NEAR EAST UNIVERSITY

INSTITUTE OF APPLIED AND SOCIAL SCIENCES

CRYPTOGRAPHY AND SECURITY OVER NETWORK USING DATA ENCRYPTION STANDARD

Nidal Meshal

MASTER THESIS

DEPARTMENT OF COMPUTER ENGINEERING

Nicosia 2003

Nidal Meshal : **Cryptography and Security Over Networks Using Data Encryption Standard**

Approval of the Graduate School of Applied and Social Sciences

Prof. Dr. Fakhraddin Mamedov
Director

We certify this thesis is satisfactory for the award of the Degree of Master of Science in Computer Engineering

Examining Committee in charge:

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

Assist. Prof. Dr. Doğan Haktanir, Committee Member , Electrical and Electronic Engineering Department, NEU

Assoc. Prof. Dr. Ilham Huseynov, Committee Member, Computer Information System Department, NEU

Assoc. Prof. Dr. Rahib Abiyev, Supervisor, Computer Engineering Department, NEU

ACKNOWLEDGMENT

First, I give special thanks to my supervisor Assoc.Prof.Dr.Rahib Abiyev for his valuable advice given throughout the duration of this thesis.

And thank my parents who encouraged me to continue beyond my undergraduate studies, to my father who proceeded before me and to my mother who encouraged me along the way.

To all my friends, especially Mohamed Abuhayya and Riyad Bader for sharing wonderful moments, advice, and for making me feel at home.

ABSTRACT

"How to keep Information securely?" was, is and will be one of the main questions related to most types of information, this is the reason for all those efforts have been done in this filed, our ancient grandfathers had generated the first code to secure their information, which was simple but efficient that time; the more time is going, the more new inventions in communicating data are found, the more complex the security codes.

Nowadays, many efforts have been done to secure data, keep privacy and to confidentiality in these global, wide, interfered networks either on the Internet or private network, "Cryptography" is the name which include all functions related to encryption and decryption, two types of cryptography are there; single key cryptography and public key cryptography.

This thesis will overview both types of cryptography, but will concern on the single key cryptography, after providing the history, back ground, techniques, and most of related points to this topic, the thesis will view my work on developing an application to secure data over communications channel using the technology of Data Encryption Standard (DES).

TABLE OF CONTENTS

INTRODUCTION

As the Internet is growing, the community has changed from a small tight group of academic users to a loose gathering of people on a global network, then, the need to secure application, data, and identification has come to be one of the important topics all over the world of Information technology.

After the very rapid growth of the internet technology and the internet literacy, many aspects have appeared, many problems have addressed, and of course many solutions have suggested. Who you are and where you are, both are two important questions have to be answered before any transaction to be accepted. How much you trust the medium your data is passing on is third question that has helped in finding a new technology to be sure who is sending, where the sender is, and to trust the sent data.

Fortunately, Application developers and security professionals has done their best to develop Cryptosystem Applications to be responsible for these security issues and to meet these risks by the encryption and decryption of data and authenticate and authorize users; it includes two types of cryptography, first is symmetric key cryptography which is the earlier cryptosystem, and second is asymmetric key cryptography (or public key).

This thesis discusses both types of cryptosystem and many related topics, but concerns on the symmetric key cryptography mechanism, algorithms, techniques and application, it also concerns on the Data Standard Encryption (DES) algorithm and the related functions, finally this thesis discusses the presence of cryptography in web security, showing the famous security problems, how it is solved and the role of cryptography in the solution of these problems.

This thesis also presents an application of symmetric key cryptography developed by me using the technology of Data Encryption Standard to secure data over network and the internet channels.

The aim of this thesis is to analyze the Data Encryption Standard algorithm and to apply this algorithm over network communication channel to secure the transfer of data.

This thesis includes four chapters covering the main topics related in the following structure:

Chapter 1, Will discuss the cryptography as whole; applications of cryptography, definition and types of cryptography, mechanism of public key cryptography, techniques used in cryptography and the key management process.

Chapter 2, Describes Data Encryption Standard; how it works, its basic components and some relevant topics.

Chapter 3, Begins with a discussion of the general requirements for Web security and then focuses on two standardized schemes that are becoming increasingly important as part of Web commerce: SSL/TLS and SET.

Chapter 4, Presents the developed application of symmetric cryptography based on the Simplified DES Algorithm.

Finally in conclusion the obtained important results for the thesis are given.

CHAPTER ONE
APPLICATION OF CRYPTOGRAPHY ALGORITHMS

1.1. Overview

Cryptography allows people to carry over the confidence found in the physical world to the electronic world, thus allowing people to do business electronically without worries of deceit and deception. Every day hundreds of thousands of people interact electronically, whether it is through *e*-mail, *e*commerce (business conducted over the Internet), ATM machines, or cellular phones. The perpetual increase of information transmitted electronically has lead to an increased reliance on cryptography.

This chapter will discuss the cryptography as whole; applications of cryptography, definition and types of cryptography, mechanism of public key cryptography, techniques used in cryptography and the key management process.

1.2. Application of Cryptography

Cryptography is widely used in many positions and many transactions, it is found where we need to secure our data, to securely authenticate and securely sign our electronic mail. This section will cover some of these applications.

• **Cryptography on the Internet** the Internet, comprised of millions of interconnected computers, allows nearly instantaneous communication and transfer of information, around the world. People use *e*-mail to correspond with one another. The World Wide Web is used for online business, data distribution, marketing, research, learning, and a myriad of other activities.

Cryptography makes secure web sites and electronic safe transmissions possible. For a web site to be secure all of the data transmitted between the computers where the data is kept and where it is received must be encrypted. This allows people to do online banking, online trading, and make online purchases with their credit cards, without worrying that any of their

account information is being compromised. Cryptography is very important to the continued growth of the Internet and electronic commerce.

• **E-commerce** is increasing at a very rapid rate. By the turn of the century, commercial transactions on the Internet are expected to total hundreds of billions of dollars a year. This level of activity could not be supported without cryptographic security. It has been said that one is safer using a credit card over the Internet than within a store or restaurant. It requires much more work to seize credit card numbers over computer networks than it does to simply walk by a table in a restaurant and lay hold of a credit card receipt. These levels of security, though not yet widely used, give the means to strengthen the foundation with which *e*-commerce can grow.

• **E-mail** People use *e*-mail to conduct personal and business matters on a daily basis. *E*-mail has no physical form and may exist electronically in more than one place at a time. This poses a potential problem as it increases the opportunity for an eavesdropper to get a hold of the transmission. Encryption protects *e*-mail by rendering it very difficult to read by any unintended party. Digital signatures can also be used to authenticate the origin and the content of an *e*-mail message.

• **Authentication** In some cases cryptography allows you to have more confidence in your electronic transactions than you do in real life transactions. For example, signing documents in real life still leaves one vulnerable to the following scenario. After signing your will, agreeing to what is put forth in the document, someone can change that document and your signature is still attached. In the electronic world this type of falsification is much more difficult because digital signatures are built using the contents of the document being signed.

• **Access Control** Cryptography is also used to regulate access to satellite and cable TV. Cable TV is set up so people can watch only the channels they pay for. Since there is a direct line from the cable company to each individual subscriber's home, the Cable Company will only send those channels that are paid for. Many companies offer pay-per-view channels to their subscribers. Pay-per-view cable allows cable subscribers to ``rent'' a movie directly

through the cable box. What the cable box does is decode the incoming movie, but not until the movie has been ``rented." If a person wants to watch a pay-per-view movie, he/she calls the cable company and requests it. In return, the Cable Company sends out a signal to the subscriber's cable box, which unscrambles (decrypts) the requested movie.

1.3. Basis of Cryptography

Cryptography is the process where data are encrypted and decrypted to keep it secure. Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient. Decryption is the process of transforming encrypted information so that it is intelligible again. A *cryptographic algorithm*, also called a *cipher*, is a mathematical function used for encryption or decryption. In most cases, two related functions are employed, one for encryption and the other for decryption.

With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, which is widely known, but on a number called a *key* that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple. Decryption without the correct key is very difficult, and in some cases impossible for all practical purposes [17].

1.3.1. Symmetric-key cryptography

With *symmetric-key encryption*, the encryption key can be calculated from the decryption key and vice versa. With most symmetric algorithms, the same key is used for both encryption and decryption, as shown in Figure 1.1.

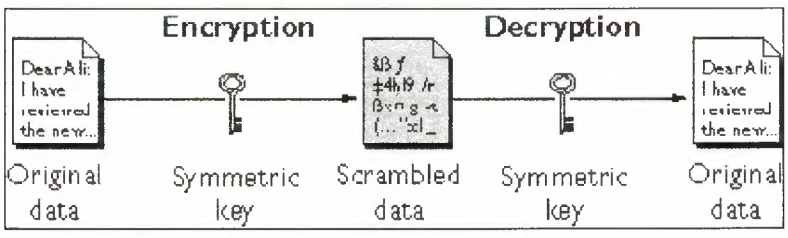


Figure 1.1 Symmetric-key cryptography

Implementations of symmetric-key encryption can be highly efficient, so that users do not experience any significant time delay as a result of the encryption and decryption. Symmetric-key encryption also provides a degree of authentication, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key. Thus, as long as the symmetric key is kept secret by the two parties using it to encrypt communications, each party can be sure that it is communicating with the other as long as the decrypted messages continue to make sense.

Symmetric-key encryption is effective only if the symmetric key is kept secret by the two parties involved. If anyone else discovers the key, it affects both confidentiality and authentication. A person with an unauthorized symmetric key not only can decrypt messages sent with that key, but can encrypt new messages and send them as if they came from one of the two parties who were originally using the key. Symmetric-key encryption plays an important role in the SSL protocol, which is widely used for authentication, tamper detection, and encryption over TCP/IP networks.

1.3.2. Public-key Cryptography

Diffie and Martin Hellman introduced the concept of public-key cryptography in 1976. Public-key cryptosystems have two primary uses, encryption and digital signatures. In their system, each person gets a pair of keys, one called the public key and the other called the private key. The public key is published, while the private key is kept secret. Figure1.2. the need for the sender and receiver to share secret information is eliminated; all communications involve only public keys, and no private key is ever transmitted or shared. In this system, it is no longer necessary to trust the security of some means of communications.

