4.4.6.2. TLS Connection Closure Effects

Closure of the TLS connection MUST cause the LDAP association to move to an anonymous authentication and authorization state regardless of the state established over TLS and regardless of the authentication and authorization state prior to TLS connection establishment.

4.4.7. Security Considerations to LDAP

The goals of using the TLS protocol with LDAP are to ensure connection confidentiality and integrity, and to optionally provide for authentication. TLS expressly provides these capabilities.

All security gained via use of the Start TLS operation is gained by the use of TLS itself The Start TLS operation, on its own, does not provide any additional security.

The use of TLS does not provide or ensure for confidentiality and/or non-repudiation of the data housed by an LDAP-based directory server. The level of security provided though the use of TLS depends directly on both the quality of the TLS implementation used and the style of usage of that implementation.

4.5. SMTP Service Extensions for Secure SMTP over TLS

Here we describe an extension to the SMTP (Simple Mail Transfer Protocol) service that allows an SMTP server and client to use TLS (Transport Layer Security) to provide private, authenticated communication over the Internet. This gives SMTP agents the ability to protect some or all of their communications from eavesdroppers and attackers.

SMTP servers and clients normally communicate in the clear over the Internet. In many cases, this communication goes through one or more router that is not controlled or trusted by either entity. Such an untrusted router might allow a third party to monitor or aJter the communications between the server and client.

Further, there is often a desire for two SMTP agents to be able to authenticate each other's identities. For example, a secure SMTP server might only allow communications from other SMTP agents it knows, or it might act differently for messages received from an agent it knows than from one it doesn't know.

TLS more commonly known as SSL, is a popular mechanism for enhancing TCP communications with privacy and authentication. TLS is in wide use with the HTTP protocol, and İs aJso being used for adding security to many other common protocols that run over TCP.

4.5.1. Security Considerations of SMTP over TLS

It should be noted that SMTP is not an end-to-end mechanism. Thus, if an SMTP client/server pair decides to add TLS privacy, they are not securing the transport from the originating mail user agent to the recipient. Further, because delivery of a single piece of mail may go between more than two SMTP servers, adding TLS privacy to one pair of servers does not mean that the entire SMTP chain has been made private. Further, just because an SMTP server can authenticate an SMTP client, it does not mean that the SMTP client authenticated the mail from the SMTP client when the client received it.

Both the SMTP client and server must check the result of the TLS negotiation to see whether an acceptable degree of authentication and privacy was achieved. Ignoring this step completely invalidates using TLS for security. The decision about whether acceptable authentication or privacy was achieved is made locally, is implementation-dependent, and is beyond the scope of this document.

The SMTP client and server should note carefully the result of the TLS negotiation. If the negotiation results in no privacy, or if it results in privacy using algorithms or key lengths that are deemed not strong enough, or if the authentication is not good enough for either party, the client may choose to end the SMTP session with an immediate QUIT command, or the server may choose to not accept any more SMTP commands.

Deleting the "250 STARTTLS" response from the server can launch a man-in-the-middle attack. This would cause the client not to try to start a TLS session. Another man-in-the-middle attack is to allow the server to announce its STARTTLS capability, but to alter the client's request to start TLS and the server's response. In order to defend against such attacks both clients and servers MUST be able to be configured to require successful TLS negotiation of an appropriate cipher suite for selected hosts before messages can be successfully transferred. The additional option of using TLS when possible SHOULD also be provided. An implementation MAY provide the ability to record that TLS was used in

communicating with a given peer and generating a warning if it is not used in a later session.

If the TLS negotiation fails or if the client receives a 454 response, the client has to decide what to do next. There are three main choices: go ahead with the rest of the SMfP session, retiy TLS at a later time, or give up and return the mail to the sender. If a failure or error occurs, the client can assume that the server may be able to negotiate TLS in the future, and should tiy negotiating TLS in a later session, until some locally-chosen timeout occurs, at which point, the client should return the mail to the sender. However, if the client and server were only using TLS for authentication, the client may want to proceed with the SMTP session, in case the server accepts some of the operations the client wanted to perform even if the client is unauthenticated.

Before the TLS handshake has begun, any protocol interactions are performed in the clear and may be modified by an active attacker.

4.6. Using TLS with IMAP, POP3 and ACAP

The TLS protocol (formerly known as SSL) provides a way to secure an application protocol from tampering and eavesdropping. The option of using such security is desirable for IMAP, POP and ACAP due to common connection eavesdropping and hijacking attacks. Although advanced SASL authentication mechanisms can provide a lightweight version of this service, TLS is complimentary to simple authentication-only SASL mechanisms or deployed clear-text password login commands.

Many sites have a high investment in authentication infrastructure (e.g., a large database of a one-way-function applied to user passwords), so a privacy layer which is not tightly bound to user authentication can protect against network eavesdropping attacks without requiring a new authentication infrastructure and/or forcing all users to change their password. Recognizing that such sites will desire simple password authentication in combination with TLS encryption, this specification defines the PLAIN SASL mechanism

for use with protocols, which lack a simple password authentication command such as ACAP and SMTP.

There is a strong desire in the IETF to eliminate the transmission of clear-text passwords over unencıypted channels. While SASL can be used for this purpose, TLS provides an additional tool with different deployability characteristics. A server supporting both TLS with simple passwords and a challenge/response SASL mechanism is likely to interoperate with a wide variety of clients without resorting to unencrypted clear-text passwords.

The STARTTLS command rectifies a number of the problems with using a separate port for a "secure" protocol variant.

4.6.1. Using **TLS Considerations**

TLS only provides protection for data sent over a network connection. Messages transferred over lMAP or POP3 are still available to server administrators and usually subject to eavesdropping, tampering and forgery when transmitted through SMTP or NNTP. TLS is no substitute for an end-to-end message security mechanism using MIME security multiparts.

A man-in-the-middle attacker can remove STARTTLS from the capability list or generate a failure response to the STARTTLS command. In order to detect such an attack, clients SHOULD warn the user when session privacy is not active and/or be configurable to refuse to proceed without an acceptable level of security.

A man-in-the-middle attacker can always cause a down-negotiation to the weakest authentication mechanism or cipher suite available. For this reason, implementations SHOULD be configurable to refuse weak mechanisms or cipher suites.
Any protocol interactions prior to the TLS handshake are performed in the clear and can be modified by a man-in-the-middle attacker. For this reason, clients MUST discard cached information about server capabilities advertised prior to the start of the TLS handshake.

Clients are encouraged to *clearly* indicate when the *level* of encryption active is known to *be* vulnerable *to* attack using modem hardware (such as encryption keys *with 56* bits of entropy or less).

The LOG.INDISABLEDIMAP capability *only* reduces the potential for passive attacks; it provides no protection against active attacks. The responsibility remains with the client to avoid sending a password over a vuJnerabJechanneJ.

The PLAIN mechanism relies on the TLS encryption layer for security. When used without TLS, it is vulnerable to a common network eavesdropping attack Therefore PLAIN MUST NOT be advertised or used unless a suitable TLS encryption layer is active or backwards compatibility dictates otherwise. When the PLAIN mechanism is used, the server gains the ability to impersonate the user to a11 services with the same password regardless of any encryption provided by TLS or other network privacy mechanisms. While many other authentication mechanisms have similar weaknesses, stronger SASL mechanisms such as Kerberos address this issue. Clients are encouraged to have an operational mode where all mechanisms, which are likely to reveal the user's password to the server, are disabled.

4.7. HTTP Upgrade to TLS

Upgrading mechanism in HTTP/1.1 to initiate Transport Layer Security (TLS) over an existing TCP connection. This allows unsecured and secured HTTP traffic to share the same we11 known port (in this case, http: at 80 rather than https: at 443). It also enables "virtual hosting", so a single HTTP + TLS server can disambiguate traffic intended for several hostnames at a single IP address.

Since HTTP/I. I de.fines Upgrade as a hop-by-hop mechanism, also there is *HTTP* CONNECT method for establishing end-to-end tunnels across HTTP proxies. Finally, this memo establishes new IANA registries for public HTTP status codes, as well as public or private Upgrade product tokens.

TLS and SSL (Secure Sockets Layer), establishes a private end-to-end connection, optionally including strong mutual authentication, using a variety of cryptosystems. Initially, a handshake phase uses three subprotocols to set up a record layer, authenticate endpoints, set parameters, as well as report errors. Then, there is an ongoing-layered record protocol that handles encryption, compression, and reassembly for the remainder of the connection. The latter is intended to be completely transparent For example, there is no dependency between TLS's record markers and or certificates and HTTP/I. 1's chunked encoding or authentication.

Either the client or server can use the HTTP/I. I Upgrade mechanism to indicate that a TLS-secured connection is desired or necessaıy Ref [13J.

4.8. HfTP over TLS

HTTP was originally used in the clear on the Internet. However, increased use of HTTP for sensitive applications has required security measures. SSL and its successor TLS were designed to provide channel-oriented security.

Conceptually, HTTP/TLS is very simple. Simply use HTTP over TLS precisely as you would use HTTP over TCP Ref [13].

4.8.1. Connection Initiation

The agent acting as the HTTP client should also act as the TLS client. It should initiate a connection to the server on the appropriate port and then send the TLS ClientHello to begin the TLS handshake. When the TLS handshake has finished. The client may then initiate the first HTTP request. All HTTP data MUST be sent as TLS "application data". Normal HTTP behavior, including retained connections should be followed.

4.8.2. Connection Closure

TLS provides a facility for secure connection closure. When a valid closure alert is received, an implementation can be assured that no further data will be received on that

connection. TLS implementations MUST initiate an exchange of closure alerts before closing a connection. A TLS implementation MAY, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an "incomplete close". Note that an implementation, which does this, MAY choose to reuse the session. This SHOULD only be done when the application knows (typically through detecting HTTP message boundaries) that it has received all the message data that it cares about.

- **Client Behavior**

Because HTTP uses connection closure to signal end of server data, client implementations MUST treat any premature closes as errors and the data received as potentially truncated. While in some cases the HTTP protocol allows the client to find out whether truncation took place so that, if it received the complete reply, it may tolerate such errors following the principle to be strict when sending and tolerant when receiving", often truncation does not show in the HTTP protocol data; two cases in particular deserve special note: A HTTP response without a Content-Length header. Since data length in this situation is signaled by connection close a premature close generated by the server cannot be distinguished from a spurious close generated by an attacker.

A HTTP response with a valid Content-Length header closed before all data has been read. Because TLS does not provide document-oriented protection, it is impossible to determine whether the server bas miscomputed the Content-Length or an attacker has truncated the connection.

There is one exception to the above rule. When encountering a premature close, a client SHOULD treat as completed all requests fur which it has received as much data as specified in the Content-Length header.

A client detecting an incomplete close SHOULD recover gracefully. It MAY resume a TLS session closed in this fashion. Clients MUST send a closure alert before closing the connection. Clients, which are unprepared to receive any more data, MAY choose not to

wait for the server's closure alert and simply close the connection, thus generating an incomplete close on the server side.

- Server Behavior

In particular, servers SHOULD be prepared to receive an incomplde close from the client, since the client can often determine when the end of server data is. Servers SHOULD be willing to resume TLS sessions closed in this fashion.

Implementation note: In HTTP implementations, which do not use pa[illegible]1 connections, the server ordinarily expects to be able to signal end of data by d05111K the connection. When Content-Length is used, however, the client may have already sent the closure alert and dropped the connection.

Servers MUST attempt to initiate an exchange of closure alerts \\ith the client before closing the connection. Servers MAY close the connection after sendin: the closure alert, thus generating an incomplete close on the client side Ref [20].

4.8.3. Port Number

The first data that an HTTP server expects to receive from the client is the [illegible] production. The first data that a TLS server (and hence an HTTP/TLS server) expects to receive is the ClientHello.

Consequently, common practice has been to run HTTP/TLS over a separate port in order to distinguish which protocol is being used. When *HTTPITLS* is being run over a TCHP connection, the default port is 443. This does not preclude HTTP/fLS from being run over another transport. TLS only presumes a reliable connection-oriented data stream.

4.8.4. URI Format

HTTPfilS is differentiated from **HTTP** URis by using the 'https' protocol identifier in place of the 'http' protocol identifier Ref [13].

4.9. Advanced Encryption Standard (AES) Ciphersuites for (TLS)

At present, the symmetric ciphers supported by TLS are RC2, RC4, IDEA, DES, and triple DES. The protocol would be enhanced by the addition of AES ciphersuites, for the following reasons:

1. RC2, RC4, and IDEA are all subject to intellectual property claims. RSA Security Inc. has trademark rights in the names RC2 and RC4, and claims that the RC4 algorithm itself *is* a trade secret Ascom Systec Ltd. owns a patent *on* the IDEA algorithm.
2 Triple DES is much less efficient than more modem ciphers.
3 Now that the AES process is completed there will be commercial pressure to use the selected cipher. The AES is efficient and has withstood extensive cryptanalytic efforts. The AES is therefore a desirable choice.

Currently the DHE ciphersuites only allow triple DES (along with some "export" variants which do not use a satisfactory key length). At the same time the DHE ciphersuites are the only ones to offer forward secrecy.

4.9.1. (AES) Ciphersuites for (TLS) Considerations

It is not believed that the new ciphersuites are ever less secured than the corresponding older ones. The AES is believed to be secure, and it has withstood extensive cryptanalytic attack.

The ephemeral Diffie-Hellman ciphersuites provide forward secrecy without any known reduction in security in other areas. To obtain the maximum benefit from these ciphersuites:

4.8.4. URI Format

HTTPfilS is differentiated from HTTP URis by using the 'https' protocol identifier in place of the 'http' protocol identifierRef[13].

4.9. Advanced Encryption Standard (AES) Ciphersuites for (TLS)

At present, the symmetric ciphers supported by TLS are RC2, RC4, IDEA, DES, and triple DES. The protocol would be enhanced by the addition of AES ciphersuites, for the following reasons:

1. RC2, RC4, and IDEA are all subject to intellectual property claims. RSA Security Inc. has trademark rights in the names RC2 and RC4; and claims that the RC4 algorithm itself is a trade secret. Ascom Systec Ltd. owns a patent on the IDEA algorithm.
2 Triple DES is much less efficient than more modem ciphers.
3 Now that the AES process is completed there will be commercial pressure to use the selected cipher. The AES is efficient and has withstood extensive cryptanalytic efforts. The AES is therefore a desirable choice.

Currently the DHE ciphersuites only allow triple DES (along with some "export" variants which do not use a satisfactory key length). At the same time the DHE ciphersuites are the only ones to offer forward secrecy.

4.9.1. (AES) Ciphersuites for (TLS) Considerations

It is not believed that the new ciphersuites are ever less secured than the corresponding older ones. The AES is believed to be secure, and it has withstood extensive cryptanalytic attack.

The ephemeral Diffie-Hellman ciphersuites provide forward secrecy without any known reduction in security in other areas. To obtain the maximum benefit from these ciphersuites:

1. The ephemeral keys should only be used once. With the TLS protocol as currently defined there is no significant efficiency gain from reusing ephemeral keys.
2. Ephemeral keys should be destroyed securely when they are no longer required.
3. The random number generator used to create ephemeral keys must not reveal past output even when its internal state is compromised.

4.10. Wireless Transport Layer Security Protocol

SSL/fLS and IPsec are both security protocols for TCP/IP networks. And although TCP/IP can be run over wireless media, there are some issues with TCP/IP over wireless that call for something different The Wireless Application Protocol (WAP) Forum has specified a number of standards for wireless communication.

In WAP, the equivalents of SSLfilS and UDP are WTLS and WDP, respectively. WTLS is the Wireless Transport Layer Security protocol. WDP is the Wireless Datagram Protocol

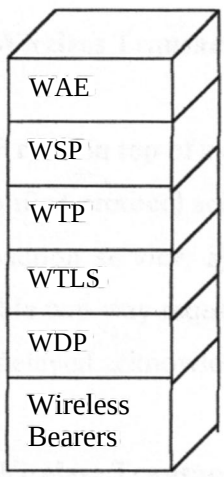
4.10.1. Issues with Wireless Communication

As well as the issues found in fixed networks, the following issues often characterize wireless networks:

- Communications are broadcast. Anyone with appropriate receiver equipment can intercept the communications.
- Bandwidth is *low.* A data rate of 9600 bit/s *is* considered relatively high.
- Bit error rate is relatively high.
- Client devices (e.g. pagers and mobile phones) have limited power, range and display capacity.

4.10.2. WAP Stack

The WAP Forum has specified a communication protocol stack following the [illegible]. The WAP protocol stack is shown in the diagram below:



Wireless Application Environment

Wireless Session Protocol

Wireless Transaction Protocol

Wireless Transport Layer Security

Wireless Datagram Protocol

Bearers

Figure 4 .2. WAP Protocol Stack

- Wireless Application Environment

The WAE defines the user interface on the phone. The application development environment facilitates the development of services that support multiple bearers. To achieve this, the WAE contains the Wireless Markup Language (WML), WMLScript, a scripting micro-language similar to JavaScript, and the Wireless Telephony Application (WTA). These are the tools that allow WAP-based applications to be developed.

- Wireless SessionProtocol

WSP is a sandwich layer that links the WAE to two session services: a) a connection oriented service operating above the Wireless Transaction Protocol, and, b) a connectionless service operating above the Wireless Datagram Protocol.

- Wireless Transaction Protocol

WTP runs on top of a datagram service such as User Datagram Protocol (UDP) to provide a simplified protocol suitable for low bandwidth mobile stations. WTP offers three classes of transaction service: a) unreliable one way request, b) reliable one way request, and c) reliable two way request-respond. WTP supports Protocol Data Unit (PDU) concatenation and delayed acknowledgment to help reduce the number of messages sent.

- Wireless Transport Layer Security

WTLS incorporates security features that are based upon the Internet Transport Layer Security (TLS) protocol standard. WTLS includes data integrity checks, privacy on the WAP Gateway to client leg and authentication.

- Wireless Datagram Protocol

WDP allows WAP to be bearer independent by adapting the transport layer of the underlying bearer. WDP presents a consistent data format to the higher layers of the WAP protocol stack thereby conferring the advantage of bearer independence to application developers.

- Bearers

A number of different physical bearer protocols are supported by the WAP.

- WAP Internet Services

To address the issues that the wireless medium brings to supporting Internet Services, the WAP protocols include features to improve performance. These include:

- o HTTP headers are transmitted in binary rather than plaintext
- o Sessions can be suspended and resumed without re-establishment.
- o WTP' significantly reduces the size and number of request-response transactions used by TCP.
- o A new certificate format with minimum required information and formatted for minimum size İs used in WTLS.

4.11. Summary

The *Transport Layer Security Protocol* (TLS) is an industry standard designed to protect the privacy of information communicated over the Internet.

From my researches on TLS over *Stream Control Transmission Protocol* (SCTP), I concluded that it is possible to authenticate TLS endpoints based on IP-addresses in certificates. Unlike TCP, SCTP associations can use multiple addresses per SCTP endpoint.

I noted that the goals of using the TLS protocol with *Lightweight Directory Access Protocol (LDAP)* are *to* ensure connection confidentiality and integrity, and to *optionally* provide for authentication. TLS expressly provides these capabilities.

The *Extension of Simple Mail Transfer Protocol* (SMTP) over TLS was explained; I noted that SMTP, is not an end-to-end mechanism. Thus, if an SMTP client/server pair decides to add TLS privacy, also I *explained* the usage of TLS with IMAP, POP3 and ACAP, I concluded that *TLS only* provides protection for *data sent* over a network connection.

Hyper Text Transfer Protocol (HTTP) upgrading to TLS and (HTTP) over TLS is explained. HTTP was originally used in the clear on the Internet. However, increased use of HTTP for sensitive applications has required securing measured; also I noted that HTTP/fLS is very simple as using HTTP over TCP.

Advanced Encryption Standard (AES) Cipher suites for TLS was defined and I concluded that the AES is believed to be secure, and it has withstood extensive cryptanalytic attack.

Another important TLS protocol was explained which wireless TLS protocol is, as well as the issues with wireless communication also explained.

5. MD5 ALGORITHM

5.1. Overview

A message digests is the fixed-length result of a one-way hash of the contents in the message, similar to a cryptographic checksum or cyclic redundancy check (CRC). This is a one-way hash-we cannot recover the original message contents from its digest. This can be easily seen given an example: there are literally an infinite number of wildly different messages that can be composed, each of differing lengths and content, but the digest itself is limited to a fixed-length, therefore it contains less information than is needed to reconstruct a message. So for each fixed-length digest, there are an **iounm:** number of messages that share that digests, but they are not similar (due to the quality of the cryptographic function within the hashing algorithm), and it is virtually impossible to create a message with a predetermined digest.

MOS message-digest algorithm takes as input a message of arbitrary length and proc\ee **Ni** output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large *file* must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA

The MOS algorithm is designed to be quite fast on 32-bit machines. In addition, the M05 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly.

The MDS algorithm is an extension of the MD4 message-digest algorithm MD5 is slightly slower than M04, but is more "conservative" in design. MOS was designed because it was felt that MD4 was perhaps being adopted for use more quickly than justified by the existing

critical review; because MD4 was designed to be exceptionally fast, it is "at the edge" in terms of risking successful cryptanalytic attack. MD5 backs off a bit, giving up a little in speed for a much greater likelihood of ultimate security. It incorporates some suggestions made by various reviewers, and contains additional optimizations. The MD5 algorithm is being placed in the public domain for review and possible adoption as a standard.

5.2. Terminologies and Notation

In this chapter a "word" is a 32-bit quantity and a "byte" is an eight-bit quantity. A sequence of bits can be interpreted in a natural manner as a sequence of bytes, where each consecutive group of eight bits is interpreted as a byte with the high-order (most significant) bit of each byte listed first. Similarly, a sequence of bytes can be interpreted as a sequence of 32-bit words, where each consecutive group of four bytes is interpreted as a word with the low-order (least significant) byte given first.

Let x_i denote "x sub i". If the subscript is an expression, we surround it in braces, as in x_{i+1}. Similarly, we use ^ for superscripts (exponentiation), so that x^i denotes x to the i-th power.

Let the symbol "+" denote addition of words (i.e., modulo-2^32 addition). Let X <<< s denote the 32-bit value obtained by circularly shifting (rotating) X left by s bit positions. Let not (X) denote the bit-wise complement of X, and Jet Xv Y denote the bit-wise OR of X and Y. Let X xor Y denote the bit-wise XOR of X and Y, and let XY denote the bit-wise AND of X and Y.

5.3. MD5 Algorithm Description

A message digest is a compact digital signature for an arbitrarily long stream of binary data. An ideal message digest algorithm would never generate the same signature for two different sets of input, but achieving such theoretical perfection would require a message digest as long as the input file. Practical message digest algorithms compromise in favour

of a digital signature of modest size created with an algorithm designed to make preparation of input text with a given signature computationally infeasible.

Message digest algorithms have much in common with techniques used in encryption, but to a different end; verification that data have not been altered since the signature was published. Many older programs requiring digital signatures employed 16 or 32 bit cyclical redundancy codes (CRC) originally developed to verify correct transmission in data communication protocols, but these short codes, while adequate to detect the kind of transmission errors for which they were intended, are insufficiently secure for applications such as electronic commerce and verification of security related software distributions.

The most commonly used present-day message digest algorithm is the 128-bit MD5 algorithm, developed by Ron Rivest of the MIT Laboratory for Computer Science and RSA Data Security, Inc.

The algorithm, with a reference implementation, was published as Internet RFC 1321 in April 1992, and was placed into the public domain at that time. Message digest algorithms such as MD5 are not deemed "encryption technology" and are not subject to the export controls some governments impose on other data security products. (Obviously, the responsibility for obeying the laws in the jurisdiction in which you reside is entirely your own, but many common Web and Mail utilities use MD5.

The MD5 algorithm has been implemented in numerous computer languages including C, Perl, and Java; if you're writing a program in such a language, track down a suitable subroutine and incorporate it into your program. The program described on this page is a command line implementation ofMD5, intended for use in shell scripts and Perl programs (it is much faster than computing an MD5 signature directly in Perl). This md5 program was originally developed as part of a suite of tools intended *to* monitor large collections of files (for example, the contents of a Web site) to detect corruption of files and inadvertent (or perhaps malicious) changes. That task is now best accomplished with more comprehensive packages such as Tripwire, but the command line md5 component continues to prove useful for verifying correct delivery and installation of software

packages, comparing the contents of two different systems, and checking for changes in specific files Ref [21].

We begin by supposing that we have ab-bit message as input, and that we wish to find its message digest. Here b is an arbitrary nonnegative integer; b may be zero, *it* need not be a multiple of eight, and it may be arbitrarily large. We imagine the bits of the message written down as follows:

m_O m_l ... m_{b-1}

The following five steps are performed to compute the message digest of the message.

5.3.1. Append Padding Bits

The message is "padded" (extended) so that its length (in bits) is congruentto 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long.

Padding is always performed, even if the length of the message is already congruent to 448, modulo 512.

Padding is performed as follows: a single "l" bit is appended to the message, and then "O" bits are appended so that the length in bits of the padded message becomes congruent to 448 modulo 512. In aJI, at least one bit and at most 512 bits are appended Ref [23].

5.3.2. Append Length

A 64-bit representation ofb (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2/\64, then only the low-order 64 bits ofb are used. (These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.)

At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words. Let M [0 ... N-1] denote the words of the resulting message, where N is a multiple of 16.

5.3.3. Initialize MD Buff er

A four-word buffer (A, B, C, D) is used to compute the message digest.
Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe de ba 98

word D: 76 54 32 10

5.3.4. Process Message in 16-Word Blocks

We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

F (X, Y, Z) =XYv not (X) Z

G (X, Y, Z) = XZ v Y not (Z)

H(X, Y,Z)=XxorYxorZ

I (X, Y, Z) = Y xor (Xv not (Z))

In each bit position F acts as a conditional: if X then Yelse Z.
The function F could have been defined using+ instead ofv since XY and not (X) Z will never have l's in the same bit position.) It is interesting to note that if the bits ofX, Y, and Z are independent and unbiased, the each bit of F (X, Y, Z) will be independent and unbiased.

The functions G, H, and I are similar to the function F, in that they act in "bitwise parallel" to produce their output from the bits of X, Y, and Z, in such a manner that if the corresponding bits ofX, Y, and Z are independent and unbiased, then each bit of G (X, Y, Z), H (X, Y, Z), and I(X,Y,Z) will be independent and unbiased. Note that the function His the bit-wise "xor" or "parity" function of its inputs.

This step uses a 64-element table T (1 ... 64] constructed from the sine function. Let T [i] denote the i-th element of the table, which is equal to the integer part of 4294967296 times abs (sin (I)), where i is in radians. The elements of the table are given in the appendix.

Do the following:

```
/* Process each 16-word block. */
  For i =Oto N/16-1 do
      /* Copy block i into X. */
   For j = 0 to 15 do
       SetX[j] to M[i*l6+j].
  end/* of loop onj */

/* Save A as AA, Bas BB, C as CC, and Das DD. */
  AA=A
  BB=B
  CC=C
  DD=D
```

/* **Round 1.** */

/* Let [abed ks i] denote the operation

a= b +((a+ F(b,e,d) + X[k] + T[i]) <<< s). */

/* Do the following 16 operations. */

[ABCD071] [DABC 1122] [CDAB2173] [BCDA3 224]

[ABCD4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]

[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 1711] [BCDA 112212]

[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

/* **Round** 2. */

/* Let [abed ks i] denote the operation

a= b +((a+ G(b,e,d) + X[k] + T[i]) <<< s). */

/* Do the following 16 operations.*/

[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]

[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA4 20 24]

[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]

[ABCD 13 5 29] [DABC 2 9 30] [CDAB 714 31] [BCDA 12 20 32]

/* **Round** 3. */

/* Let [abed k s t] denote the operation

a= b +((a+ H(b,e,d) + X[k] + T[i]) <<< s). */

/* Do the following 16 operations. */

[ABCD 5 4 33] (DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]

[ABCD 1437] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]

[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA6 23 44]

[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

/* **Round 4.** */

/* Let [abed ks t] denote the operation

a= b +((a+ I(b,e,d) + X[k] + T[i]) <<< s). */

/* Do the following 16 operations. */

[ABCD 0 6 49] [DABC 710 50] [CDAB 14 15 51] [BCDA 5 21 52]

[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]

[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]

[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

/* Then perform the following additions. (That is increment each of the four registers by *the* value it had before this block was started.)*/

A=A+AA

B=B+BB

C=C+CC

D=D+DD

end/* of loop on **i** */

5.3.5. **Output**

The message digest produced, as output is A, B, C, and D. That is, we begin with the low-order byte of A, and end with the high-order byte ofD.

5.4. Differences Between MD4 and MD5

The following are the differences between MD4 and MD5:

- A fourth round has been added.
- Each step now has a unique additive constant.
- The function gin round 2 was changed from (XY v XZ v YZ) to (XZ v Y not(Z)) to make g less symmetric.
- Each step now adds in the result of the previous step. This promotes a faster "avalanche effect".
- The order in which input words are accessed in rounds 2 and 3 is changed, to make these patterns less like each other.
- The shift amounts in each round have been approximately optimized, to yield a faster "avalanche effect." The shifts in different rounds are distinct.

5.5. **Summary**

The MDS message-digest algorithm is simple to implement, and provides a "fingerprint" or message digest ofa message of arbitrary length.

It is conjectured that the difficulty of coming up with two messages having the same message digest is on the order of 2/\64 operations, and that the difficulty of coming up with any message having a given message digest is on the order of2AJ28 operations. The MDS algorithm has been carefully scrutinized for weaknesses. It is, however, a relatively new algorithm and further security analysis is of course justified, as is the case with any new proposal of this sort

6. SECURE CHAT APPLICATION BY IMPLIMENTAING MD5 HASH FUNCTION AND STEGNOGROPHIC

6.1. **Overview**

Steganography is the practice of hiding information in computer pictures or music and relies on the fact that digital images and MP3 music files are made up of thousands of pieces of binary code, which tell a computer to color a pixel or produce a sound. Because so many small pieces of digital information are involved, a handful can easily be altered to convey secret messages, without changing the overall effect to the naked eye or ear.
The MD5 algorithm takes as *input* a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5. algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.

6.2.What is Stenography?

While I am discussing it in terms of computer security, steganography is really nothing new, as it has been around since the times of ancient Rome. For example, in ancient Rome and Greece, text was traditionally written on wax that was poured on top of stone tablets. If the sender of the information wanted to obscure the message - for purposes of military intelligence, for instance - they would use steganography: the wax would be scraped off and the message would be inscribed or written directly on the tablet, wax would then be poured on top of the message, thereby obscuring not just its meaning but its very existence Ref [24].