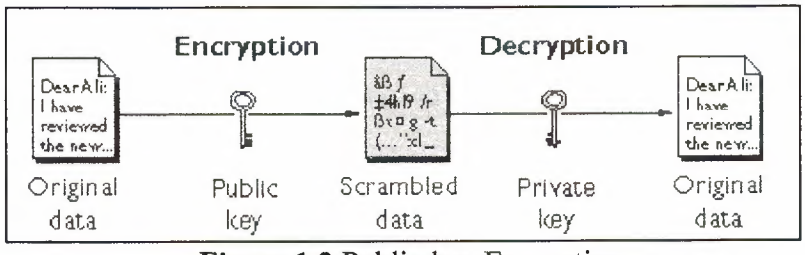


Figure 1.2 Public-key Encryption

The only requirement is that public keys be associated with their users in a trusted (authenticated) manner (for instance, in a trusted directory). Anyone can send a confidential message by just using public information, but the message can only be decrypted with a private key, which is in the sole possession of the intended recipient. Furthermore, public-key cryptography can be used not only for privacy (encryption), but also for authentication (digital signatures) and other various techniques.

In a public-key cryptosystem, the private key is always linked mathematically to the public key. Therefore, it is always possible to attack a public-key system by deriving the private key from the public key. Typically, the defense against this is to make the problem of deriving the private key from the public key as difficult as possible. For instance, some public-key cryptosystems are designed such that deriving the private key from the public key requires the attacker to factor a large number, it this case it is computationally infeasible to perform the derivation. This is the idea behind the RSA public-key cryptosystem.

Encryption When Alice wishes to send a secret message to Bob, she looks up Bob's public key in a directory, uses it to encrypt the message and sends it off. Bob then uses his private key to decrypt the message and read it. No one listening in can decrypt the message. Anyone can send an encrypted message to Bob, but only Bob can read it (because only Bob knows Bob's private key).

Digital Signatures to sign a message, Alice does a computation involving both her private key and the message itself. The output is called a digital signature and is attached to the message. To verify the signature, Bob does a computation involving the message, the purported signature, and Alice's public key. If the result is correct according to a simple, prescribed mathematical relation, the signature is verified to be genuine; otherwise, the signature is fraudulent, or the message may have been altered.

1.3.3. Public-key Cryptography versus Symmetric-key Cryptography

The primary advantage of public-key cryptography is increased security and convenience: private keys never need to be transmitted or revealed to anyone. In a symmetric-key system,

by contrast, the secret keys must be transmitted (either manually or through a communication channel) since the same key is used for encryption and decryption. A serious concern is that there may be a chance that an enemy can discover the secret key during transmission.

Another major advantage of public-key systems is that they can provide digital signatures that cannot be repudiated. Authentication via symmetric-key systems requires the sharing of some secret and sometimes requires trust of a third party as well. As a result, a sender can repudiate a previously authenticated message by claiming the shared secret was somehow compromised by one of the parties sharing the secret. For example, the Kerberos symmetric-key authentication system involves a central database that keeps copies of the secret keys of all users; an attack on the database would allow widespread forgery. Public-key authentication, on the other hand, prevents this type of repudiation; each user has sole responsibility for protecting his or her private key. This property of public-key authentication is often called non-repudiation.

A disadvantage of using public-key cryptography for encryption is speed. There are many symmetric-key encryption methods that are significantly faster than any currently available public-key encryption method. Nevertheless, public-key cryptography can be used with symmetric-key cryptography to get the best of both worlds. For encryption, the best solution is to combine public- and symmetric-key systems in order to get both the security advantages of public-key systems and the speed advantages of symmetric-key systems. Such a protocol is called a *digital envelope* [10].

Public-key cryptography may be vulnerable to impersonation, even if users' private keys are not available. A successful attack on a certification authority will allow an adversary to impersonate whomever he or she chooses by using a public-key certificate from the compromised authority to bind a key of the adversary's choice to the name of another user.

In some situations, public-key cryptography is not necessary and symmetric-key cryptography alone is sufficient. These include environments where secure secret key distribution can take place, for example, by users meeting in private. It also includes environments where a single

authority knows and manages all the keys, for example, a closed banking system. Since the authority knows everyone's keys already, there is not much advantage for some to be "public" and others to be "private." Note, however, that such a system may become impractical if the number of users becomes large; there are not necessarily any such limitations in a public-key system.

Public-key cryptography is usually not necessary in a single-user environment. For example, if you want to keep your personal files encrypted, you can do so with any secret key encryption algorithm using, say, your personal password as the secret key. In general, public-key cryptography is best suited for an open multi-user environment.

Public-key cryptography is not meant to replace symmetric-key cryptography, but rather to supplement it, to make it more secure. The first use of public-key techniques was for secure key establishment in a symmetric-key system; this is still one of its primary functions. Symmetric-key cryptography remains extremely important and is the subject of much ongoing study and research. Some symmetric-key cryptosystems are discussed in the sections on block ciphers and stream ciphers.

1.3.4. Key Size

The key size that should be used in a particular application of cryptography depends on two things. First of all, the value of the key is an important consideration. Secondly, the key size depends on what cryptographic algorithm is being used.

Due to the rapid development of new technology and cryptanalytic methods, the correct key size for a particular application is continuously changing. The table 1.1 contains key size limits and recommendations from different sources for block ciphers, the RSA system, the elliptic curve system, and DSA. Some comments:

Export grade or nominal grade gives little real protection; the key sizes are the limits specified in the Wassenaar Arrangement.

Recommendations are normally based on the traditional approach of counting MIPS-years for the best available key breaking algorithms. There are several reasons to call this approach in question. For example, an algorithm with massive memory requirements is probably not equivalent to an algorithm with low memory requirements.

The last rows in the table give lower bounds for commercial applications as suggested by Lenstra and Verheul. The first of these rows shows recommended key sizes of today, while the second row gives estimated lower bounds for 2010. The bounds are based on the assumption that DES was sufficiently secure until 1982 along with several hypotheses, which are all extrapolations in the spirit of Moore's Law (the computational power of a chip doubles every 18 months). One questionable assumption they make is that computers and memory will be able for free. It seems that this assumption is not realistic for key breaking algorithms with large memory requirements. One such algorithm is the General Number Field Sieve used in RSA key breaking efforts.

Table 1.1 Minimal key lengths in bits for different grades.

	Block Cipher	RSA	Elliptic Curve	DSA
Export Grade	56	512	112	512 / 112
Traditional	80	1024	160	1024 / 160
Recommendations	112	2048	224	2048 / 224
Lenstra/Verheul 2000	70	952	132	952 / 125
Lenstra/Verheul 2010	78	1369	146 / 160	1369 / 138

Notes. The RSA key size refers to the size of the modulus. The Elliptic Curve key size refers to the minimum order of the base point on the elliptic curve; this order should be slightly smaller than the field size. The DSA key sizes refer to the size of the modulus and the minimum size of a large subgroup, respectively (the size of the subgroup is often considerably larger in applications). In the last row there are two values for elliptic curve cryptosystems; the

choice of key size should depend on whether any significant cryptanalytic progress in this field is expected or not.

1.3.5. Key Life Cycle

Keys have limited lifetimes for a number of reasons. The most important reason is protection against cryptanalysis. Each time the key is used, it generates a number of ciphertext. Using a key repetitively allows an attacker to build up a store of ciphertext (and possibly plaintexts) which may prove sufficient for a successful cryptanalysis of the key value. Thus keys should have a limited lifetime. If you suspect that an attacker may have obtained your key, the key should be considered compromised, and its use discontinued.

Research in cryptanalysis can lead to possible attacks against either the key or the algorithm. For example, recommended RSA key lengths are increased every few years to ensure that the improved factoring algorithms do not compromise the security of messages encrypted with RSA. The recommended key length depends on the expected lifetime of the key. Temporary keys, which are valid for a day or less, may be as short as 512 bits. Keys used to sign long-term contracts for example, should be longer, say, 1024 bits or more.

Another reason for limiting the lifetime of a key is to minimize the damage from a compromised key. It is unlikely a user will discover an attacker has compromised his or her key if the attacker remains "passive." Relatively frequent key changes will limit any potential damage from compromised keys. The life cycle of any key as is:

- Key generation and possibly registration (for a public key).
- Key distribution.
- Key activation/deactivation.
- Key replacement or key update.
- Key revocation.
- Key termination, involving destruction or possibly archival.

1.3.6. Losing and Compromising of Private-Key

If your private key is compromised or lost, that is, if you suspect an attacker may have obtained your private key, then you should assume the attacker can read any encrypted messages sent to you under the corresponding public key, and forge your signature on documents as long as others continue to accept that public key as yours. The seriousness of these consequences underscores the importance of protecting your private key with extremely strong mechanisms.

You must immediately notify any certifying authorities for the public keys and have your public key placed on a certificate revocation list; this will inform people that the private key has been compromised or lost and the public key has been revoked. Then generate a new key pair and obtain a new certificate for the public key. You may wish to use the new private key to re-sign documents you had signed with the compromised or lost private key, though documents that had been time stamped as well as signed might still be valid. You should also change the way you store your private key to prevent a compromise of the new key.

1.4. Confidentiality, Authentication, Integrity and Non-repudiation

Public key cryptography schemes provide mechanisms supporting confidentiality, authenticity, integrity and non-repudiation for the network and will now be described.

1.4.1. Confidentiality

Confidentiality is sometimes called secrecy or privacy. It involves keeping a message or data private. Typically it is provided by encryption.

1.4.2. Integrity

It is a measure of the state of wholeness or goodness of the resource or the degree to which it is accurate, complete, genuine, and reliable. Typically it is provided by digital signatures in such a way that a massage or data is not alterable without detection

1.4.3. Authentication

Authentication refers to mechanisms for confirming the identiy of people, systems or information. Mechanisms include passwords, access tokens, biometrics, watermarks, and in network environment digital signatures. They ensure that the quality or condition of information is authentic, trustworthy, and genuine and that users or senders of information are who they claim to be. Authenticity is typically provided by digital signatures.