According to www.Dictionaıy.com, steganography (also known as "steg" or "stego") is "the art of writing in cipher, or in characters, which are not intelligible except to persons who have the key; cryptography"Ref [25]. In computer terms, steganography has evolved into the practice of hiding a message within a larger one in such a way that others cannot discern the presence or contents of the hidden message Ref [26]. In contemporary terms, steganography has evolved into a digital strategy of hiding a file in some form of multimedia, such as an image, an audio file (like a. wav or mp3) or even a video file.

6.3. What is Steganography Used for?

Like many security tools, steganography can be used for a variety of reasons, some good, some not so good. Legitimate purposes can include things like watermarking images for reasons such as copyright protection. Digital watermarks (aJso known as fingerprinting, significant especiaJly in copyrighting material) are similar to steganography in that they are overlaid in files, which appear to be part of the original file and are thus not easily detectable by the average person. Steganography can aJso be used as a way to make a substitute for a one-way hash value (where you take a variable length input and create a static length output string to verify that no changes have been made to the original variable length input)[sr4]. Further, steganography can be used to tag notes to online images (like post-it notes attached to paper files). Finally, steganography can be used to maintain the confidentiaJity of vaJuable information, to protect the data from possible sabotage, theft, or unauthorized viewing Ref [27].

Unfortunately, steganography can also be used for illegitimate reasons. For instance, if someone was trying to steal data, they could conceal it in another file or files and send it out in an innocent looking email or file transfer. Furthermore, a person with a hobby of saving pornography, or worse, to their hard drive, may choose to hide the evidence through the use of steganography. And, as was pointed out in the concern for terroristic purposes, it can be used as a means of covert communication. Of course, this can be both a legitimate and an illegitimate application.

6.4. Steganography in Images

In essence, image steganography is about exploiting the limited powers of the human visual system (HVS). Within reason, any plain text, ciphertext, other images, or anything that can be embedded in a bit stream can be hidden in an image. Image steganography has come quite far in recent years with the development of fast, powerful graphical computers, and steganographic software is now readily available over the Internet for everyday users.

6.5. Image Encoding Techniques

Information can be hidden many different ways in images. Straight message insertion can be done, which will simply encode every bit of information in the image. More complex encoding can be done to embed the message only in " noisy" areas of the image that will attract less attention. The message may also be scattered randomly throughout the cover image.

The most common approaches to information hiding in images are:

- Least significant bit (LSB) insertion
- Masking and filtering techniques
- Algorithms and transformations

6.5.1. Least Significant Bit Insertion

The least significant bit insertion method is probably the most well known image steganography technique. It is a common, simple approach to embedding information in a graphical image file. Unfortunately, it is extremely vulnerable to attacks, such as image manipulation. A simple conversion from a GIF or BMP format to a lossy compression format such as JPEG can destroy the hidden information in the image.

When applying LSB techniques to each byte of a 24-bit image, three bits can be encoded into each pixel. (As each pixel is represented by three bytes.) Any changes in the pixel bits

will be indiscernible to the human eye. For example, the letter A can be hidden in three pixels. Assume the original three pixels are represented by the three 24-bit words below:

(00100111 111O1001 11001000) (00100111 11001000 111O1001) (11001 000001001 11 11101001)

The binary value for the letter A is (10000011). Inserting the binary value of A into the three pixels, starting from the top left byte, would result in:

(00100111 1110100011001000) (0010011011001000 11101000) (1100100000100111 11101001)

The emphasized bits are the only bits that actually changed. The main advantage of LSB insertion is that data can be hidden in the least and second to least bits and still the human eye would be unable to notice it.

When using LSB techniques on 8-bit images, more care needs to be taken, as 8-bit formats are not as forgiving to data changes as 24-bit formats are. Care needs to be taken in the selection of the cover image, so that changes to the data will not be visible in the stego-image. Commonly known images, (such as famous paintings, like the *Mona Lisa)* should be avoided. In fact, a simple picture of your dog would be quite sufficient.

When modifying the LSB bits in 8-bit images, the pointers *to* entries in the palette are changed. It is important to remember that a change of even one bit could mean the difference between a shade of red and a shade of blue. Such a change would be immediately noticeable on the displayed image, and is thus unacceptable. For this reason, data-hiding experts recommend using grey-scale palettes, where the differences between shades is not as pronounced. AJtematively, images consisting mostly of one color, such as the so-called Renoir palette, named because it comes from a 256 colour version of Renoir's "Le Moulin de la Galette".

6.6. **Secure Chat Application Development**

Well now I am going to try to explain the process and the need to develop a secure chat application implementing two major concepts of md5 hash's and image stegnography.