1.4.4. Non-repudiation

Non-repudiation means that a person cannot deny having by requiring the sender to digitally sign the information. At a later time a judge or a third party can establish that the sender really did send a message.

1.5. Digital Signatures

A major benefit of public key cryptography is that it provides a method for employing digital signatures. Digital signatures enable the recipient of information to verify the authenticity of the information's origin, and also verify that the information is intact. Thus, public key digital signatures provide authentication and data integrity. A digital signature also provides non-repudiation, which means that it prevents the sender from claiming that he or she did not actually send the information. These features are every bit as fundamental to cryptography as privacy, if not more.

A digital signature serves the same purpose as a handwritten signature. However, a handwritten signature is easy to counterfeit. A digital signature is superior to a handwritten signature in that it is nearly impossible to counterfeit, plus it attests to the contents of the information as well as to the identity of the signer.

Some people tend to use signatures more than they use encryption. For example, you may not care if anyone knows that you just deposited $1000 in your account, but you do want to be darn sure it was the bank teller you were dealing with.

Instead of encrypting information using someone else's public key, you encrypt it with your private key. If the information can be decrypted with your public key, then it must have originated with you.

1.6. Some Techniques in Cryptography

Cryptographic algorithms are the basic building blocks of cryptographic applications and protocols. This section presents most of the important encryption algorithms.

1.6.1. RSA

The RSA cryptosystem is a public-key cryptosystem that offers both encryption and digital signatures (authentication). Ronald Rivest, Adi Shamir, and Leonard Adleman developed the RSA system in 1977; RSA stands for the first letter in each of its inventors' last names.

1.6.2. DES

DES is an acronym for the Data Encryption Standard, is the name of the Federal Information Processing Standard (FIPS), which describes the data encryption algorithm (DEA). The DEA is also defined in the ANSI standard X3.92.

1.6.3. AES

The AES is the Advanced Encryption Standard. The AES was issued as FIPS PUB 197 by NIST (National Institute of Standards and Technology) standard is the successor to DES. In January 1997 the AES initiative was announced and in September 1997 the public was invited to propose suitable block ciphers as candidates for the AES. The AES algorithm was selected in October 2001 and the standard was published in November 2002. NIST's intent was to have a cipher that will remain secure well into the next century. AES supports key sizes of 128 bits, 192 bits, and 256 bits, in contrast to the 56-bit keys offered by DES [10].

1.6.4. DSA and DSS

The National Institute of Standards and Technology (NIST) published the Digital Signature Algorithm (DSA) in the Digital Signature Standard (DSS), which is a part of the U.S. government's Capstone project. DSS was selected by NIST, in cooperation with the NSA, to

be the digital authentication standard of the U.S. government. The standard was issued in May 1994.

DSA is based on the discrete logarithm problem and is related to signature schemes that were proposed by Schnorr and ElGamal. While the RSA system can be used for both encryption and digital signatures the DSA can only be used to provide digital signatures.

1.6.5. Elliptic Curve Cryptosystems

Elliptic curve cryptosystems were first proposed independently by Victor Miller and Neal Koblitz in the mid-1980s. At a high level, they are analogs of existing public-key cryptosystems in which modular arithmetic is replaced by operations defined over elliptic curves. The elliptic curve cryptosystems that have appeared in the literature can be classified into two categories according to whether they are analogs to the RSA system or to discrete logarithm based systems.

1.6.6. Diffie-Hellman

The Diffie-Hellman key agreement protocol (also called exponential key agreement) was developed by Diffie and Hellman in 1976 and published in the ground-breaking paper "New Directions in Cryptography." The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.

1.6.7. RC2 and RC4

RC2 is a variable key-size block cipher designed by Ronald Rivest for RSA Data Security (now RSA 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 faster than DES in software.

RC4 is a stream cipher designed by Rivest for RSA Data Security (now RSA Security). It is a variable key-size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. Analysis shows that the period of the cipher is overwhelmingly

likely to be greater than 10^{100}. Eight to sixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software. Independent analysts have scrutinized the algorithm and it is considered secure.

1.6.8 RC5 and RC6

RC5 is a fast block cipher designed by Ronald Rivest for RSA Data Security (now RSA Security) in 1994. It is a parameterized algorithm with a variable block size, a variable key size, and a variable number of rounds. Allowable choices for the block size are 32 bits (for experimentation and evaluation purposes only), 64 bits (for use a drop-in replacement for DES), and 128 bits. The number of rounds can range from 0 to 255, while the key can range from 0 bits to 2040 bits in size. Such built-in variability provides flexibility at all levels of security and efficiency.

RC6 is a block cipher based on RC5 and designed by Rivest, Sidney, and Yin for RSA Security. Like RC5, RC6 is a parameterized algorithm where the block size, the key size, and the number of rounds are variable; again, the upper limit on the key size is 2040 bits. The main goal for the inventors has been to meet the requirements of the AES.

1.6.9. SHA and SHA-1

The Secure Hash Algorithm (SHA), the algorithm specified in the Secure Hash Standard (SHS, FIPS 180), was developed by NIST). SHA-1 is a revision to SHA that was published in 1994; the revision corrected an unpublished flaw in SHA. Its design is very similar to the MD4 family of hash functions developed by Rivest. SHA-1 is also described in the ANSI X9.30 standard.

1.6.10. MD2, MD4, and MD5

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 arbitrary length and produce a 128-bit message digest. While the structures of these algorithms

are somewhat similar, the design of MD2 is quite different from that of MD4 and MD5. MD2 was optimized for 8-bit machines, whereas MD4 and MD5 were aimed at 32-bit machines.

1.7. Key Management

Key management deals with the secure generation, distribution, and storage of keys. Secure methods of key management are extremely important. Once a key is randomly generated, it must remain secret to avoid unfortunate mishaps (such as impersonation). In practice, most attacks on public-key systems will probably be aimed at the key management level, rather than at the cryptographic algorithm itself.

Users must be able to securely obtain a key pair suited to their efficiency and security needs. There must be a way to look up other people's public keys and to publicize one's own public key. Users must be able to legitimately obtain others' public keys; otherwise, an intruder can either change public keys listed in a directory, or impersonate another user. Certificates are used for this purpose. Certificates must be unforgeable. The issuance of certificates must proceed in a secure way, impervious to attack. In particular, the issuer must authenticate the identity and the public key of an individual before issuing a certificate to that individual.

If someone's private key is lost or compromised, others must be made aware of this, so they will no longer encrypt messages under the invalid public key nor accept messages signed with the invalid private key. Users must be able to store their private keys securely, so no intruder can obtain them, yet the keys must be readily accessible for legitimate use. Keys need to be valid only until a specified expiration date but the expiration date must be chosen properly and publicized in an authenticated channel.

1.7.1. Certificates

Although Alice could have sent a private message to the bank, signed it and ensured the integrity of the message, she still needs to be sure that she is really communicating with the bank. This means that she needs to be sure that the public key she is using corresponds to the bank's private key. Similarly, the bank also needs to verify that the message signature really corresponds to Alice's signature.

If each party has a certificate which validates the other's identity, confirms the public key, and is signed by a trusted agency, then they both will be assured that they are communicating with whom they think they are. Such a trusted agency is called a Certificate Authority (CA), and certificates are used for authentication.

1.7.2. Public Key Infrastructure (PKI)

A PKI (public key infrastructure) enables users of a basically unsecured public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority. The public key infrastructure provides for a digital certificate that can identify an individual or an organization and directory services that can store and, when necessary, revoke the certificates. Although the components of a PKI are generally understood, a number of different vendor approaches and services are emerging. Meanwhile, an Internet standard for PKI is being worked on [12].

The public key infrastructure assumes the use of public key cryptography, which is the most common method on the Internet for authenticating a message sender or encrypting a message. Traditional cryptography has usually involved the creation and sharing of a secret key for the encryption and decryption of messages. This symmetric or private key system has the significant flaw that if the key is discovered or intercepted by someone else, messages can easily be decrypted. For this reason, public key cryptography and the public key infrastructure is the preferred approach on the Internet.

Public Key Infrastructure Consists of:

1. A certificate authority (CA) that issues and verifies digital certificate. A certificate includes the public key or information about the public key.
2. A registration authority (RA) that acts as the verifier for the certificate authority.
3. Digital certificate is issued to a requestor.
4. One or more directories where the certificates (with their public keys) are held.
5. A certificate management system.

1.7.3 Certificate Authorities

By first verifying the information in a certificate request before granting the certificate, the Certificate Authority assures the identity of the private key owner of a key-pair. For instance, if Alice requests a personal certificate, the Certificate Authority must first make sure that Alice really is the person the certificate request claims.

1.7.3.1. Certificate Chains

A Certificate Authority may also issue a certificate for another Certificate Authority. When examining a certificate, Alice may need to examine the certificate of the issuer, for each parent Certificate Authority, until reaching one which she has confidence in. She may decide to trust only certificates with a limited chain of issuers, to reduce her risk of a "bad" certificate in the chain.

1.7.3.2. Certificate Management

Establishing a Certificate Authority is a responsibility which requires a solid administrative, technical, and management framework. Certificate Authorities not only issue certificates, they also manage them - that is, they determine how long certificates are valid, they renew them, and they keep lists of certificates that have already been issued but are no longer valid (Certificate Revocation Lists, or CRLs).

Say Alice is entitled to a certificate as an employee of a company. Say too, that the certificate needs to be revoked when Alice leaves the company. Since certificates are objects that get passed around, it is impossible to tell from the certificate alone that it has been revoked. When examining certificates for validity, therefore, it is necessary to contact the issuing Certificate Authority to check CRLs - this is not usually an automated part of the process.

1.8. Summary

Last pages covered many topics related to cryptography; the applications of cryptography and where it is used, the basics of the cryptography system and its types, and we got a comparison between the two types of cryptography , it covered specific topics like the size of the key , key life cycle and losing and compromising a key, confidentiality, authentication and digital

signature were covered too, techniques and algorithms used by any cryptography system were discussed in more details, and finally we talked about the key management and the certificate authorities. The next chapter will cover more details in the data standard encryption and how it is related to cryptography system.

CHAPTER TWO
DATA ENCRYPTOIN STANDARD

2.1. Overview

This chapter describes about Data Encryption Standard; how it works, its basic components and some relevant topics.

2.2. Block Cipher Principles

Virtually all symmetric block encryption algorithms in current use are based on a structure referred to as a Feistel block cipher. We are going to discuss the stream cipher and block cipher and Feistel cipher [1].

2.2.1. Stream Cipher

A stream cipher is a type of symmetric encryption algorithm. Stream ciphers can be designed to be exceptionally fast, much faster than any block. While block ciphers operate on large blocks of data, stream ciphers typically operate on smaller units of plaintext, it encrypts data one bit or one byte at a time usually bits. The encryption of any particular plaintext with a block cipher will result in the same ciphertext when the same key is used. With a stream cipher, the transformation of these smaller plaintext units will vary, depending on when they are encountered during the encryption process.

2.2.2. Block Cipher

A block cipher is a type of symmetric-key encryption algorithm that transforms a fixed-length block of *plaintext* (unencrypted text) data into a block of *ciphertext* (encrypted text) data of the same length. This transformation takes place under the action of a user-provided secret key. Decryption is performed by applying the reverse transformation to the ciphertext block using the same secret key. The fixed length is called the block size, and for many block ciphers, the block size is 64 bits. In the coming years the block size will increase to 128 bits as processors become more sophisticated.

When we use a block cipher to encrypt a message of arbitrary length, we use techniques known as modes of operation for the block cipher. To be useful, a mode must be at least as secure and as efficient as the underlying cipher. Modes may have properties in addition to those inherent in the basic cipher. The standard DES modes have been published in FIPS and as ANSI X3.106. A more general version of the standard generalized the four modes of DES to be applicable to a block cipher of any block size. The standard modes are Electronic Code Book, Cipher Block Chaining, Cipher Feedback, and Output Feedback.

2.2.3. Feistel Cipher

In Feistel ciphers the ciphertext is calculated from the plaintext by repeated application of the same transformation or round function. Feistel ciphers are sometimes called *DES-like ciphers.*

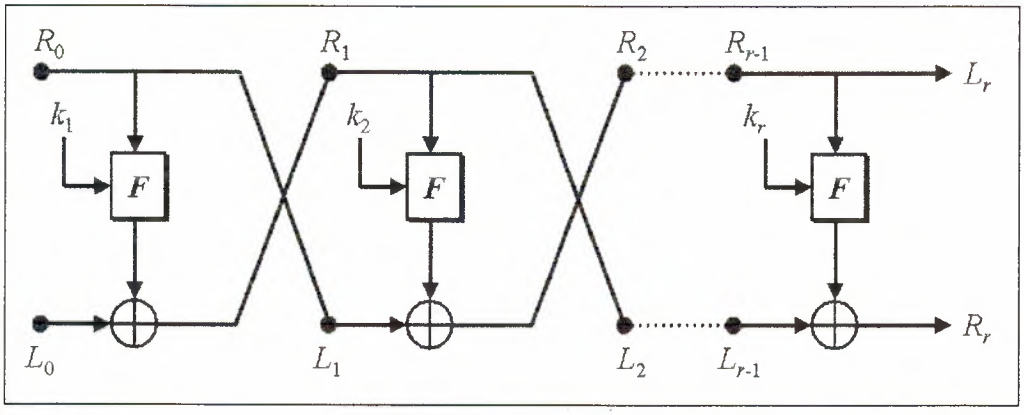


Figure 2.1 Feistel Cipher[10].

In a Feistel cipher Figure 2.1 the text being encrypted is split into two halves. The round function f is applied to one half using a subkey and the output of f is XORed with the other half. The two halves are then swapped. Each round follows the same pattern except for the last round where there is no swap. A feature of a Feistel cipher is that encryption and decryption are structurally identical, though the subkeys used during encryption at each round are taken in reverse order during decryption. The exact realization of a feistel network depends on the choice of the following parameters and design features:

- *Block size*: larger block sizes mean greater security (all other things beingequals) but reduced encryption/decryption speed. A block size of 64 bits is a reasonable tradeoff and is nearly universal in block cipher design.

- *Key size*: large key size means greater security but may decrease encryption/decryption speed. Key size of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.

- *Number of rounds*: the essence of the feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

2.3. Data Encryption Standard

Data Encryption Standard (DES) is an algorithm developed in the 1970s. by the National Bureau of Standards, now the National Institute of Standards and Technology(NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). For DES, data are encrypted in 64-bit blocks using 56-key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

2.3.1. DES Algorithm

The algorithm of DES has been made publicly available and therefore it is widely studied. There are no known trapdoors but many weaknesses are verified. The principle of DES is illustrated in figure 2.2.

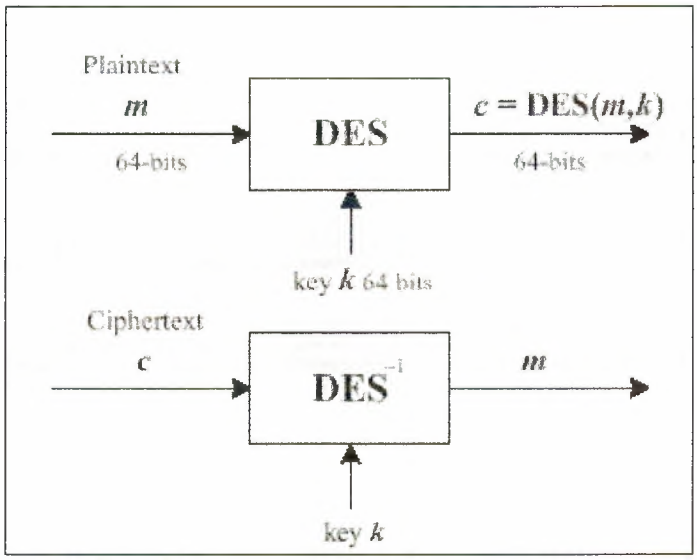


Figure 2.2 DES encryption and decryption principles.

- **DES Encryption**

The overall scheme for DES encryption is illustrated in Figure 2.3. As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.

Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an

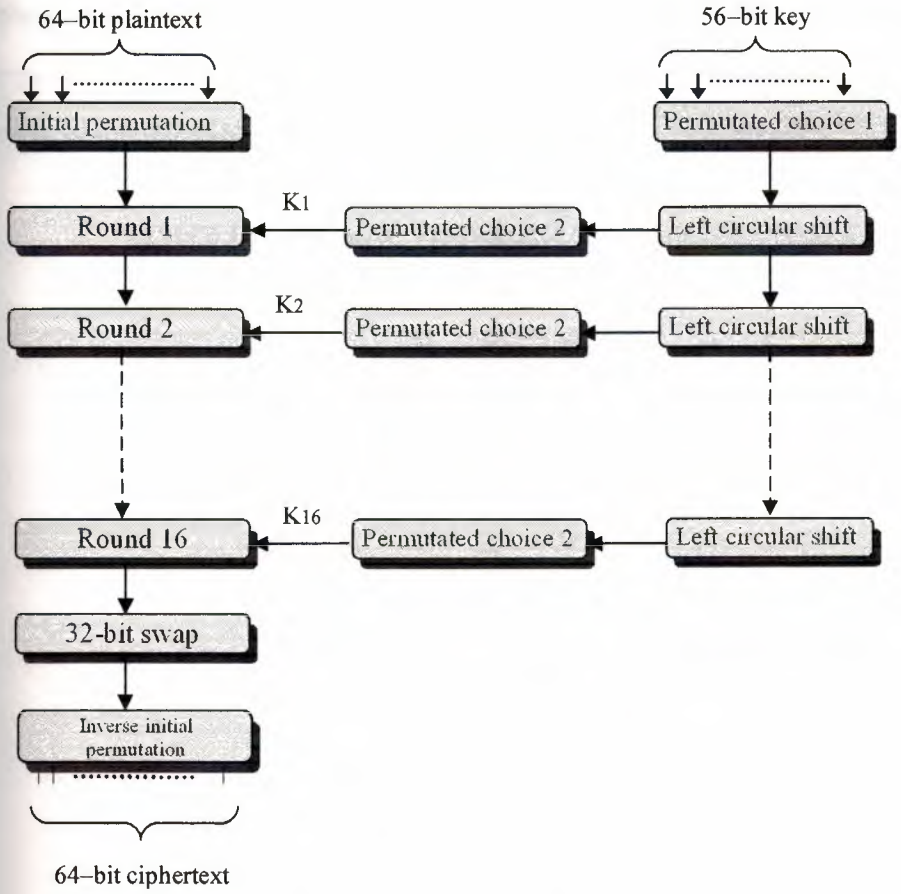


Figure 2.3 General Depiction of DES Encryption Algorithm.

Initial permutation (IP) that rearrange the bits to produce the *permuted input.* This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation arid substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext arid the key. The left and right halves of the output are swapped to produce the preoutput. Finally, the preoutput is passed through a permutation (IP^{-1}) that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.

The right-hand portion of Figure 2.3 shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the 16 rounds,

a subkey (K_i) *is* produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated iteration of the key bits [1].

Table 2.1 Permutation Table for DES. (a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
67	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	1	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

- **Initial Permutation**

The initial permutation and its inverse are defined by tables, as shown in Tables 2.1 (a) and 2.1.(b), respectively. To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:

M1	M2	M3	M4	M5	M6	M7	M8
M9	M10	M11	M12	M13	M14	M15	M16
M17	M18	M19	M20	M21	M22	M23	M24
M25	M26	M27	M28	M29	M30	M31	M32
M33	M34	M35	M36	M37	M38	M39	M40
M41	M42	M43	M44	M45	M46	M47	M48
M49	M50	M51	M52	M53	M54	M55	M56
M57	M58	M59	M60	M61	M62	M63	M64

Where, M_i is a binary digit. Then the permutation X = IP(M) is as follows:

M58	M50	M42	M34	M26	M18	M10	M2
M60	M52	M44	M36	M28	M20	M12	M4
M62	M54	M46	M38	M30	M22	M14	M6
M64	M56	M48	M40	M32	M24	M16	M8
M57	M49	M41	M33	M25	M17	M9	M1
M59	M51	M43	M35	M27	M19	M11	M3
M61	M53	M45	M37	M29	M21	M13	M5
M63	M55	M47	M39	M31	M23	M15	M7

If we take the inverse permutation $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, it can be seen that the original ordering of the bits restored.