6.7. Explanation of MDS Secure Program by Block Diagram

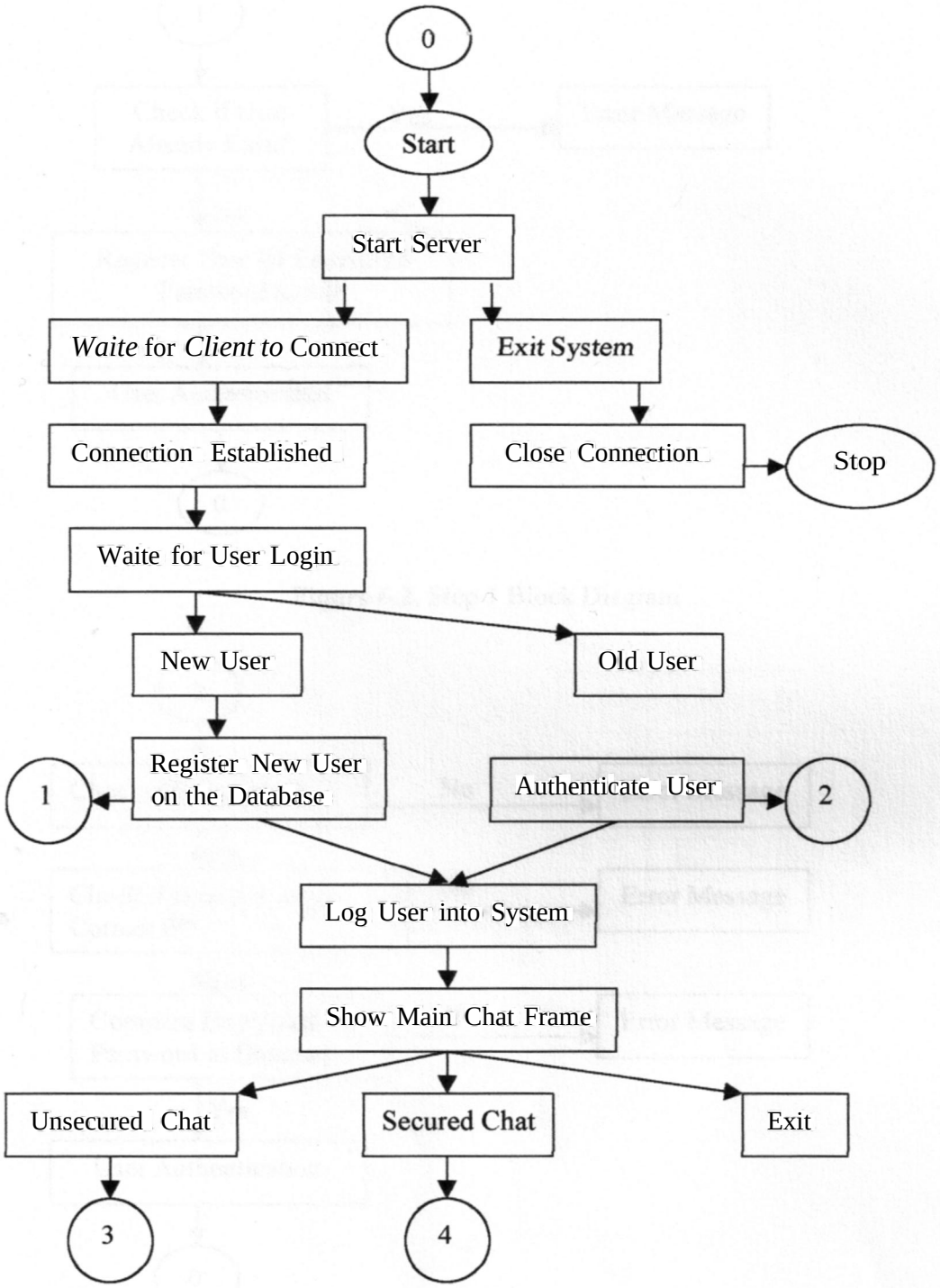


Figure 6.1 Server Side Block Diagram

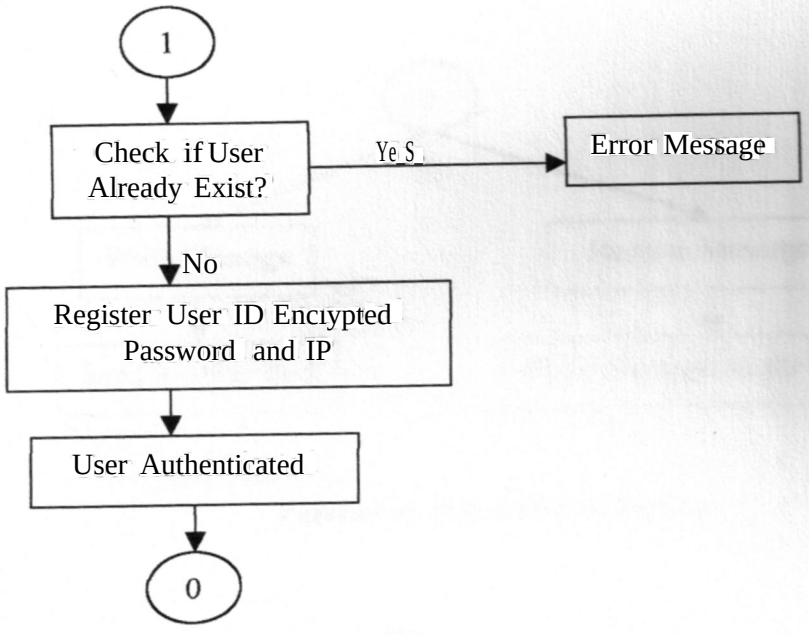


Figure 6.2. Step I Block Diagram

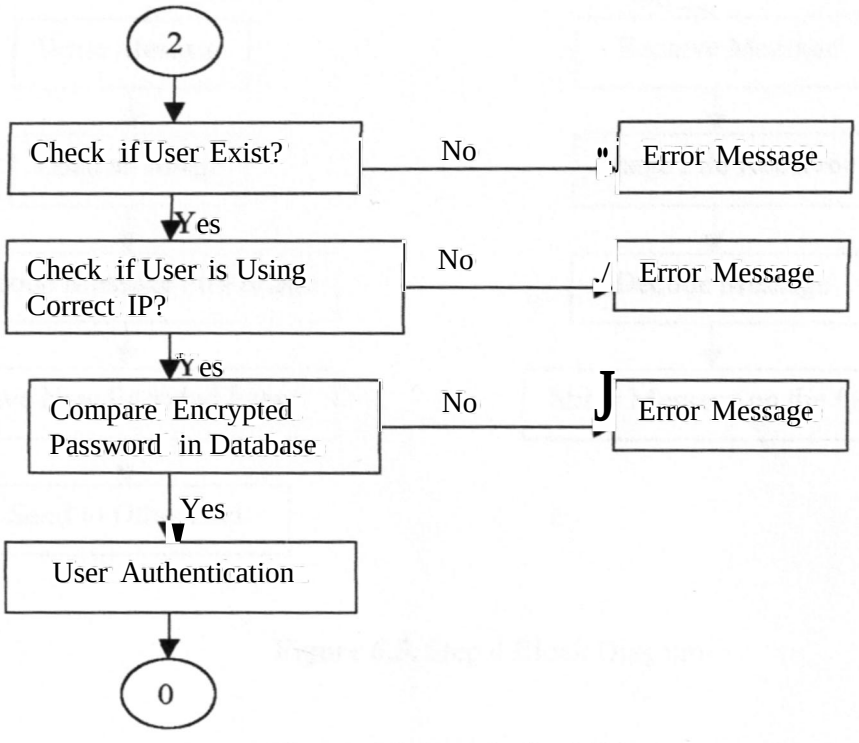


Figure 6.3. Step 2 Block Diagram

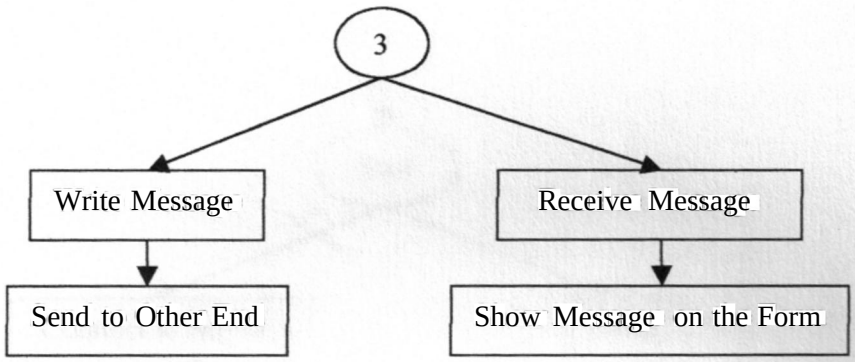


Figure 6.4. Step 3 Block Diagram

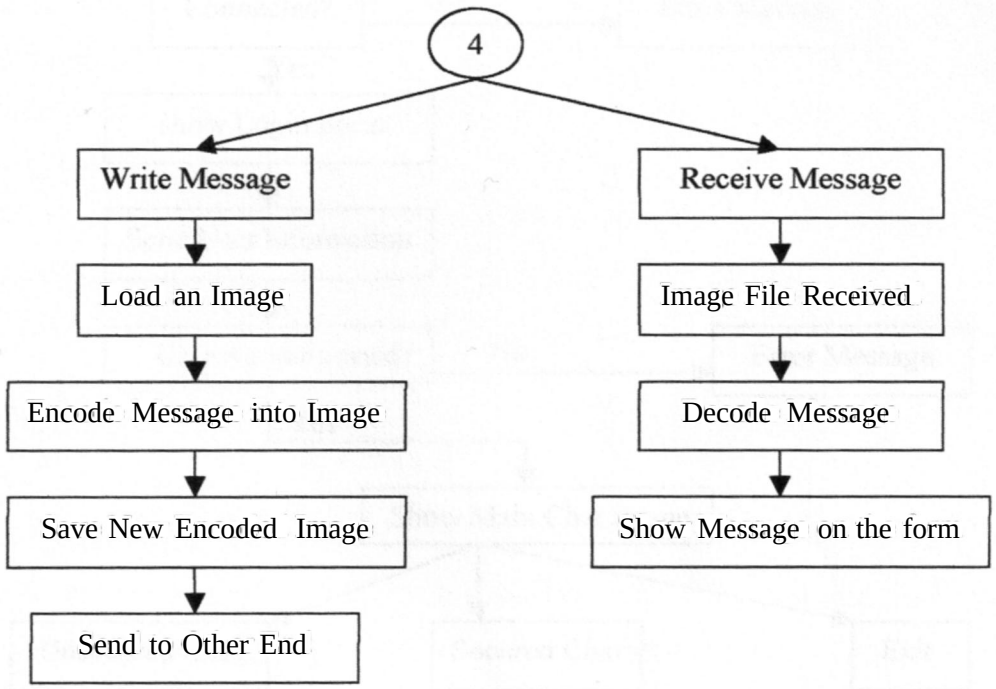


Figure 6.5. Step 4 Block Diagram

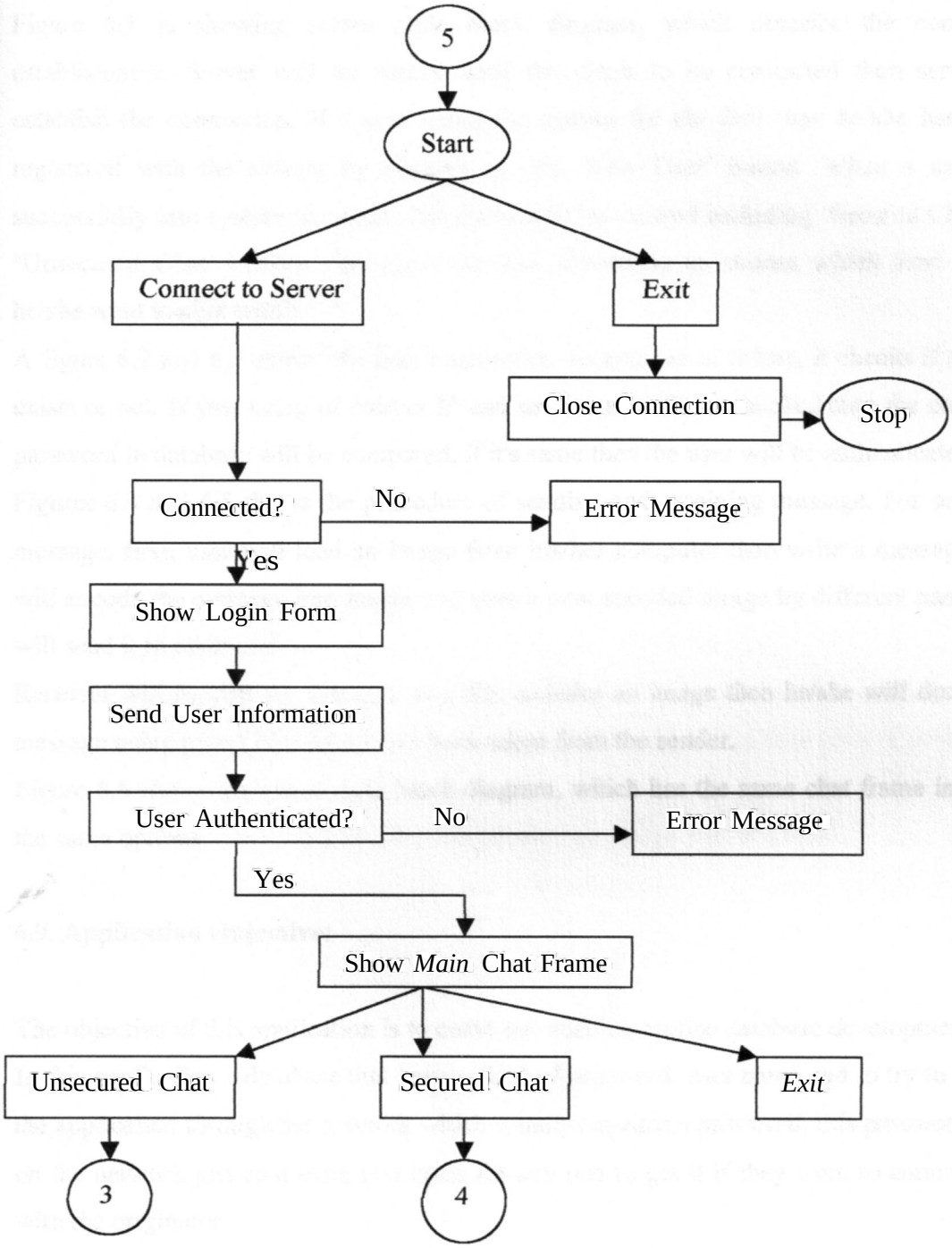


Figure 6.6. Client Side Block Diagram

6.8. Analysis of Application's Block Diagram

Figure 6.1 is showing server slide block diagram, which describe the connection establishment. Server will be waited until the client to be connected then server can establish the connection. ff a user using the system for the first time be/she has to get registered with the system by clicking on the 'New User' button. When a user loge successfulfy into system the main chat frame wilt be showed including 'Secured' Chat' and 'Unsecured Chat' buttons that gives the user alternative to choose which kind of chat he/she want to char within.

A figure 6.2 and 6.3 shows the user registration acceptance or refuse, it checks if the user exists or not. If yes; using of correct IP- and user name wilt be checked' then the encrypted' password in database will be compared, if it's same then the user will be authenticated.

Figures 6.4 and 6.5 shows the procedure of sending and' receiving message. For sending a message, first; user will load an image from his/her computer then write a message. User will encode the message into image and save a new encoded image by different name, then will send it to other end.

Receiver witt receive the message as a fite includes an image then he/she wilt decode the message using privet key, which has been taken from the sender.

Figure 6.6 shows the client stkte block diagram, which has the same chat frame Inchiding the same options.

6.9. Application Objective:

Tiie objective of this application is to come out which a routine database development.

In this application a database that have a field of password, user name and to try to login to the application through the network which usually requires a password, this passwordtravet on the network just as a plain text open for any one to get it if they want to communicate with the originator.

After the initial information the developed application takes and fulfils; the main objectives which are:

Crafting a secure login system that can later be used- in any kind' of application for authentication and password security.

Develop a chat solution which not only use secure authentication but also can do a secure data transfer.

6.10. Application Analysis:

The application consists of two parts mainly a Server and- a Client First a client part connects to the server and then after connection a user try to login by using its user name and password'.

Now if a user is using the system for the first time he/ she has to get registered with the system. So when the user provides the password' the password is encrypted' into a message digest and sent to the server.

The server receives the encrypted' password' and registers it on the database. Now even some know the encrypted password and the key, which generates it, due to the property of the MD5 it cannot be decrypted, as MD5 is a one-way bash function.

Now comes the chatting part, after logging on to server as an authentic user the user have an option to encrypt or encode bis/her data in an image file and then send it to the other end.

This way on the other end' the user wilt receive an image and if some one even hack the data they will just get an image file and without knowing the special key they cannot decrypt the data out of the image.

6.11. Summary

This was the most important chapter of this thesis, which describes the stenography in details. The author developed a Visual Basic based security application program using the MD5 encryption algorithm to encode and decode by image any kind- of information between a client and server, and include an explanation of the program, application analysis, application objectives and using program, also block diagram was provided in this chapter, the program developed by author is given in an appendix at the end of the thesis.

CONCLUSION

Virtually all businesses, most government agencies, and many individuals now have Web sites. The number of individuals and companies Internet access is expanding rapidly, and all of these have browsers. As a result, businesses are enthusiastic about setting up facilities on the Web for electronic commerce. But the reality is that the Internet and the Web are extremely vulnerable to compromises of various sorts. As businesses wake up to this reality, the demand for secure Web services grows.

Web services are fast becoming a reality for both developers and enterprises. Web services make it easy for enterprises to integrate existing legacy applications. Protected behind firewalls, enterprise Web services exchange sales, inventory, and customer relationship data, even though the applications that manage that data reside on diverse software platforms.

There is strong momentum to bring Web services technology into the mainstream of network computing. Thus many companies tried to support Authentication and Authorization by some thing called cryptographic Algorithms and digital signatures.

Many Web browsers protect transmissions using the protocol SSL (Secure Sockets Layer). The client and server machines exchange nonces and compute session keys from them. The latest version of the protocol is called TLS (Transport Layer Security) it closely resembles SSL 3.0. Is TLS really secured? Many proofs suggest that it is.

TLS is the successor to the Secure Sockets Layer, and is nearly identical to SSL except that it implements.

The thesis has included six chapters followed by a Visual Basic program developed by the author to allow the server and client to authenticate each other and to negotiate an encryption algorithm to revert the message to an encrypted photo before the application protocol transmits.

Chapter 1 has presented an explanation of internet security essentials and the main definitions of servers and web services security have been described.

To provide our Engineers with tools to implement the information protection reguested the Cryptography and Web security have been described in chapter 2 in relation to the ciphers used with SSL/TLS protocols.

To know how to transfer information privately and securely across the internet the Transport Layer Security technique and the advantages of SSL/TLS has described in chapter 3.

Transport Layer Security protocols, the differences between TLS protocol and other security protocols have been presented in chapter 4.

Chapter 5 has described the usage of message-digest security algorithm MDS and shows the differences between MD4 and MD5 by giving example in details.

Finally, chapter six has described steganography and analyzed the application. The author has developed a Visual Basic program to encode and decode by image any kind of information between a client and server.

REFERENCES

[I] Network Security (Private Communication in a Public World):
Charlie Kaufman, Radla Perlman, Mike Specinep. Computer Security Resource Center, Feb 1995

[2] RSA R.Rivest, A Shamir, and L. M. Adleman:
"Amethod for Obtaining Digital Signature and Public-Key Cryptosystem". Communication of the ACM, V.21, N.2, Feb 1978, PP. 120-126.

[3] "Security Mechanism", http://www.versign.com/wss.pdf

[4] Adams, C. "Simple and Effective Key Scheduling for Symmetric Ciphers." Proceeding, Workshop in Selected Areas of Cryptography, SAC" 94. 1994.

[5] "Secure Socket Layers Protocol and Application" Allan Schiffman:
Tersia System.Inc{http://www.tersia.com}.

[6] Wiener, M. "Efficient DES Key Search." Proceeding; Crypto'93 , 1993.

[7] Comparison of Protocols for Secure www.SSL and S-HTTP:
National Institute of Standards and Technology, U.S. Department of Commerce.DRAFT, May 1994

[8] "Wireless Security", Models, Threats and Solution:
Randall K. Nichols, Danos C. Lekkas. New York, 1997.

[9] "Security the Internet Without Wires":
Deborah Durham- Vichr (deborah@nemix.com), Freelance Writer,
Kimberly Getgen (kgetgen@rsasecurity.com), RSA Security.

(10] D. Balenson, "Privacy Enhancement for Internet ElectronicMail: Part III--Algorithms, Modes, and Identifiers," RFC 1423, February 1993.

(11] M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution, "In Advances in Cryptology--CRYPTO '93 Proceedings, Springer-Verlag, 1993.

(12] Tim Berners-Lee, Robert Cailliau, Jean-Francois Groff, and Bernd Pollermann,"World-Wide Web: The Information Universe,"Electronic Networking.Research, Applications, and Policy, volume 1, no 2, Spring 1992.

[13] Tim Berners-Lee, "Hypertext Transfer Protocol," available electronically at URL ftp://info.cern.ch/pub/www/doc/http-spec.ps, 1993.

[14] F. Rabitti, E. Bertino, W. Kim, and D. Woelk, "A Model of Authorization for Next-Generation Database Systems," ACM Transactions on Database Systems, 16: 1, 1991.

[15] B. Schneier Applied Cryptography: Protocol, Algorithm, and Source Code in C, Published by Jon Wiley and Sons, Inc. 1994

(16] Codes and Cryptography: Dominic Welsh, June, 1994.

[l 7] Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design:_Electronic Frontier Foundation, John Gilmore (Editor). CESG Report, August 1993.

[18] Handbook of Applied Cryptography (CRC Press Series on Discrete Mathematics and Its Applications): National Institute of Standards and Technology, U.S. Department of Commerce, 18 May 1994.

[19] Adam, C. "Simple and Effective Key Scheduling for Symmetric Ciphers." Proceedings, Workshop in Selected Areas of Cryptography, SAC" 94. 1994

(20] Note of Lecture, http://www.shore.net/ws/security2e.htm1

[21] Adam, C. "Constructing Symmetric Ciphers Using CAST Design, Codes and Cryptography, 1997

[22] Public Key Cryptography: Second International Workshop on Practice and Theory in Public Key Cryptography, Pkc'99, Kamakura, Japan, March 1-3, 1999 International Workshop on Practice and Theory in Public Key cryptography.

[23] The Twofish Encryption Algorithm: A 128-Bit Block Cipher: Bruce Schneier. Inc. 1994

[24] Steganography, by Neil F. Johnson, George Mason University, http://www.jjtc. corn/ stegdoc/ sec202.htrnl

[25] http:// dictionary. reference. com/search?q=steganography

[26] The Free On-line Dictionary of Computing, © 1993-2001 Denis Howe http ://www. nightflight. corn/foldoc/index. html

[27] Steganography: Hidden Data, by Deborah Radcliff, June 10, 2002, http ://www.computerworld.com/securitytopics/security/story/0, 10801, 71726, 00. html

APPENDIX A

```
Option Explicit
Dim lgin As Integer
Dim cop As Integer
Dim fload As Integer
Dim men As Boolean
Dim fnarne As String
Dim p As String
Dim bufferQ As Byte
Dim IBytes As Long
Dim temp As String

Private Sub EncodeByte(Value As Byte)
 End Sub

Private Sub cmdStopSer_Click()
If scServer.State <> 0 Then
 scServer. Close
End If
stBar.Panels(l).Text = scMsg(scServer.State, Me, scServer)
End Sub

Private Sub cmdEncode _Click()
cmdSend.Enabled = True
End Sub
```

```
Private Sub cmdLoad_ Clickt)
With cmd
.DialogTitle = "Select Picture"
.Filter= "Windows Bitmap (*.bmp)I*. brnp] Jpeg (* .jpeg)I*.jpg"
.ShowOpen
fname = .FileName

End With
 cmdEncode.Enabled = True
 End Sub

Private Sub cmdSend_ Clickt)

If cop = 1 Then
   'scServer.SendData "eül "
   'scServer.SendData txtmsg.Text
   'scServ.SendData txtmsg. Text
   'txtchat.Text = txtchat.Text & vbCrLf & "Server ..:" & txtmsg.Text
  lBytes = 0
 ReDim buffer(FileLen(cmd.FileName)- - 1)
 Open cmd.FileName For Binary As 1
 Get #1, 1, buffer
 Close #1
 Load wsTcp
 wsTcp.RemoteHostIP = scServer.RemoteHostIP
 wsTcp.RemotePort = 1111
 wsTcp.Connect
Elself cop = 0 Then

scServer.Send.Data txtmsg. Text
txtchat.Text = txtchat.Text & vbCrLf & "Server ..:" & txtmsg.Text
```

```
End If
End Sub

Private Sub Form_Load()
Dim a As Boolean
a = openConnO

stBar.Panels{1 ).Text= scMsg(scServer.State, Me, scServer)
lgin=O
stBar.Panels(3). Text= stUser(Jgin)
fload = 0
men =False
p =App.Path
End Sub

Private Sub mnExit_ ClickO
IfMsgBox(" .. ::Do you Wish to Exist the Server?:: ..", vbYesNo + vbQuestion,
Me.Caption)= vb Yes Then
 scServer. Close
 Unload Me
 End
End If
End Sub

Private Sub rnnStart_ Clickt)
If scServer. State <> 0 Then
  scServer. Close
  scServer.Listen
Else
   scServer.Listen
End If
```

```
stBar.Panels(l).Text = scMsg(scServer.State, Me, scServer)
End Sub

Private Sub mnStop _Clickt)
If scServer. State <> 0 Then
 scServer. Close
End If
stBar.Panels(l).Text = scMsg(scServer.State, Me, scServer)
End Sub

Private Sub Option 1_ClickQ
cop= 1
cmdSend.Enabled = False
cmd.Load.Enabled = True
'cmd.Encode.Enabled = True
'cmdDecode.Enabled = True
End Sub

Private Sub Option2_Click0
cop=O
cmd.Load.Enabled = False
cmdEncode.Enabled = False
cmdDecode.Enabled = False
cmdSend.Enabled = True
End Sub

Private Sub scServ_ConnectionRequest(ByVal requestID As Long)
scServ. Close
scServ.Accept requestID
End Sub
```

```
Private Sub scServer_ConnectionRequest(ByVal requestID As Long)
scServer. Close
scServer.Accept requestID
stBar.Panels(l).Text = scMsg(scServer.State, Me, scServer) & "By:." &
scServer.RemoteHostlP
End Sub

Private Sub scServer_DataArrival(ByVal bytesTotal As Long)
Dim uida As String
Dim passa As String
Dim ipa As String
Dim rData As String
Dim res As String

scServer.GetData rData, vbString

If lgin = 0 Then
   passa= Mid(rData, 2, 33)
   uida = Mid(rData, 34)
   ipa = scServer.RemoteHostlP

   IfMid(rData, 1, 1) = "n" Then

   lgin = regUser(gcnn, uida, passa, ipa, 1, stBar)
   stBar.Panels(4).Text = lgin
     If lgin = 0 Then
   scServer. SendData ("O")
     Elself lgin = 1 Then
   frchat. Visible = True

   scServer.SendData ("I")
```

```
    Elself lgin = 2 Then
  scServer.SendData ("2")
  frchat.Visible = False
    End If

  ElseifMid(rData, 1, 1) = "o" Then·

  lgin = regUser(gcnn, uida, passa, ipa, 2, stBar)
  stBar.Panels(4). Text= lgin
    If lgin = 0 Then
  scServer.SendData ("O")
    Elseif lgin = 1 Then
  scServer.SendData ("1 ")
  frchat. Visible= True
    Elseif lgin = 2 Then
  scServer.SendData ("2")
  frchat.Visible = False
    End If
  End If

Elself lgin = 1 The
      txtchat.Text = txtchat.Text & vbCrLf & "Client.:" & rData
End If
End Sub


Private Sub tbar_ButtonClick(ByVal Button As MSComctlLib.Button)
Select Case Button.Index
  Case 1
  If scServer.State <> 0 Then
```

```
  scServer. Close
  scServer.Listen

Else
   scServer.Listen
End If
stBar.Panels(l).Text   = scMsg(scServer.State,  Me,  scServer)
 Timerl  .Enabled  = True

 Case2
 If scServer. State  <> 0 Then
 scServer. Close
 lgin=O
 frchat.Visible  = False

End If
stBar.Panels(l  ). Text=  scMsg(scServer.State,  Me,  scServer)
 End Select
End Sub

Private Sub Timerl_TimerQ
stBar.Panels(l).Text   = scMsg(scServer.State,  Me,  scServer)
If scServer.State  = 7 Then
  stBar.Panels(2).Text  = "Waiting for User To Login"
Elself scServer.State  = 8 Then
   stBar.Panels(2). Text= "Client Disconnected  Waiting to Connect.'.. .."
Elself scServer.State  = 0 Then
   stBar.Panels(2).Text  = "Server is Closed Initialize It:: .."
Else
   stBar.Panels(2).Text  = ""
End If
```

```
End Sub
Private Function CalculateSeed(ByVal password As String) As Long

  ' Calculate a numeric seed based on the password
  'You may use any method here, as long as the result is always
  ' the same for the same password.

  Dim Value As Long
  Dim ch As Long
  Dim shiftl As Long
  Dim shift2 As Long
  Dim i As Integer
  Dim str_len As Integer

  shiftl = 3
  shifl:2 = 17
  str_len = Len(password)

  For i = 1 To str_len
    ch = Asc(Mid$(password, i, 1))
    Value= Value Xor (ch* 2 /\ shiftl)
    Value= Value Xor (ch* 2 /\ shift2)
    shiftl = (shiftl + 7) Mod 19
    shift2 = (shift2 + 13) Mod 23
  Nexti
  CalculateSeed = Value
End Function

Private Sub wsTcp_Close()
 Unload wsTcp
End Sub
```

```
Private Sub wsTcp _Connectı)
wsTcp.SendData cmd.FileTitle & vbCrLf
End Sub

Private Sub wsTcp_DataArrival(ByVal bytesTotal As Long)
 wsTcp. GetData temp
 IflnStr(temp, vbCrL.f) <> 0 Then temp = Left(temp, InStr(temp, vbCrLf) - 1)
 If temp= "OK" Then
  wsTcp.SendData buffer
 Else
  Unload wsTcp
 End If
End Sub

Private Sub wsTcp_SendCompleteO
 If temp = "OK" Then
  temp=""
  Unload wsTcp
 End If
End Sub

Private Sub wsTcp_SendProgress(ByVal bytesSent As Long, ByVal bytesRemaining As
Long)
 Iftemp = "OK" Then
  IBytes = IBytes + bytesSent
 End If
End Sub
```