- **Details of Single Round**

Figure 2.5 shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labelled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i).$$

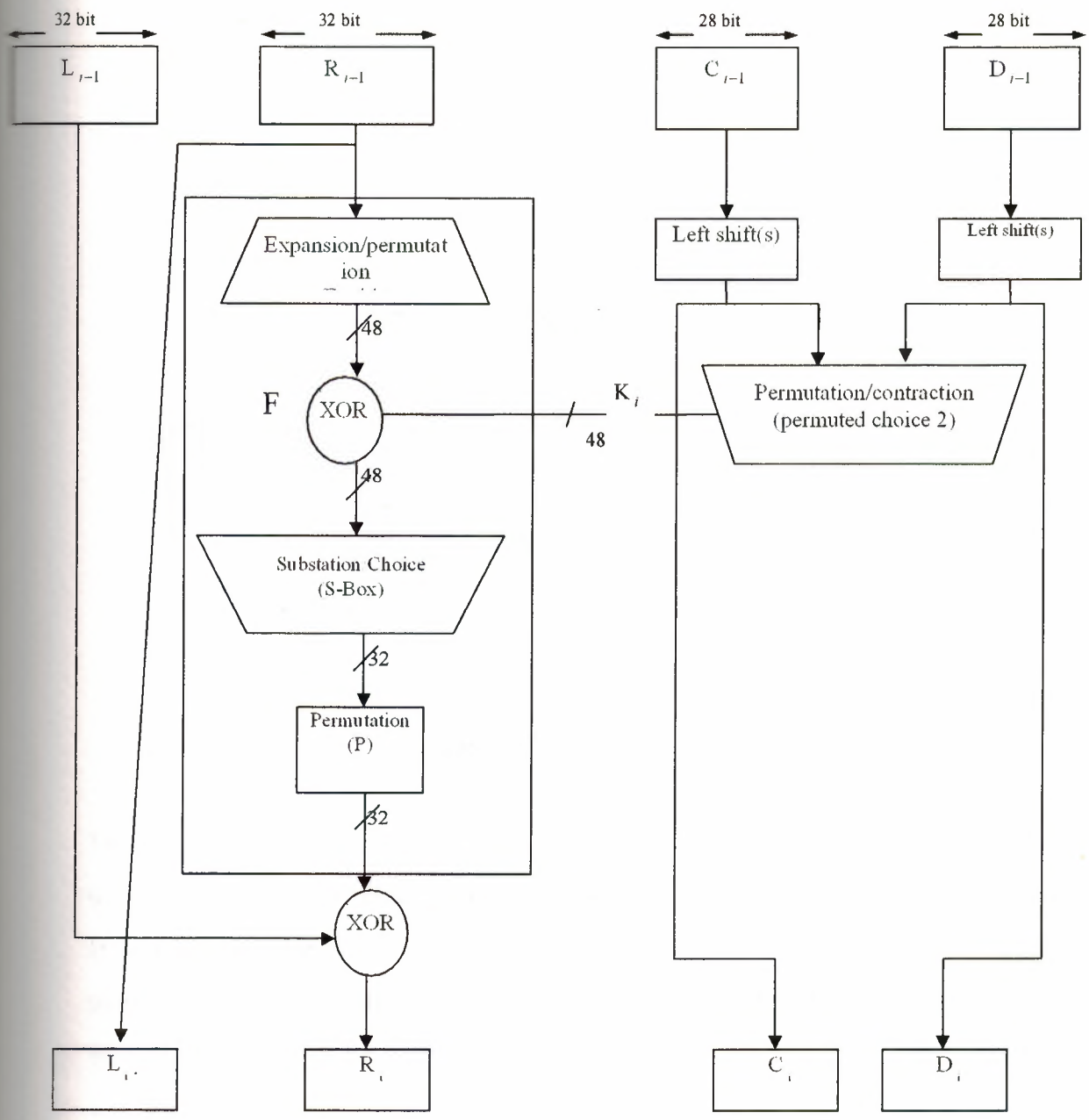


Figure 2.5 Single Round of DES algorithms

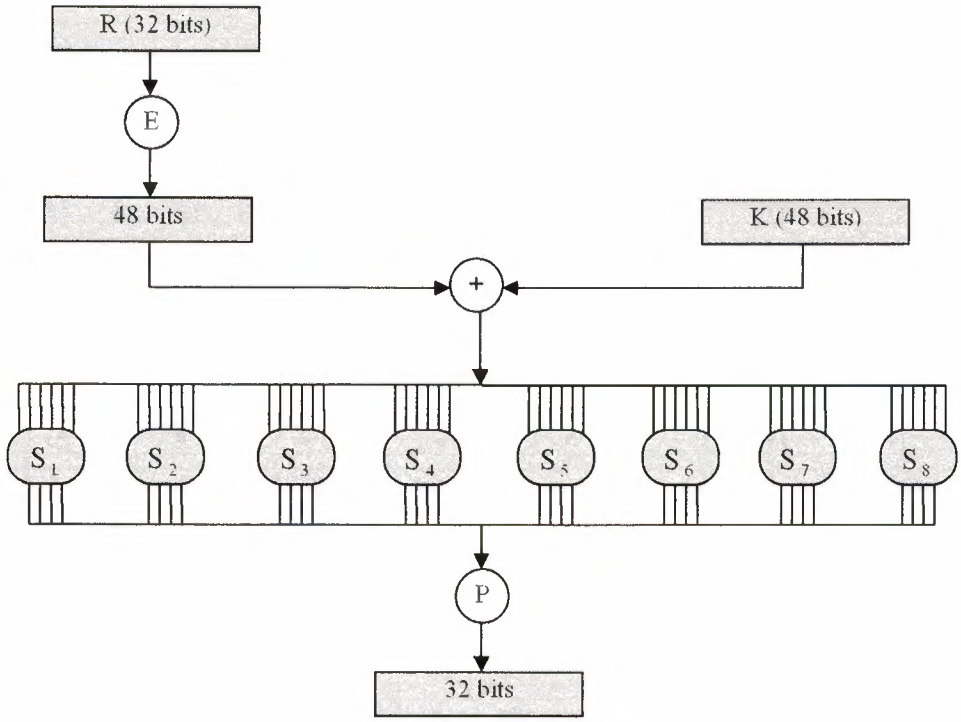


Figure 2.6 calculation F(R,K).

The round key K_i is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits (Table 2.1 c). The resulting 48 bits are X0Red with K_i. This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table 2.1 d.

The role of the S-boxes in the function **F** is illustrated in Figure 2.6. The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These- transformations are defined in Table 2.2, which is interpreted as follows: The first and last bits of the input to box S, form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si. The middle 4 bits select a particular column. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output. For example---in S, for input

011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001 [1]. Each row of an S-box defines a general reversible substitution.

Table 2.2 S-box in DES

S 1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	2	9	1	7	5	11	3	14	10	0	6	13
S 2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	4	6	10	2	8	5	14	12	11	5
	0	4	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S 3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S 4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S 5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S 6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S 7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S 8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The outer two bits of each group select one of four possible substitutions (one row of an S-box). Then a 4-bit output value is substituted for the particular 4-bit input (the middle four input bits). The 32-bit output from the eight S-boxes is then permuted, so that on the next round the output from each S-box immediately affects as many others as possible.

- **Key Generation**

Returning to Figures 2.3 and 2.4, we see that the 56-bit key used as input to the algorithm is first subjected to a permutation governed by a table labeled Per muted Choice One (Table 2.3 a). The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_0 and D_0 At each round, C_{i-1} and D_{i-1}. 1 are separately subjected to a circular left shift, or rotation, of 1 or 2 bits, as governed by Table 4.4c. These shifted values serve as input to the next round. They also serve as input to Permuted Choice Two (Table 2.3 b), which produces a 48-bit output that serves as input to the function $f(R_{i-1},K_i)$ [1].

Table 2.3 Tables Used for DES Key Schedule Calculation

(a) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(b) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(c) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	I	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

- **DES Decryption**

As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

2.4. Triple-DES

A variant of DES, triple-DES or 3DES, is based on using DES three times, normally in an encrypt-decrypt-encrypt sequence with three different, unrelated key. This is shown in figure 2.4. Since DES is not a group then the resulting ciphertext is harder to break using an exhaustive search. The only known attacks are brute-force attacks. Since 2^{56} a combination is no longer out of reach to powerful computers, triple-DES, which extends the key length to 112 bits is generally considered strongly secure.

Single DES is considered "exportable" by U.S. authorities, in other words breakable by the U.S. government. Earlier this year, 1000000 PCs were working together on the internet and they managed to break the DES in 22 hours 15 minutes. Yet, the rumor is that the national security agency of the U.S government can crack DES in 3 to 15 minutes, depending on how much preprocessing can be done. Breaking the 3DES is substantially harder, since there is need for 2^{112} attempts instead of 2^{56} attempts [1].

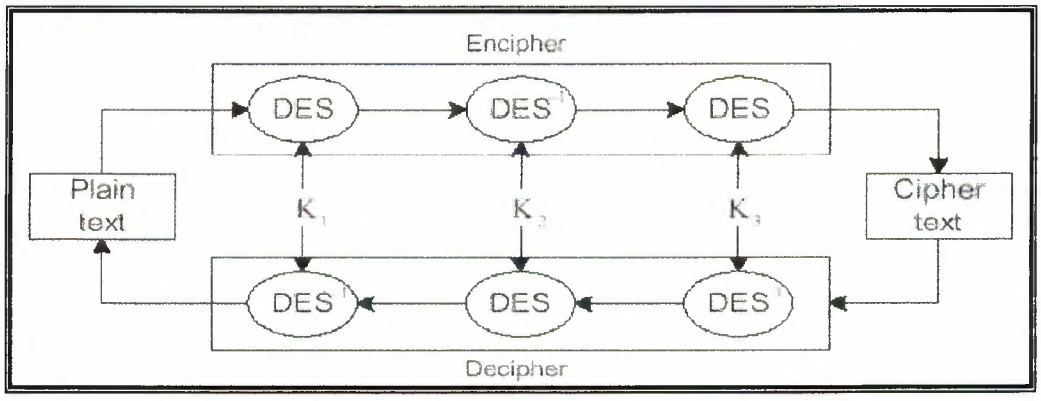


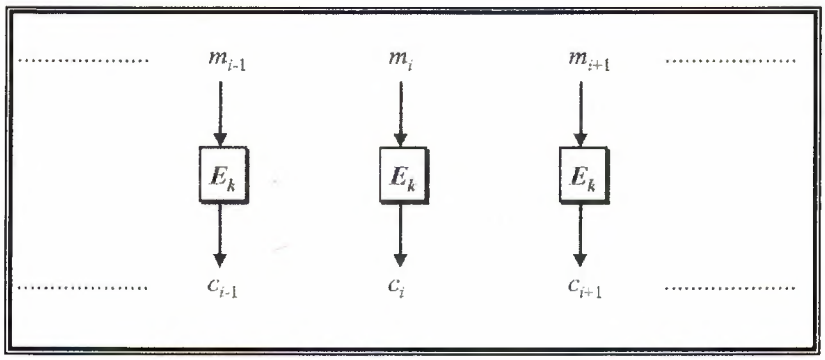
Figure 2.4 Triple-DES

2.5. Modes of Operation

The DES algorithm is a basic building block for providing data security. To apply DES in a variety of applications, four modes are intended to cover virtually all the possible applications of encryption for which DES could be used, the modes are Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB). The modes are summarized in table 2.10.

2.5.1. Electronic Codebook Mode

In ECB mode Figure 2.5, each plaintext block is encrypted independently with the block cipher.



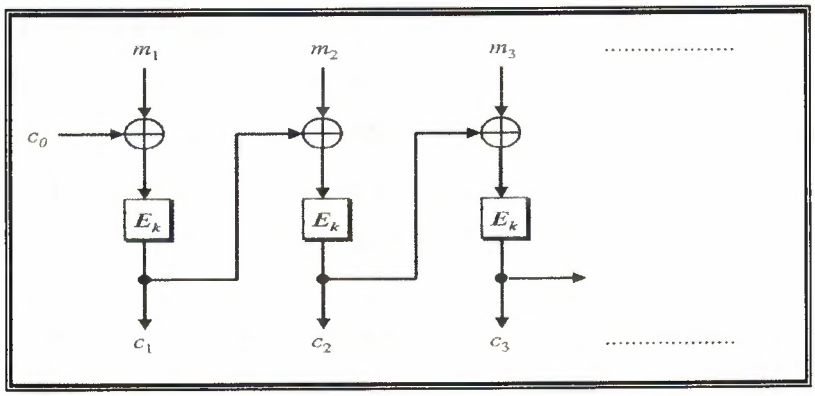
$c_i = E_k(m_i) \quad m_i = D_k(c_i)$

Figure 2.5 Electronic Code Book mode

ECB mode is as secure as the underlying block cipher. However, plaintext patterns are not concealed. Each identical block of plaintext gives an identical block of ciphertext. The plaintext can be easily manipulated by removing, repeating, or interchanging blocks. The speed of each encryption operation is identical to that of the block cipher. ECB allows easy parallelization to yield higher performance [10].

2.5.2. Cipher Block Chaining Mode

In CBC mode Figure 2.6, each plaintext block is XORed with the previous ciphertext block and then encrypted. An initialization vector c_0 is used as a "seed" for the process.



$$c_i = E_k(c_{i-1} \oplus m_i) \qquad m_i = c_{i-1} \oplus D_k(c_i)$$

Figure 2.6 Cipher Block Chaining mode.

CBC mode is as secure as the underlying block cipher against standard attacks. In addition, any patterns in the plaintext are concealed by the XORing of the previous ciphertext block with the plaintext block. Note also that the plaintext cannot be directly manipulated except by removal of blocks from the beginning or the end of the ciphertext. The initialization vector should be different for any two messages encrypted with the same key and is preferably randomly chosen. It does not have to be encrypted and it can be transmitted with the ciphertext [10].

The speed of encryption is identical to that of the block cipher, but the encryption process cannot be easily parallelized, although the decryption process can be. PCBC mode is a variation on the CBC mode of operation and is designed to extend or propagate a single bit error in the ciphertext. This allows errors in transmission to be captured and the resultant plaintext to be rejected. The method of encryption is given by

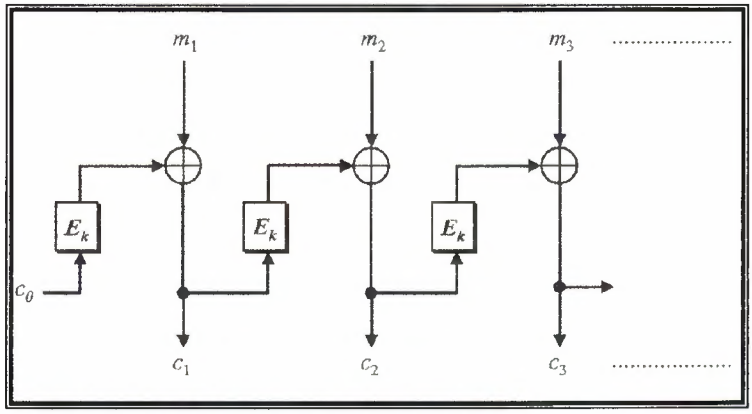
$$c_i = E_k(c_{i-1} \oplus m_{i-1} \oplus m_i)$$

and decryption is achieved by computing

$$m_i = c_{i-1} \oplus m_{i-1} \oplus D_k(c_i).$$

2.5.3. Cipher Feedback Mode

In CFB mode Figure 2.7, the previous ciphertext block is encrypted and the output produced is combined with the plaintext block using XOR to produce the current ciphertext block. It is possible to define CFB mode so it uses feedback that is less than one full data block. An initialization vector c_0 is used as a "seed" for the process.



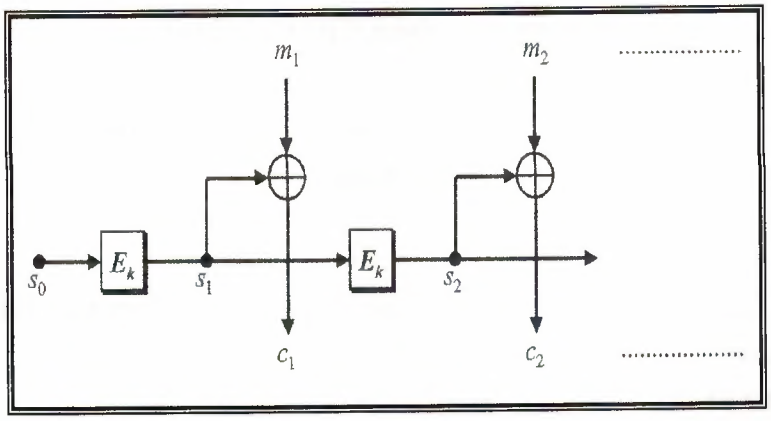
$c_i = E_K(c_{i-1}) \oplus m_i \quad m_i = E_K(c_{i-1}) \oplus c_i$

Figure 2.7 Cipher Feedback mode.

CFB mode is as secure as the underlying cipher and plaintext patterns are concealed inthe ciphertext by the use of the XOR operation. Plaintext cannot be manipulated directlyexcept by the removal of blocks from the beginning or the end of the ciphertext.

2.5.4. Output Feedback Mode

OFB mode Figure 2.8 is similar to CFB mode except that the quantity XORed with each plaintext block is generated independently of both the plaintext and ciphertext. An initialization vector s_0 is used as a ``seed" for a sequence of data blocks s_i, and each data block s_i is derived from the encryption of the previous data block s_{i-1}. The encryption of a plaintext block is derived by taking the XOR of the plaintext block with the relevant data block.



$c_i = m_i \oplus s_i \quad m_i = c_i \oplus s_i \quad s_i = E_k(s_{i-1})$

Figure 2.8 Output Feedback mode

Feedback widths less than a full block are not recommended for security. OFB mode has an advantage over CFB mode in that any bit errors that might occur during transmission are not propagated to affect the decryption of subsequent blocks. The security considerations for the initialization vector are the same as in CFB mode.

A problem with OFB mode is that the plaintext is easily manipulated. Namely, an attacker who knows a plaintext block m_i may replace it with a false plaintext block x by XORing $m_i \oplus x$ to the corresponding ciphertext block c_i. There are similar attacks on CBC and CFB modes, but in those attacks some plaintext block will be modified in a manner unpredictable by the attacker. Yet, the very first ciphertext block (that is, the initialization vector) in CBC mode and the very last ciphertext block in CFB mode are just as vulnerable to the attack as the blocks in OFB mode. Attacks of this kind can be prevented using for example a digital signature scheme or a MAC scheme [10].

Table 2.10 DES modes operation [1]

Mode	Description	Typical Application
Electronic CodeBook(ECB)	Each block of 64 plaintext bits is encoded independently using the same key	• Secure transmission of single values (eg., and encryption value)
Cipher Block Chaining(CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of cipher text	• Generation-purpose block-oriented transmission • Authentication
Cipher Feedback(CFB)	Input is processed bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is OXRed with plaintext to produce next unit of ciphertext.	• General-purpose-stream-oriented transmission • Authentication
Output feedback(OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	• stream-oriented transmission over noisy channel (e.g., satellite communication)

2.6 Summary

Last pages talked about data standard encryption as a one of the modern and effective techniques of cryptography systems, it talked about block cipher principles and Feistel block cipher, about the origins of data standard encryption, DES algorithm and 3DES, it also discussed the modes of operation. The next chapter will discuss how cryptography applications and techniques are applied in web security.