Md5 Calculating Module

```
Option Explicit

Private lngTrack As Long
Private arrLongConversion( 4) As Long
Private arrSplit64(63) As Byte

Private Const OFFSET_4 = 4294967296#
Private Const MAX1NT_4 = 2147483647

Private Const S11 = 7
Private Const S12 = 12
Private Const S13 = 17
Private Const SI4 = 22
Private Const S21 = 5
Private Const S22 = 9
Private Const S23 = 14
Private Const S24 = 20
Private Const S31 = 4
Private Const S32 = 11
Private Const S33 = 16
Private Const S34 = 23
Private Const S41 = 6
Private Const S42 = 10
Private Const S43 = 15
Private Const S44 = 21

Private Function MD5Round(strRound As String, a As Long, b As Long, C As Long, d As
Long, X As Long, S As Long, ac As Long) As Long

  Select Case strRound
    Case Is = "FF"
```

```
            a= MD5LongAdd4(a, (b And C) Or (Not (b) And d), X, ac)
            a= .MD5Rotate(a, S)
            a= MD5LongAdd(a, , b)

        Case Is = "GG"
            a= MD5LongAdd4(a, (b And d) Or (C And Not (d)), X, ac)
            a= MD5Rotate(a, S)
            a= MD5LongAdd(a, , b)

        Case Is = "HH"
            a = .MD5LongAdd4(a, , b Xor C Xor d, X, ac)
            a = MD5Rotate(a, S)
            a= MD5LongAdd(a, b)

        Case Is = "IT"
            a= MD5LongAdd4(a, C Xor (b Or Not (d)), X, ac)
            a = MD 5Rotate( a, S)
            a= MD5LongAdd(a, , b)
    End Select
End Function

Private Function MD5Rotate(lngValue As Long, lngBits As Long) As Long

    Dim lngSign As Long
    Dim lngl As Long

    lngBits = (lngBits Mod 32)

    IflngBits = 0 Then MD5Rotate = lngValue: Exit Function

    For lngl = 1 To lngBits
```

```
        lngSign = lngValue And &HC0000000
        lngValue = (lngValue And &H3FFFFFFF) * 2
        lngValue= lngValue Or ((lngSign < 0) And 1) Or (CBool(lngSign And &H40000000)
And &H80000000)
    Next
    MD5Rotate = lngValue
End Function

Private Function TRIDO As String
    Dim sngNum As Single, lngnum As Long
    Dim strResult As String

    sngNum = Rnd(2147483648#)
    strResult = CStr(sngNum)

    strResult = Replace(strResult, "0.", "")
    strResult = Replace(strResult, ".", "")
    strResult = Replace(strResult, "E-", "")

    TRID = strResult
End Function
Private Function MD564Split(lngLength As Long, bytBuffer() As Byte) As String

    Dim lngBytesTotal As Long, lngBytesToAdd As Long
    Dim intLoop As Integer, intLoop2 As Integer, lngTrace As Long
    Dim intlnnerLoop As Integer, intLoop3 As Integer

    lngBytesTotal = lngTrack Mod 64
    lngBytesToAdd = 64 - lngBytesTotal
    lngTrack = (lngTrack + lngLength)
```

```
  IflngLength >= lngBytesToAdd Then
    For intLoop = 0 To lngBytesToAdd - 1
      arrSplit64(lngBytesTot.al + intLoop) = bytBuffer(intLoop)
    Next intLoop

    MD5Conversion arrSplit64
    lngTrace = (lngLength) Mod 64

    For intLoop2 = lngBytesToAdd To lngLength - intLoop - lngTrace Step 64
      For intlnnerLoop = 0 To 63
        arrSplit64(intfunerLoop) = bytBuffer(intLoop2 + intlnnerl.oop)
      Next intlnrıerl.oop
      MD5Conversion arrSplit64

    Next intLoop2
    lngBytesTotal = 0
  Else

   intLoop2 = 0
  End If
  For intLoop3 = 0 To lngLength- intLoop2 - I

    arrSplit64(lngBytesTotal + intLoop3) = bytBuffer(intLoop2 + intLoop3)
  Next intLoop3
End Function

Private Function MD5StringArray(strlnput As String) As Bytet)

  Dim intLoop As Integer
  Dim bytBufferO As Byte
  ReDim bytBuffer(Len(strinput))
```

```
    For intLoop = 0 To Len(strinput) - 1
        bytBuffer(intLoop) = Asc(Mid(strinput, intLoop + I, 1))
    Next intLoop
    MD5StringArray = bytBuffer
End Function

Private Sub MD5Conversion(bytBufferO As Byte)

    Dim X(l 6) As Long, a As Long
    Dim b As Long, C As Long
    Dim dAsLong

    a= arrLongConversion(l)
    b = arrLongConversion(2)
    C = arrLongConversion(3)
    d = arrLongConversion( 4)

    MD5Decode 64, X, bytBuffer
    MD5Round "FF", a, b, C, d, X(O), SI 1, -680876936
    MD5Round "FF", d, a, b, C, X{l), S12, -389564586
    MD5Round "FF", C, d, a, b, X(2), S13, 606105819
    MD5Round "FF", b, C, d, a, X(3), S14, -1044525330
    MD5Round "FF", a, b, C, d, X(4), Sl l, -176418897
    MD5Round "FF", d, a, b, C, X(S), S12, 1200080426
    MD5Round "FF", C, d, a, b, X(6), S13, -1473231341
    MD5Round "FF", b, C, d, a, X(7), S14, -45705983
    MD5Round "FF", a, b, C, d, X(8), Sl 1, 1770035416
    MD5Round "FF", d, a, b, C, X(9), S12, -1958414417
    MD5Round "FF", C, d, a, b, X(IO), S13, -42063
    MD5Round "FF", b, C, d, a, X(l 1), S 14, -1990404162
```

```
MD5Round "FF", a, b, C, d, X(l2), SI 1, 1804603682
MD5Round "FF", d, a, b, C, X(13), S12, -40341101
MD5Round "FF", C, d, a, b, X(l4), Sl3, -1502002290
MD5Round "FF", b, C, d, a, X(l5), Sl4, 1236535329

MD5Round "GG", a, b, C, d, X(l), S21, -165796510
MD5Round "GG", d, a, b, C, X(6), S22, -1069501632
MD5Round "GG", C, d, a, b, X(l 1), S23, 643717713
MD5Round "GG", b, C, d, a, X(O), S24, -373897302
MD5Round "GG", a, b, C, d, X(5), S21, -701558691
MD5Round "GG", d, a, b, C, X(lO), S22, 38016083
MD5Round "GG", C, d, a, b, X(l5), S23, -660478335
MD5Round "GG", b, C, d, a, X(4), S24, -405537848
MD5Round "GG", a, b, C, d, X(9), S21, 568446438
MD5Round "GG", d, a, b, C, X(14), S22, -1019803690
MD5Round "GG", C, d, a, b, X(3), S23, -187363961
MD5Round "GG", b, C, d, a, X(8), S24, 1163531501
MD5Round "GG", a, b, C, d, X(l 3), S21, -1444681467
MD5Round "GG", d, a, b, C, X(2), S22, -51403784
MD5Round "GG", C, d, a, b, X(7), S23, l 735328473
MD5Round "GG", b, C, d, a, X(l2), S24, -1926607734

MD5Round "HH", a, b, C, d, X(5), S31, -378558
MD5Round "HH", d, a, b, C, X(8), S32, -2022574463
MD5Round "Jffi", C, d, a, b, X(l 1), S33, 1839030562
MD5Round "HH", b, C, d, a, X(l4), S34, -35309556
MD5Round "HH", a, b, C, d, X(l ), S31, -1530992060
MD5Round "HH", d, a, b, C, X(4), S32, 1272893353
MD5Round "HH", C, d, a, b, X(7), S33, -155497632
J\.ID5Round "HH", b, C, d, a, X(lO), S34, -1094730640
J\.ID5Round "HH", a, b, C, d, X(13), S31, 681279174
```

```
    MD5Round "HH", d, a, b, C, X(O), S32, -358537222
    MD5Round "HR", C, d, a, b, X(3), S33, -722521979
    MD5Round "HH", b, C, d, a, X(6), S34, 76029189
    MD5Round "HH", a, b, C, d, X(9), S31, -640364487
    MD5Round "HH", d, a, b, C, X(12), S32, -421815835
    MD5Round "IDI", C, d, a, b, X(l 5), S33, 530742520
    MD5Round "HH", b, C, d, a, X(2), S34, -995338651

    MD5Round "Il", a, b, C, d, X(O), S41, -198630844
    .MD5Round "II", d, a, b, C, X(7), S42, 1126891415
    MD5Round "II", C, d, a, b, X(l4), S43, -1416354905
    MD5Round "Il", b, C, d, a, X(5), S44, -57434055
    MD5Round "Il", a, b, C, d, X(l2), S41, 1700485571
    MD5Round "Il", d, a, b, C, X(3), S42, -1894986606
    .MD5Round "II", C, d, a, b, X(l 0), S43, -1051523
    MD5Round "Il", S.C, d, a, X(l), S44, -2054922799
    MD5Round "II", a, b, C, d, X(8), S41, 1873313359
    .MD5Round "Il", d, a, b, C, X(15), S42, -30611744
    MD5Round "II", C, d, a, b, X(6), S43, -1560198380
    MD5Round "Il", b, C, d, a, X(13), S44, 1309151649
    MD5Round "Il", a, b, C, d, X(4), S41, -145523070
    MD5Round "Il", d, a, b, C, X(l l), S42, -1120210379
    MD5Round "Il", C, d, a, b, X(2), S43, 718787259
    MD5Round "Il", b, C, d, a, X(9), S44, -343485551

    arrLongConversion(l) = MD5LongAdd(arrLongConversion(l ), a)
    arrLongConversion(2) = .MD5LongAdd(arrLongConversion(2), b)
    arrLongConversion(3) = .MD5LongAdd(arrLongConversion(3), C)
    arrLongConversion(4) = MD5LongAdd(arrLongConversion(4), d)
End Sub
```

```
Private Function MD5LongAdd(lngVall As Long, lngVal2 As Long) As Long

    Dim lngHigh Word As Long
    Dim lngLowWord As Long
    Dim lngOverflow As Long

    lngLowWord = (lngVall And &HFFFF&) + (lngVal2 And &HFFFF&)
    lngOverflow = lngLowWord \ 65536
    lngHighWord = (((lngVall And &HFFFFOOOO)\65536) + ((lngVal2 And
&HFFFFOOOO)\ 65536) + lngOverflow) And &HFFFF&

    MD5LongAdd = MD5LongConversion((lngHighWord * 65536#) + (lngLowWord And
&HFFFF&))
End Function

Private Function MD5LongAdd4(lngVall As Long, lngVal2 As Long, lngVa13 As Long,
lngVal4 As Long) As Long

    Dim lngHighWord As Long
    Dim lngLowWord As Long
    Dim lngOverflow As Long

    lngLowWord = (lngVall And &HFFFF&) + (lngVal2 And &HFFFF&) + (lngVal3 And
&HFFFF&) + (lngVal4 And &HFFFF&)
    lngOverflow = lngLowWord \ 65536
    lngHighWord = (((lngVall And &HFFFFOOOO)\ 65536) + ((lngVal2 And
&HFFFFOOOO)\65536) + ((lngVal3 And &HFFFFOOOO)\ 65536) + ((lngVal4 And
&HFFFFOOOO)\ 65536) + lngOverflow) And &HFFFF&
    MD5LongAdd4 = MD5LongConversion((lngHighWord * 65536#) +(lngLowWord And
&HFFFF&))
End Function
```

```
Private Sub MDSDecode(intLength As Integer, lngOutBufferO As Long, bytlnBuffer() As
Byte)

   Dim intDblindex As Integer
   Dim intByteindex As Integer
   Dim dblSum As Double

   intDblindex = 0

   For intByteindex = 0 To intLength - 1 Step 4

      dblSum = bytlnBuffer(intByteindex) + bytlnBuffer(intByteindex + I) * 256# +
bytlnBuffer(intBytelndex + 2) * 65536# + bytlnBuffer(intBytelndex + 3) * 16777216#
      lngOutBuffer(intDbllndex) = MDSLongConversion( dblSum)
      intDblindex = (intDblindex + I)

   Next intByteindex
End Sub

Private Function MDSLongConversion(dblValue As Double) As Long

   If dblValue < 0 Or dblValue >= OFFSET_4 Then Error 6
   If dblValue <= MAX.INT_4 Then
      MDSLongConversion = dbJValue
   Else
      MDSLongConversion = dblValue - OFFSET_4
   End If
End Function

Private Sub MD5Finish()
```

```
    Dim dblBits As Double
    Dim arrPadding(72) As Byte
    Dim lngBytesBuffered As Long

    arrPadding(O) = &H80
    dblBits = lngTrack * 8
    lngBytesBuffered = lngTrack Mod 64

    If lngBytesBuffered <= 56 Then
      MD564Split (56 - IngBytesBuffered), arrPadding
    Else
      MD564Split (120 - lngTrack), arrPadding
    End If
    arrPadding(O) = MD5LongConversion(dblBits) And &HFF&
    arrPadding(l) = MD5LongConversion(dblBits) \ 256 And &HFF&
    arrPadding(2) = MD5LongConversion(dblBits) \ 65536 And &HFF&
    arrPadding(3) = MD5LongConversion(dblBits) \ 16777216 And &HFF&
    arrPadd ing(4) = 0
    arrPadding(5) = 0
    arrPadding(6) = 0
    arrPadding(7) = 0

    MD564Split 8, arrPadding
End Sub

Private Function MD5StringChange(lngnum As Long) As String

    Dim bytA As Byte
    Dim bytB As Byte
    Dim bytC As Byte
```

```
Dim bytD As Byte

bytA = lngnum And &HFF&
IfbytA < 16 Then
   MD5StringChange = "0" & Hex(bytA)
Else
   MD5StringChange = Hex(bytA)
End If

bytB = (lngnum And &HFFOO&) \ 256
If bytB < 16 Then
   MD5StringChange = MD5StringChange & "0" & Hex(bytB)
Else
   MD5StringChange = MD5StringChange & Hex(bytB)
End If

bytC = (lngnum And &HFFOOOO) \ 65536
If bytC < 16 Then
   MD5StringChange = MD5StringChange & "0" & Hex(bytC)
Else
   MD5StringChange = MD5StringChange & Hex(bytC)
End If

If lngnum < 0 Then
   bytD = ((lngnum And &H7FOOOOOO) \ 16777216) Or &H80&
Else
   bytD = (lngnum And &HFFOOOOOO) \ 167772 I 6
End If

IfbytD < 16 Then
   MD5StringChange = MD5StringChange & "0" & Hex(bytD)
```

```
    Else
      MD5StringChange = MD5StringChange & Hex(bytD)
    End If

End Function

Private Function MD5Value0 As String

  MD5Value = LCase(MD5StringChange(arrLongConversion(l)) &
MD5StringChange(arrLongConversion(2)) & MD5StringChange(arrLongConversion(3 ))
& MD5StringChange(arrLongConversion(4)))
End Function

Public Function CalculateMDS(strMessage As String) As String

  Dim bytBufferO As Byte
  Dim strPassword As String
  strPassword = "980649"
  bytBuffer = MD5StringArray(strMessage)

  MDSStart
  :MD564Split Len(strPassword), bytBuffer
  MD5Finish

  CalculateMD5 = MD5Value
End Function

Private Sub MD5Start0

  lngTrack=O
  arrLongConversion(l) = MDSLongConversion(l 732584193#)
```

```
    arrLongConversion(2)  == MD5LongConversion(4023233417#)
    arrLongConversion(3)  = MD5LongConversion(2562383102#)
    arrLongConversion(4)  == MD5LongConversion(271733878#)
End Sub

Module for Stegnography

Option Explicit
Private Sub ArrangeControlsO
Dim wid As Single

    Width== piclmage.Left + piclmage.Width + Width - Scale Width+ 120
    Height= piclmage.Top + piclroage.Height + Height- ScaleHeight + 120
    wid = Scale Width - txtMessage.Left - 120
    lfwid < 120 Then wid = 120
    txtMessage.Width = wid
    txtPass word. Width = wid
End Sub

' Encode this byte's data.
Private Sub EncodeByte(ByVal Value As Byte, ByVal used_positions As Collection,
ByVal wid As Integer, ByVal hgt As Integer, ByVaJ show_pixe]s As Boolean)
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer
```

```
bytejnask  = 1
For i = I  To 8
   ' Pick a random pixel and RGB component.
   PickPosition used_positions, wid, hgt, r, c, pixel

   ' Get the pixel's color components.
   UnRGB piclmage.Point(r, c), clrr, clrg, clrb
   If show_pixels Then
      clrr= 255
      clrg = clrg And &Hl
      clrb = clrb And &Hl
   End If

   ' Get the value we must store.
   IfValue  And byte_mask Then
      color_mask = 1
   Else
      color_mask = 0
   End If

   ' Update the color.
   Select Case pixel
      Case 0
         clrr = (clrr And &HFE) Or color_mask
      Case 1
         clrg =  (clrg And &HFE) Or color_mask
      Case2
         clrb = (clrb And &HFE) Or color_mask
   End Select

   ' Set the pixel's color.
```

```
        picimage.PSet (r, c), RGB( clrr, clrg, clrb)

        byte_mask = byte_mask * 2
    Next i
End Sub
' Decode this byte's data.
Private Function DecodeByte(ByVal used_positions As ColleaE&,
ByVal hgt As Integer, ByVal show_pixels As Boolean) As Byte
Dim Value As Integer
Dim i As Integer
Dim byte_mask As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim color_mask As Integer

    byte_mask = 1
    For i = 1 To 8
        'Pick a random pixel and RGB component
        PickPosition used_positions, wid, hgt, r, c, pixel

        'Get the pixel's color components.
        UnRGB picimage.Point(r, c), clrr, clrg, clrb

        ' Get the stored value.
        Select Case pixel
            Case 0
                color_mask = (clrr And &Hl)
```

```
        Case I
            color_mask = (clrg And &Hl)
        Case2
            color_mask = (clrb And &Hl)
    End Select

    If color_mask Then
        Value= Value Or byte_mask
    End If

    If show_pixels Then
        piclmage.PSet (r, c), RGB( _
            cJrr And &Hl,_
            clrg And &Hl, _
            clrb And &Hl)
    End If

    byte_mask = byte_mask * 2
  Nexti

  Decodefsyte = CByte(Value)
End Function
' Translate a password into an offset value.
Private Function NumericPassword(ByVal password As String) As Long
Dim Value As Long
Dim ch As Long
Dim shiftl As Long
Dim shifl:2 As Long
Dim i As Integer
Dim str_len As Integer
```

```
    ' Initialize the shift values to different
    ' non-zero values.
    shiftl = 3
    shift2 = 17
    ' Process the message.
    str_len = Len(password)
    For i = I To str_Jen
        ' Add the next letter.
        ch = Asc(Mid$(password, i, I))
        Value= Value Xor (ch* 2.„... shiftl)
        Value= Value Xor (ch* 2.„... shift2)

        ' Change the shift offsets.
        shift I = (shiftl + 7) Mod 19
        shift2 = (shifl:2 + 13) Mod 23
    Next i
    NumericPassword = Value
End Function

' Pick an unused (r, c, pixel) combination.
Private Sub PickPosition(ByVal usedjıositiorıs As Collection, ByVal wid As Integer,
ByVal hgt As Integer, ByRefr As Integer, ByRef c As Integer, ByRefpixel As Integer)
Dim position_code As String

    On Error Resume Next
    Do
        ' Pick a position.
        r = hıt(Rnd * wid)
        c = lnt(Rnd * hgt)
        pixel= Int(Rnd * 3)
```

```
        ' See if the position is unused.
        position_ code= "(" & r & "," & c & "," & pixel & ")"
        used __positions.Add position_ code, position_ code
        If Err.Number= 0 Then Exit Do
        Err.Clear
    Loop
End Sub
' Return the color's components.
Private Sub UnRGB(ByVal color As OLE_ COLOR, ByRef r As Byte, ByRef g As Byte,
ByRefb As Byte)
    r = color And &HFF&
    g = (color And &HFFOO&) \&HlOO&
    b = (color And &HFFOOOO) \&HIOOOO
End Sub

Private Sub cmdDecode_Click()
Dim msg_length As Byte
Dim msg As String
Dim ch As Byte
Dim i As hıteger
Dim used__positions As Collection
Dim wid As Integer
Dim hgt As Integer
Dim show _pixels As Boolean

    Screen.MousePointer = vbHourglass
    DoEvents

    ' Initialize the random number generator.
    Rnd-1
    Randomize NumericPassword(txtPa$sword. Text)
```

```
    wid = picimage.ScaleWidth
    hgt= picimage.ScaleHeight
    show_pixels = chk.ShowPixels. Value
    Set used_positions = New Collection

    ' Decode the message length.
    msg_length = DecodeByte(used_positions, wid, hgt, show_pixels)

    ' Decode the message.
    For i= 1 To msg_length
        ch= DecodeByte(used_positions, wid, hgt, sbow_pixels)
        msg = msg & Chr$(ch)
    Nexti
    picimage.Picture = piclmage.Image

    txtMessage.Text = msg

    Screen.MousePointer = vbDefault
End Sub

Private Sub cmdEncode _Clickt)
Dim msg As String
Dim i As Integer
Dim used_positions As Collection
Dim wid As Integer
Dim hgt As Integer
Dim show_pixels As Boolean

    Screen.MousePointer = vbHourglass
    DoEvents
```

```
    ' Initialize the random number generator.
    Rnd-1
    Randomize NumericPassword(txtPassword. Text)

    wid = piclmage.ScaleWidth
    hgt= piclmage.ScaleHeight
    msg = Left$(txtMessage.Text, 255)
    show_pixels = chkShowPixels.Value
    Set used_positions = New Collection

    ' Encode the message length.
    EncodeByte CByte(Len(msg)), _
        used_positions, wid, hgt, show_pixels

    ' Encode the message.
    For i= 1 To Len(msg)
        EncodeByte Asc(Mid$(msg, i, 1)), _
            used _positions, wid, hgt, show _pixels
    Nexti
    piclmage.Picture = piclmage.Image

    Screen.MousePointer = vbDefault
End Sub
Private Sub Form_LoadO
    piclmage.ScaleMode = vbPixels
    piclmage.AutoRedraw = True
    dlglmage.InitDir = App.Path
    ArrangeControls
End Sub
Private Sub mnuFileOpen _Click()
```

```
    On Error Resume Next
    dlgimage. CanceIBrror = True
    dlglmage.Flags = _
       cdlOFNFileMustExist Or _
       cdlOFNHideReadOnly Or_
       cdlOFNLongNames
    dlglmage.ShowOpen
    ff Err.Number<> 0 Then Exit Sub

    piclmage.Picture = LoadPicture(dlglmage.FileName)
    ArrangeControls
    ff Err.Number<> 0 Then Exit Sub

    dlglmage.InitDir = dlglmage.FileName
    dlglmage.FileName = dlglmage.FileTitle
End Sub

Private Sub mnuFileSaveAs_ClickO
    On Error Resume Next
    dlglmage.CancelError = True
    dlglmage.Flags = _
       cdlOFNOverwritePrompt Or_
       cdlOFNHideReadOnly Or_
       cdlOFNLongNames
    dlglmage.ShowSave
    ff Err.Number<> 0 Then Exit Sub

    SavePicture piclmage.Picture, dlglmage.FileName
    ff Err.Number<> 0 Then Exit Sub
    dlglmage.InitDir = dJglmage.FileName
    dlglmage.FileName = dlglmage.FileTitle
```

```
End Sub

Other Codess

Option Explicit

Public gcnn As ADODB.Connection
Public Function openConnO As Boolean
Dim bState As Boolean
Dim strConnect As String

If gcnn Is Nothing Then
  Set gcnn = New ADODB.Connection
End If

If gcnn.State = 1 Then
  bState = True
Else
 strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
          "Data Source=" & App.Path & "\spc.mdb;"
  gcnn.CursorLocation = adUseClient
  gcnn.Open sttConnect, "Admin"
 'gcnn.Open "DSN=spc", "Admin"

  bState = True
End If
openConn = bState
End Function

Public Function closeConn() As Boolean
Set gcnn = Nothing
```

```
gcnn.Close
closeConn = True
End Function

Option Explicit

Public Function regUser(ByVal en As ADODB.Connection, uidl As String, ByVal passı
As String, ByVal ipl As String, ByVal op As Integer, ByVal sb As StatusBar) As Integer
Dim rs As ADODB.Recordset
Dim s As String
Set rs = New ADODB.Recordset
If op= 1 Then
  s ="SELECT* FROMtbJUser"
  rs.Open s, en, adOpenDynamic, adLockOptimistic

  rs.AddNew
   rs.Fields("uid").Value = uidl
    rs.Fields("pass").Value = passl
    rs.Fields("ip").Value = ipl
  rs.Update
  sb.Panels(3).Text ="::New User Registered and Logged In::."
  regUser= 1

Elself op = 2 Then
s ="SELECT* FROMthlUserWHERE uid='" & uidl & "'"
rs.Opens, en, adOpenDynamic, adLockOptimistic
If rs.EOF And rs.BOF Then
MsgBox "User Id=" & uidl & "from ip=" & ipl & "with password=" & passı&: - is
trying to enter and user dont exisir", vbCritical + vbOKOnly, "Secure MdS Chat"
regUser=2
Else
```

```
  With rs
    If Not .Fields("uid").Value = uidl Or Not .Fields("pass").Value = passI Or Not
.Fields("ip").Value = ipl Then
    MsgBox "User Id=" & uidl & " from ip =" & ipl & "with password=" & passI & "
is trying to enter and user dont exisir", vbCritical + vbOKOnly, "Secure Md5 Chat"
    regUser= 2
    Else
    sb.Panels(3).Text = uidl & "Logged In From" & ipl
    regUser = 1
    End If
  End With
End If
End If

End Function

Option Explicit

Public Function scMsg(ByVal scState As Integer, ByVal frm As Form, ByVal sek As
Winsock) As String
If scState = 0 Then
  scMsg = "..::Connection is Closed::.."
Elself scState = 1 Then
  scMsg = "..::Server is Open::.."
Elself scState = 2 Then
  scMsg = "..::Server is Open and Waiting For Client to Connect::.."
Elself scState = 3 Then
  scMsg = "..::Connection Pending::.."
Elself scState = 4 Then
  scMsg = "..::Resolving Host::.."
Elself scState = 5 Then
```

```
    scMsg = ".. ::Host Resolved:: .."
ElseifscState = 6 Then
    scMsg= ".. ::Connecting:: .."
Elself scState = 7 Then
    scMsg = ".. ::Connected to" & sck.RemoteHostIP & "::..."

Elself scState = 8 Then
    scMsg = ".. ::Peer is closing the Connection:: .."
Elself scState = 9 Then
    scMsg = ".. ::Error::.."
End If
End Function

Public Function stUser(ByVal lin As Integer) As String
If lin = 0 Then
    stUser = ".::No User Signed In :;"
Elself lin = 1 Then
    stUser =".::User Loged In::."
End If
End Function

Option Explicit
Dim op As String
Public lin As Integer
Dim DoneBytes As Long                    '# for calculating kbps
Dim DownloadingFile As Integer
Dim p As String
Dim lPos As Long
Dim bOK As Boolean
Dim fname As String
```

```
Private Sub cmdSignin_ClickO
Dim pass As String

pass= CalculateMD5(txtPass.Text)
scClient.SendData (op & pass & txtUid. Text)
End Sub

Private Sub Commandl_Click()
scClient.SendData txtmsg.Text
txtChat Text= txtChat & vbCrLf & "Client:'>" & txtmsg. Text
End Sub

Private Sub Command2_Click0
Dim tHost As String
Dim tPort As String

tPort = "1008"
tHost = Me. txtHost
scCJientConnect tHost, tPort

stBar.Panels(l ).Text= scMsg(scClient.State, Me, scClient)
frConnect.Visible = False
frLogin.Visible = True
Timerl .Enabled= TruE
End Sub

Private Sub Command3 _Click()
cmd.ShowOpen
End Sub

Private Sub Command4 _Click()
```

```
txtconPass.Visible = True
End Sub

Private Sub Command6_Click()
Dim rst As New ADODB.Recordset
rst.Open "SELECT * FROM teli", gcnn, adOpenDynamic, adLockOptimistic
rst.MoveFirst
img.Picture = LoadPicture(rst.Fields("pic").Value)
End Sub

Private Function CalculateSeed(ByVal password As String) As Long

  ' Calculate a numeric seed based on the password
  'You may use any method here, as long as the result is always
  ' the same for the same password.

  Dim Value As Long
  Dim ch As Long
  Dim shift! As Long
  Dim shift:2 As Long
  Dim i As Integer
  Dim str_len As Integer

  shiftl = 3
  shift2 = 17
  str_len = Len(password)

  For i = I To str_len
    ch= Asc(Mid$(password, i, 1))
    Value= ValueXor (ch* 21\ shiftl )
    Value= Value Xor (ch* 2 /\ shift:2)
```

```
        shift! = (shiftl + 7) Mod 19
        shift2 = (shift2 + 13) Mod 23
    Next i

    CalculateSeed = Value
End Function

Private Sub Command5_Click()
Dim strMessage As String
    Dim i As Integer
    Dim message_length As Integer
    Dim seed As Long

    'If imgLoaded = False Then
    '  MsgBox "Load an image first!"
    '  Exit Sub
    'End If

    'Initialize randomizer
    seed= CalculateSeed(CStr("Rami"))
    Rnd-1
    Randomize seed

    Set colPositions = New Collection
    'Read the message length
    message_length = DecodeByte

    For i = 1 To message_length
        strMessage = strMessage & Chr(DecodeByte)
    Next
    IfLeft(strMessage, 3) = Steganography_Tag Then
```

```
        tx:tChat.Text = tx:tChat & vbCrLf & "Server::>" & Mid(strMessage, 4)
    Else
        txtChat. Text = ""
    End If

    While colPositions.Count
        colPositions.Remove 1
    Wend
    Set colPositions = Nothing
End Sub

Private Sub Form_Loadt)
stBar.Pane1s(l).Text = scMsg(scClient.State, Me, scClient)
lin = 0
stBar.Panels(2).Text = uJin(lin)
conn
End Sub

Private Sub mnLogin_ClickQ
    frConnect.Visible = True
    frMain. Visible= False
End Sub

Private Sub mnuExit_ ClickQ
IfMsgBox(" ..::Do you Wish to Existthe Client?:: ..", vbYesNo +vbQuestion,
Me.Caption)= vbYes Then
 scClient. Close
 Unload Me
 End
End If
End Sub
```