CHAPTER THREE
CRYPTOGRAPHY IN WEB SECURITY

3.1. 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. 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.

This chapter starts with web security considerations, and then it describes about some famous web threats, how to protect it and finally the main standards of web security used in electronic business and commerce.

3.2. Web security considerations

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. Web presents new challenges not generally appreciated in the context of computer and network security. The Internet is two way; 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 servers 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 vulnerable to a variety of security attacks [1].

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 local 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.

3.3. Web Security Threats

Table 3.3 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. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web 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.

Table 3.3 a comparison of threats on the web [1]

	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

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 local 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.

3.3. Web Security Threats

Table 3.3 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. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web 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.

Table 3.3 a comparison of threats on the web [1]

	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	• Misrepresentation of user • Belief that false information is valid	• Cryptographic techniques

3.4. Secure Sockets Layers (SSL) Protocol

this section covers the main features and uses of secure socket layers (SSL) protocol in some details, it talks about the main protocols involved in, and also about the advantages and disadvantages of SSL.

3.4.1. Features of SSL Protocol

SSL protocol provides confidentiality and integrity of exchanged data, and authentication of peers. Authentication is accomplished by means of public key cryptography, which is also used to exchange keys. Confidentiality and integrity of the exchanged data is based on symmetric cryptography and cryptographic checksums. SSL can be used to secure basically any TCP connections. Since it is located between the Network layer and the Application layer, the application level protocols need not be changed. HTTPS, HTTP over SSL, is the most common example. Shown in figure 3.1.

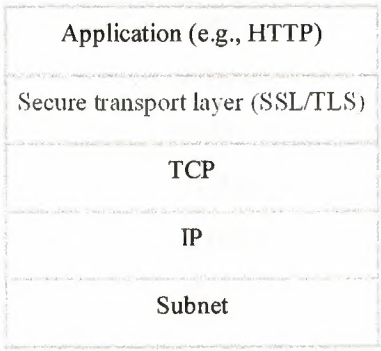


Figure 3.1 SSL runs above TCP/IP and below high-level application protocols.

The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.

These capabilities address fundamental concerns about communication over the Internet and other TCP/IP networks:

- **SSL server authentication** allows a user to confirm a server's identity. SSL-enabled client software can use standard techniques of public-key cryptography to check that a server's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the client's list of trusted CAs. This confirmation might be important if the user, for example, is sending a credit card number over the network and wants to check the receiving server's identity.

- **SSL client authentication** allows a server to confirm a user's identity. Using the same techniques as those used for server authentication, SSL-enabled server software can check that a client's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the server's list of trusted CAs. This confirmation might be important if the server, for example, is a bank sending confidential financial information to a customer and wants to check the recipient's identity.

- **An encrypted SSL connection** requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality. Confidentiality is important for both parties to any private transaction. In addition, all data sent over an encrypted SSL connection is protected with a mechanism for detecting tampering that is, for automatically determining whether the data has been altered in transit.

3.4.2. SSL Architecture

SSL 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, is shown in figure 3.2.

The SSL record protocol defines the format used to transmit data. The SSL handshake protocol involves using the SSL record protocol to exchange a series of messages between an SSL-enabled server and an SSL-enabled client when they first establish an SSL connection. [4].

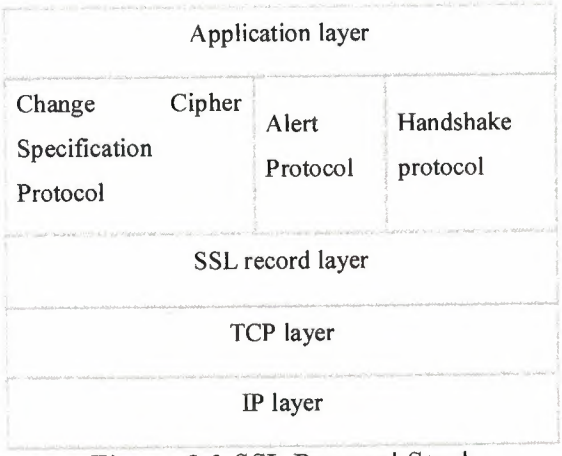


Figure 3.2 SSL Protocol Stack

3.4.2.1. Record Protocol

The SSL Record Protocol is the lowest layer in the SSL protocol. It takes a sequence of data from a higher-level protocol, fragments it to fragments of maximum 2^{14} bytes. Then it calculates a MAC, performs padding and then encrypts it with a block cipher or stream

cipher. The padding may optionally be of a random length, to make certain traffic analysis attacks more difficult to perform.

The MAC (message authentication code) before the encryption ensures the authenticity and integrity of the data. Before encryption the fragmented data may optionally be compressed, although neither the SSL3.0 nor the TLS 1.0 specifications specifies any compression methods.

3.4.2.2. Alert protocol

Alert messages are used to indicate errors discovered in the received data. It is also used to indicate that one of the peers wants to end an connection. If a connection is ended before a close notifies alert is received, it could be an indication of a truncation attack or simply a network failure. There are several severity levels of the alert messages, some are sent just as warnings, and others indicate that the session must be closed immediately and not reused [22].

3.4.2.3. Handshake protocol

The purpose of the handshake protocol is to set up a session and agree upon Cryptographic parameters. It is also optionally used to exchange certificates which authenticate the peers for each other. Public key encryption techniques are used to establish a shared secret between the peers that is used for cryptographic keys and Mac secrets. The server also authenticates itself by sending a chain of certificates to the client. Optionally the client could also authenticate itself. Throughout the handshake procedure both ends of the communication are in different phases, or states of the handshake procedure. When the handshake is finished and the normal communication can start, the state of the handshake is said to be in its finished state.

3.5. Transport Layer Security (TLS) Protocol

The TLS (Transport Layer Security) protocol is based upon the SSL3.0 (Secure Sockets Layer) protocol. The differences are rather small. SSL was a protocol designed and originally implemented by Netscape, whereas TLS is an official IETF1 standard. The most commonly used version today is SSL3.0 but newer applications are beginning to use TLS.

Older web browsers used the SSL2 protocol, for which support is now being phased out due to security flaws found in it.

3.5.1. Purpose of the TLS Protocol

TLS is a protocol designed to provide privacy and data integrity between two communicating applications. Some of the goals that the TLS protocol tries to satisfy are the following:

- Privacy, the data sent over the channel should be kept secret for an eavesdropper.
- Authentication, the applications should know that they are talking to the intended recipient and not an imposer.
- Transparency, it could be used like a normal TCP connection after the setup is done.
- Integrity, the integrity of the channel should be maintained. It should be infeasible to alter or counterfeit messages on the channel. In practice, for an user that wishes to make an online purchase on a web based store, TLS provides the following:

    1. The customer can be sure the pages he sees from the e-commerce site is delivered from the site that it says that it does, and isn't any sort of counterfeited from an imposer.
    2. The customer can be sure that the information that he sends is sent in privacy, no one can monitor the transfer and for example steal the customers' credit card number.

3.6. IP Security (IPSec)

IPSec provides the capability to secure communications across a LAN, across private and public wide area networks (WANs), and across the internet.

3.6.1. IPSec Protocols

Security protocols provide data and identity protection for each IP packet. IPSec uses the Authentication Header and Encapsulating Security Payload to provide these services.

• **Authentication Header (AH)** AH provides authentication, integrity, and anti-replay for the entire packet (both the IP header and the data carried in the packet); AH signs the entire packet. It does not encrypt the data, so it does not provide confidentiality. The data is readable, but protected from modification. AH uses HMAC algorithms to sign the packet.

For example, Alice on Computer A sends data to Bob on Computer B. The IP header, the AH header, and the data are protected from modification by the signature. This means Alice can be certain it was really Bob who sent the data and that the data was unmodified. [23]

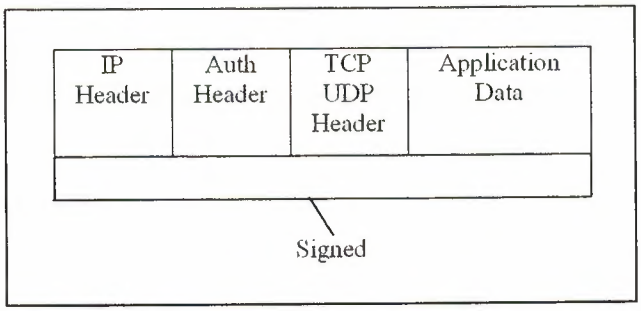


Figure 3.4 Authentication Header

Integrity and authentication are provided by the placement of the AH header between the IP header and the transport protocol header (TCP or UDP).shown in figure 3.4.

• **Encapsulating Security Payload (ESP)** ESP provides confidentiality, in addition to authentication, integrity, and anti-replay. ESP does not normally sign the entire packet unless it is being tunneled. Ordinarily, only the data is protected, not the IP header.

For example, Alice on Computer A sends data to Bob on Computer B. The data is encrypted because ESP provides confidentiality. Upon receipt, after the verification process is complete, the data portion of the packet is decrypted. Alice can be certain it was really Bob who sent the data, which data is unmodified, and that no one else was able to read it.

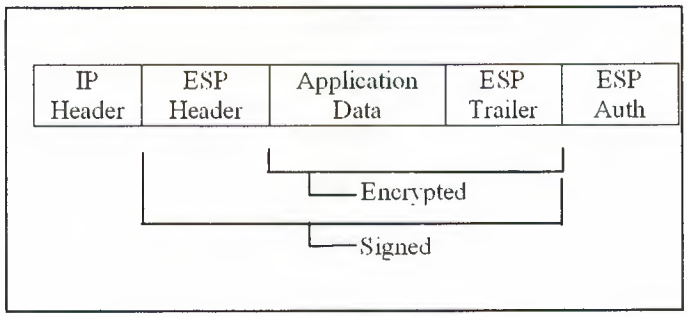


Figure 3.5 Encapsulating Security Payload

Security is provided by the placement of the ESP header between the IP header and the transport protocol header (TCP or UDP). Expert users can select which protocol will be used for a communication by configuring security methods in the IPSec policy [23].