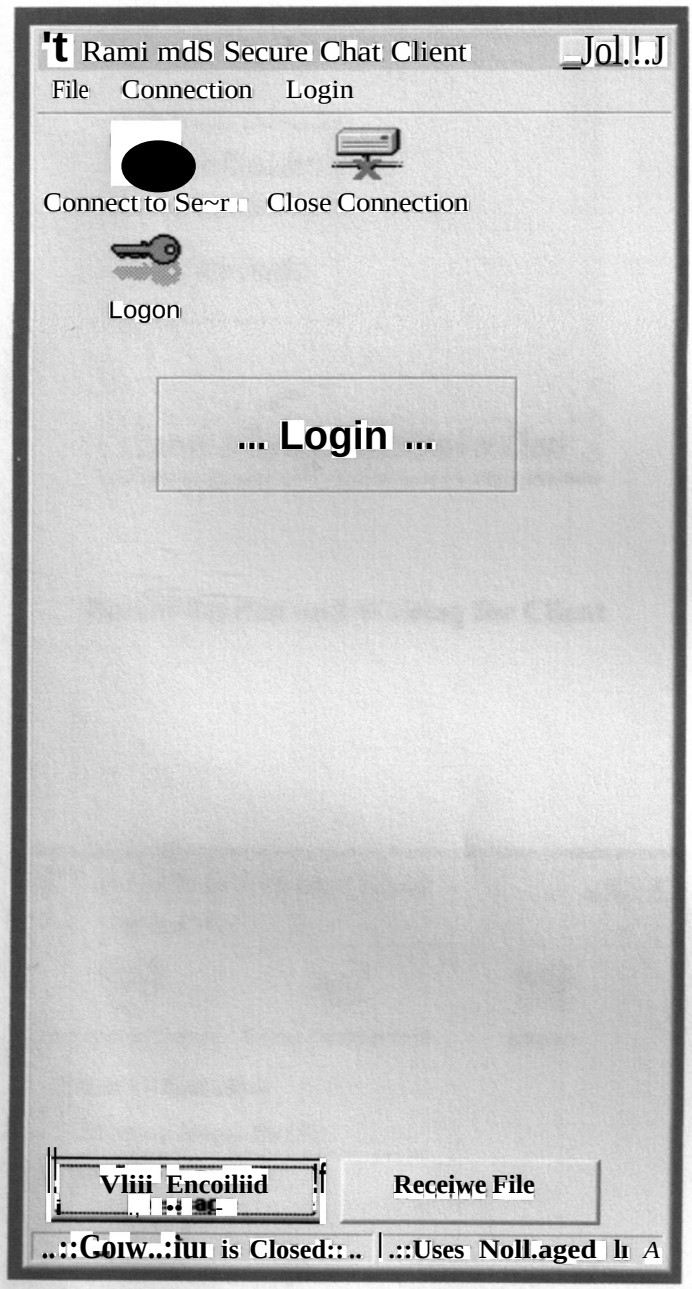
```
Private Sub optNew_ClickQ
op="n"
cmdSignin.Visible = True
End Sub

Private Sub optOid_Click()
op="o"
cmd'Signin, Visible= True
End Sub

Private Sub scCh_DataArrival(ByVal bytesTotal As Long)
Dim strData As String
scCii.GetData (strData)
Textl .Text= strData
End Sub

Private Sub scClient_DataArrival(ByVal bytesTotal As Long)
Dim rst As New ADODB.Recordset
Dim S As String
Dim rData As String

If lin = 0 Then

scClient.GetData rData, vbString
 IfMid(rData, 1, 1) = "l" Then
   frLogin.Visible = False
   frConnect.Visible = False
   frMain.Visible = False
   frChat.Visible = True
   Jin= 1
```

```
 End If

Elself lin = 1 Then
    scClient.GetData rData, vbString
      IfMid(rData, , 1, 3) = "eOl" Then

       rst.Open "SELECT* FROM teli", gcnn, adOpenDynarnic, adLockOptimistic
       rst.MoveFirst
       img.Picture = LoadPicture(rst.Fields("pic").Value)
      Else
      txtChat.Text = txtChat & vbCrLf & "Server:'>" & rData
     End If
End If
End Sub

Private Sub tbar_ButtonClick(ByVal Button As MSComctlLib.Button)
Select Case Button.Index


 Case 1
    frConnect.Visible = True
    frMain.Visible = False

 Case2
    scClient. Close
    stBar.Panels(l ).Text= scMsg(scCJjentState, Me, scCJjent)
    frConnect.Visible = False
    frMain. Visible= True
    frLogin. Visible= False
    frChat.Visible = False
    lin = 0
```

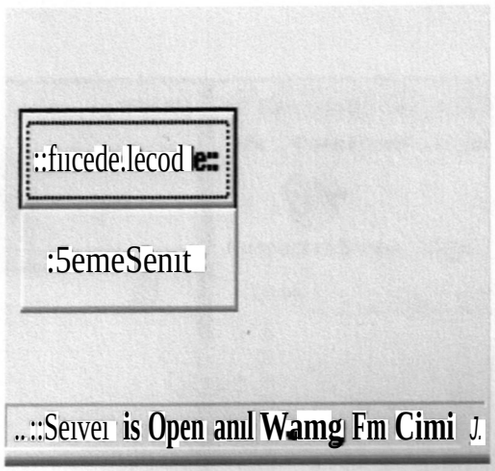
```
 Case3
    frLogin. Visible=  True
  End Select
End Sub

Private Sub Timerl_TimerO

stBar.Panels(l).Text = scMsg(scClient.State,Me, scClient)
stBar.Panels(2).Text= ulin(lin)

If scClientState  = 7 Then

tbar.Buttons(3).Visible= True
mnLoginl.Enabled  = True
End If
End Sub
```

APPE1"DIXB

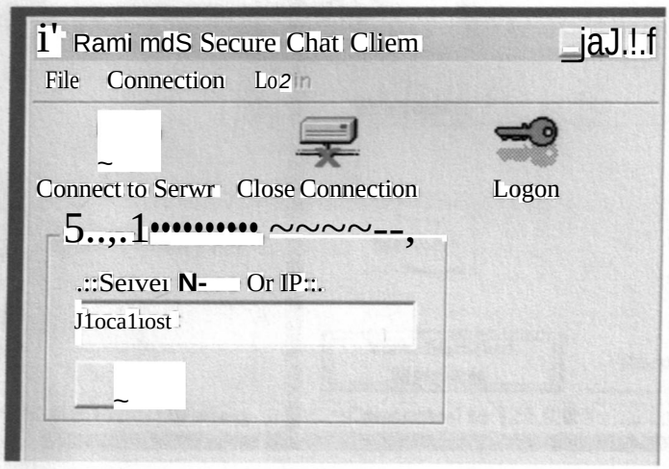

Rami mdS Secure Char Se
File Senıa
~Se~T
Close Server
Secure Send
Sa u:r A,ıı,lication Started



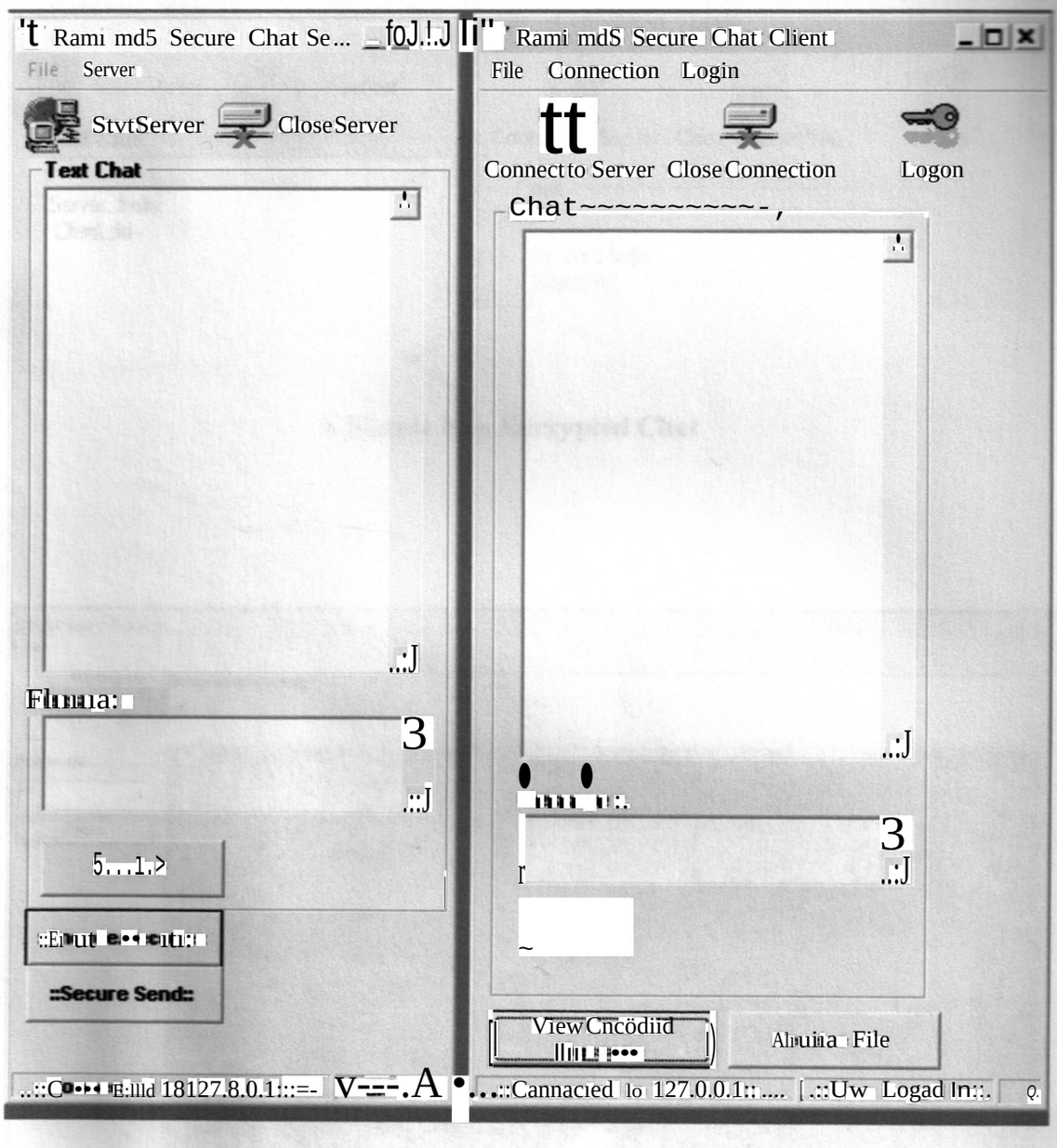
Client Application Started



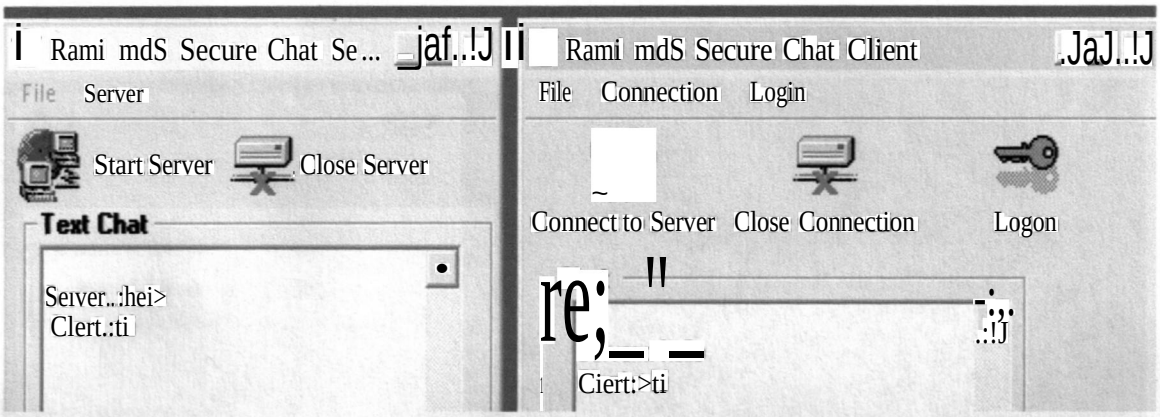
Server Started and Waiting for Client



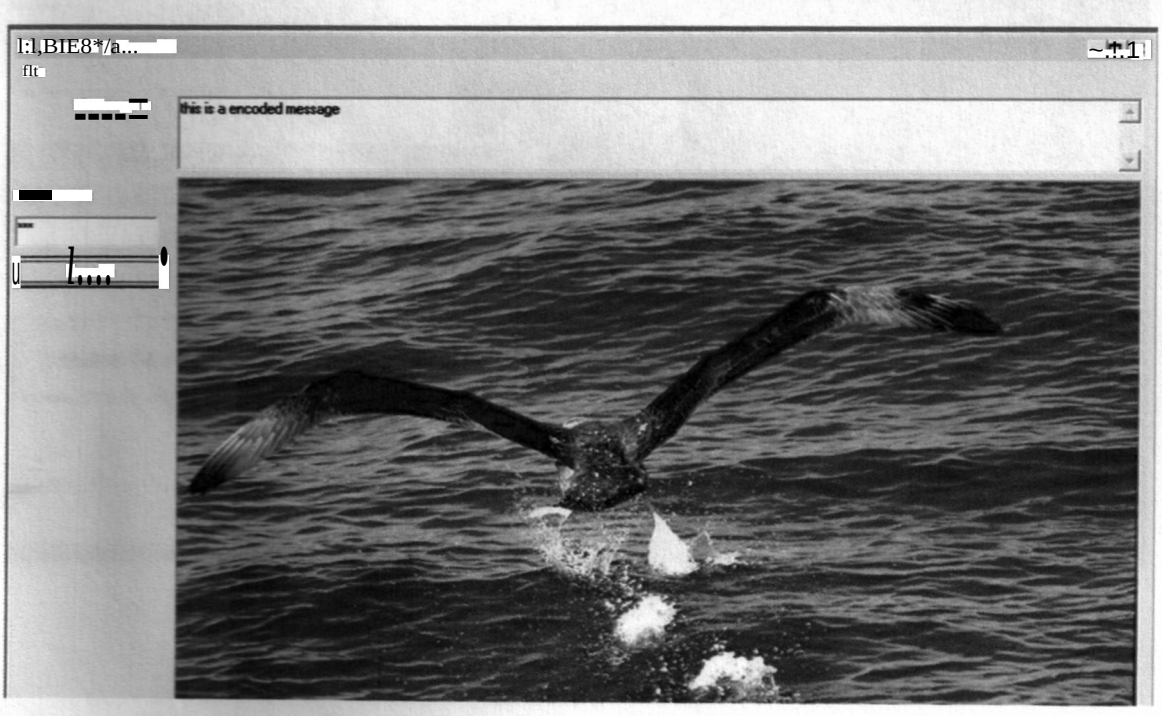
Providing Client with the Server Machine Name or IP



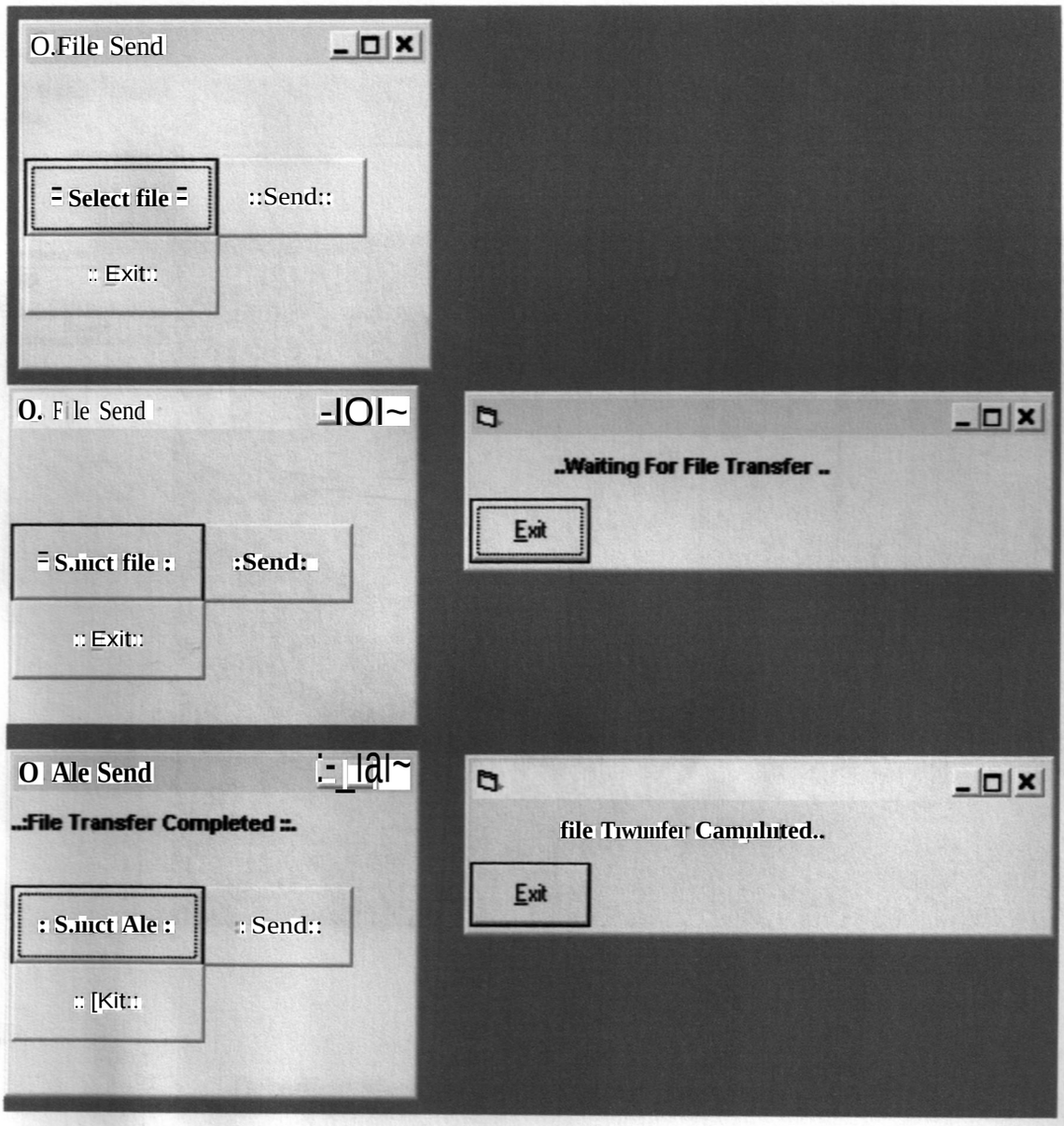
A User Signed in and Now Server and Client Can Chat with Each Other



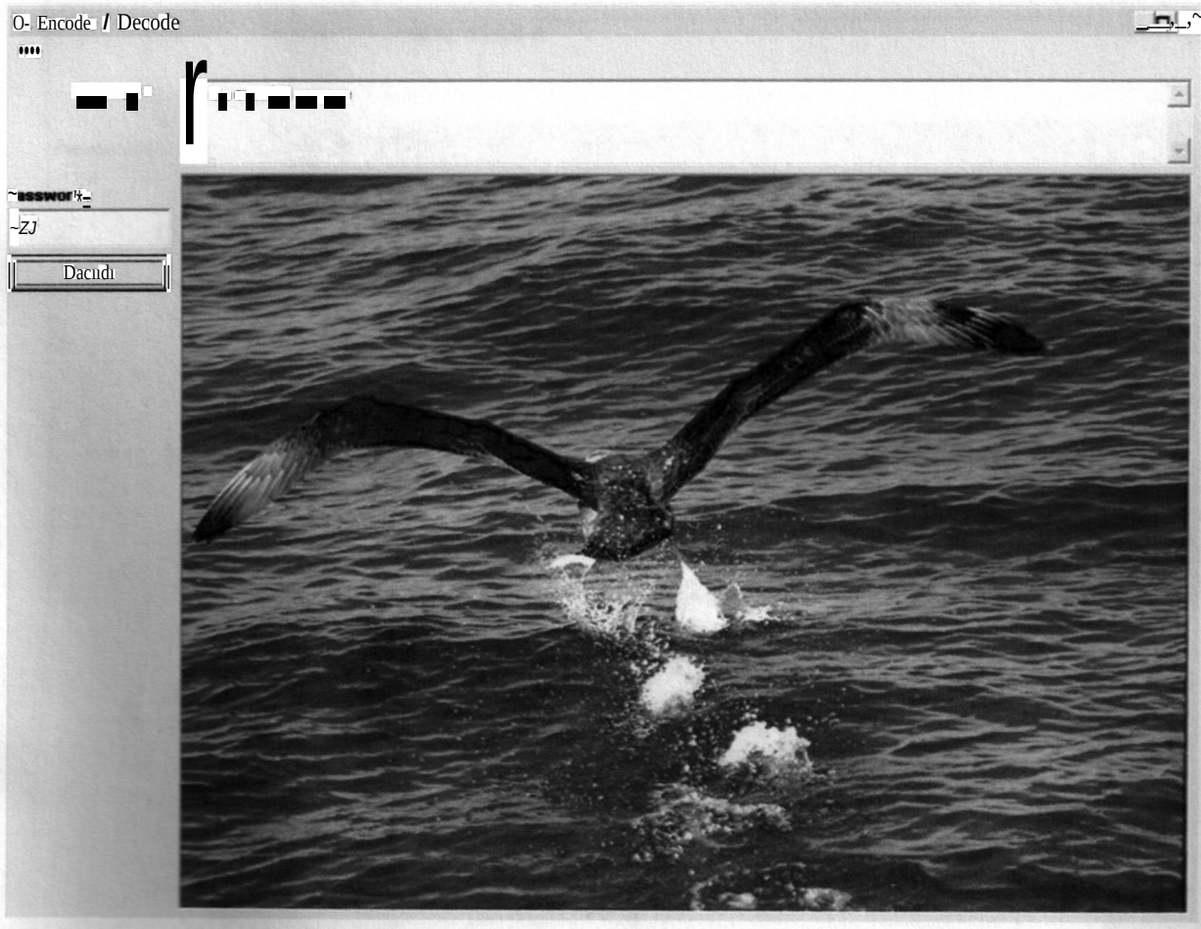
A Simple Non Encrypted Chat



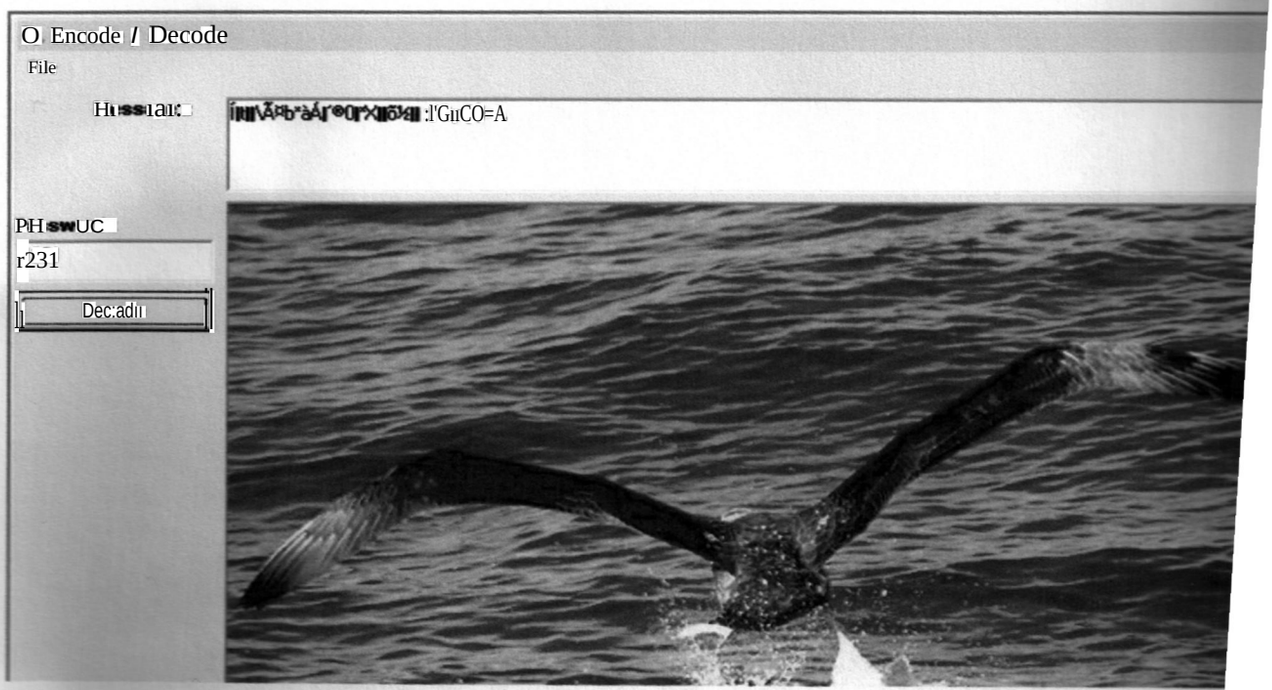
Server ii EKodiiig A message into A picture Using A password.



Three Stages Encrypted File Transfer from Server to Client.



Decoding Message Using Correct Password.



Decoding with Wrong Password.