3.6.2. IPSec Mechanism

For simplicity, this example illustrates IPSec from a domain computer to a domain computer. Alice, using an application on Computer A, sends a message to Bob. Shown figure 3.6.

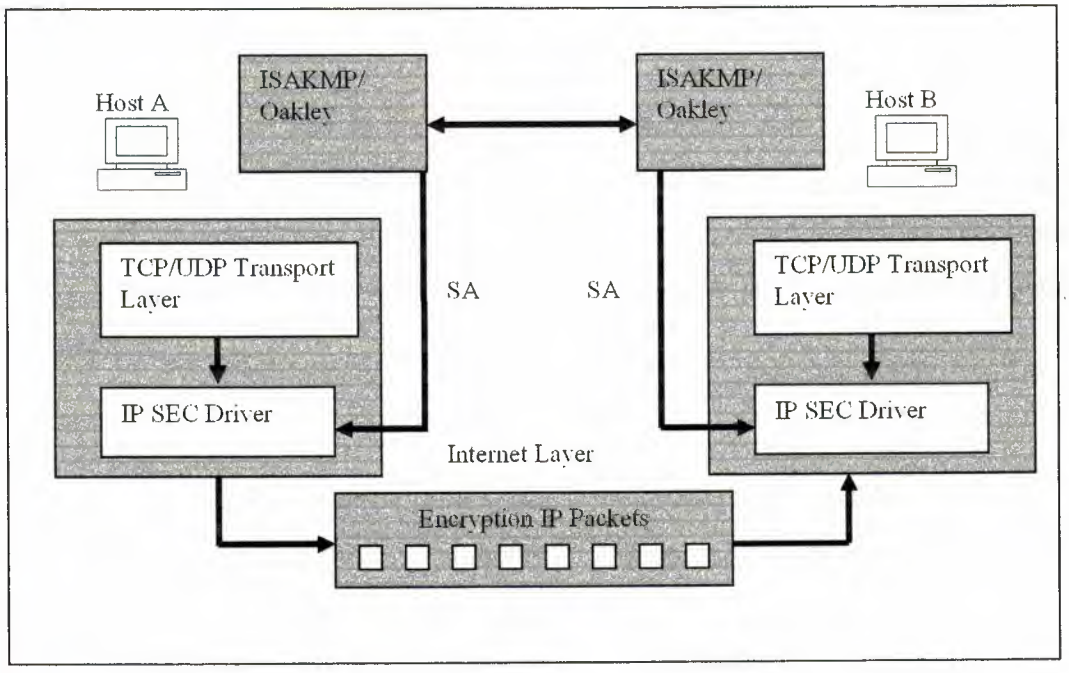


Figure 3.6 IPSec Mechanism

1. The IPSec driver on Computer A checks the IP Filter List in the active policy for a match with the address or traffic type of the outbound packets.
2. The IPSec driver notifies internet security association and key management protocol (ISAKMP) to begin security negotiations with Computer B.
3. The ISAKMP service on Computer B receives a request for security negotiations.
4. The two computers perform a key exchange, establish an ISAKMP security association (SA) and a shared, secret key.
5. The two computers negotiate the level of security for the data transmission, establishing a pair of IPSec SAs and keys for securing the IP packets.
6. Using the outbound IPSec SA and key, the IPSec driver on Computer A signs the packets for integrity, and encrypts the packets if confidentiality has been negotiated.
7. The IPSec driver on Computer A transfers the packets to the appropriate connection type for transmission to Computer B.
8. Computer B receives the secured packets and transfers them to the IPSec driver.
9. Using the inbound SA and key, the IPSec driver on Computer B checks the integrity signature and decrypts the packets, if necessary.
10. The IPSec driver on Computer B transfers the decrypted packets to the TCP/IP driver, which transfers them to the receiving application.

Alice and Bob never see any of the process. The standard routers or switches in the data path between the peers do not require IPSec. They will automatically forward the encrypted IP packets to the destination. However, if a router is functioning as a firewall, security gateway, or proxy server, you must enable special filtering to enable the secured IP packets to pass through.

3.7. Kerberos Protocol

Kerberos is a Trusted Third Party (TTP) protocol for authentication and key exchange. It is based on the Needham and Schröder authentication protocol [32]. In Kerberos, the TTP is called Key Distribution Center (KDC). KDC shares a secret key with every entity in the network and knowledge of that secret key equal's proof of identity. The Kerberos client acts on behalf of the user, which may be a person or a process. The user is also called a

principal. The principal authenticates to a verifier, which is a server; a KDC, Ticket-Granting Service (TGS), or a service. Kerberos is based on secret key cryptography. Kerberos provides single sign-on, key exchange, and a way to delegate user's credentials to back-end servers.

3.8. Secure Electronic Transaction (SET)

The SET Secure Electronic Transaction technology is an encryption technology that helps protection of the transfer of payment information over open networks, such as the Internet. SET uses advanced security technology, which allows cardholders to make secure payments to merchants on the Internet. SET technology protects payment information in four ways:

- Authenticate that a merchant is authorized to accept payment cards.
- Authenticate the payment card being used.
- Protect personal payment information.
- Payment information is read only by the intended recipient.

Message data is encrypted using randomly generated key that is further encrypted using the recipient's public key. This is referred as the "digital envelope" of the message and is sent to the recipient with the encrypted message. The recipient decrypts the digital envelope using a private key and then uses the symmetric key to unlock the original message. This protocol neither depends on transport security mechanisms nor prevents their use.

3.9. Secure Electronic Mail

One of the services most often used in distributed computer systems is the email service. It is located in the application layer within the internet protocol family. The basic protocol for electronic mail is the Simple Mail Transfer Protocol SMTP. The MIME (Multipurpose Internet Mail Extension) standard is a set of specifications that provided the exchange of text between different character sets. It allows structuring the message body into certain body parts. MIME allows the 8 bit patterns created by multimedia applications to be

appended to emails says that "The MIME specification provides a general structure for the content of type of an e-mail message and allows extensions for new content types.".

SMTP itself does not provide any security services such as confidentiality, integrity or non-repudiation; we often require the following services when using email.

- No message interception (confidentiality).
- No message interception (blocked delivery).
- No message interception and subsequent replay.
- No message content modification.
- No message origin modification.
- No message content forgery by outsider.
- No message origin forgery by outsider.
- No message content forgery by recipient.
- No message origin forgery by recipient.

These services are derived from the general security goals of message confidentiality, message integrity, sender authenticity and non-repudiation of origin, as they where partly set out earlier. There are three primary schemes for email security. These are:

- Privacy Enhanced Mail (PEM).
- Secure Multipurpose Internet Mail Extension (S/MIME).
- Pretty Good Privacy (PGP).

3.9.1 Privacy Enhanced Mail (PEM)

PEM was primarily designed to work with mail systems based on the SMTP protocol.
PEM provide the following security features when using email services:

- Data Origin Authentication.
- Message Integrity.
- Non-repudiation of origin
- Confidentiality

- Key Management

The first three are always automatically incorporated in PEM messages. Confidentiality is optional when sending PEM messages. To achieve these security goals a variety of existing algorithms can be used. For a achieving message integrity and digital signatures MD5 and MD2 could be used. For encryption DES in one valid algorithm. But PEM cannot provide access control and non-repudiation of the receipt.

PEM can be seen as the pioneering project for application of public-key cryptography in the internet. It used both, symmetric and asymmetric encryption techniques. The specification of PEM was the model for hierarchical public-key system.

3.9.2. Secure MIME (S/MIME)

S/MIME was developed to extend the existing MIME standard with the required security mechanisms. Again it refers to a specification rather than to a software package. S/MIME is based on the Public Key Cryptography Standards (PCKSs) several computer security market leaders agreed on. The algorithms used for signing and encryption are identical with those used in PEM. S/MIME is accepted and implemented by many vendors of standard browsers such as Netscape and Microsoft.

Using digital signatures and encryption, S/MIME provides

- Message origin authentication
- Non-repudiation of origin
- Message integrity and confidentiality

As in PEM many services are at the total discretion of the user. What makes S/MIME appealing to many developers is the fact that it is not limited to the SMTP protocol only. Any transport protocols such as HTTP, which support MIME objects can use these security features as well.

3.9.3. Pretty Good Privacy (PGP)

Some consider PGP as a program and that's definitely true. But when talking about PGP we often refer to its certificate infrastructure or trust model which it is called "web of trust". Although a software, the concept of PGP is about to become an internet standard.

PGP is a freeware program, only commercial users have to pay license fees. It makes use of encryption and /or digital signatures to provide

- Data confidentiality,
- Message authentication
- Data integrity
- Non-repudiation of origin

It can be used to encrypt emails and any other kind of files. Furthermore it allows for the detachment and separate transmission of digital signatures or matters of archiving the signature. PGP uses compression (standard ZIP) that makes any cryptanalysis more difficult at the same time reducing the amount of data to be transmitted.

3.10. Summary

Last chapter has covered many topics related to the web and network security considerations, has viewed some of the famous web threats, how to protect it and finally the main standards of network security.

CHAPTER FOUR
IMPLEMENTATION DES ENCRYPTION USING DELPHI

4.1. Overview

With the rapid growth of the Internet technology and its application especially in electronic commerce and electronic male systems, the need for more secure networks has become critical for surviving in this open world. As we have mentioned before cryptography systems is the solution for this problem of security and the DES encryption protocol has helped to fill that need by providing an easy to understand implementation of symmetric key encryption.

This chapter is to introduce an application developed by author to secure transaction over networks and by using single key cryptography and the S-DES protocol.

4.2. Program Explanation

What we will introduce here is how to use my application to secure your data transfer over networks, this application was developed by using DELPHI programming language and the source code of this application is found in Appendix A, the reason for using DELPHI programming language is because it is capable to develop applications over networks and the sophisticated functions it offers.

4.3. Using the Program

The usage of the application is very easy and the interface of the application that is shown in figure 4.1 and figure 4.4, by following direction stated below the user will be able to use efficiently. To complete the encryption, transfer and decryption and the program can be used for five tasks; Write message, Encryption, Send Data, Enter secret share key, and Decryption.

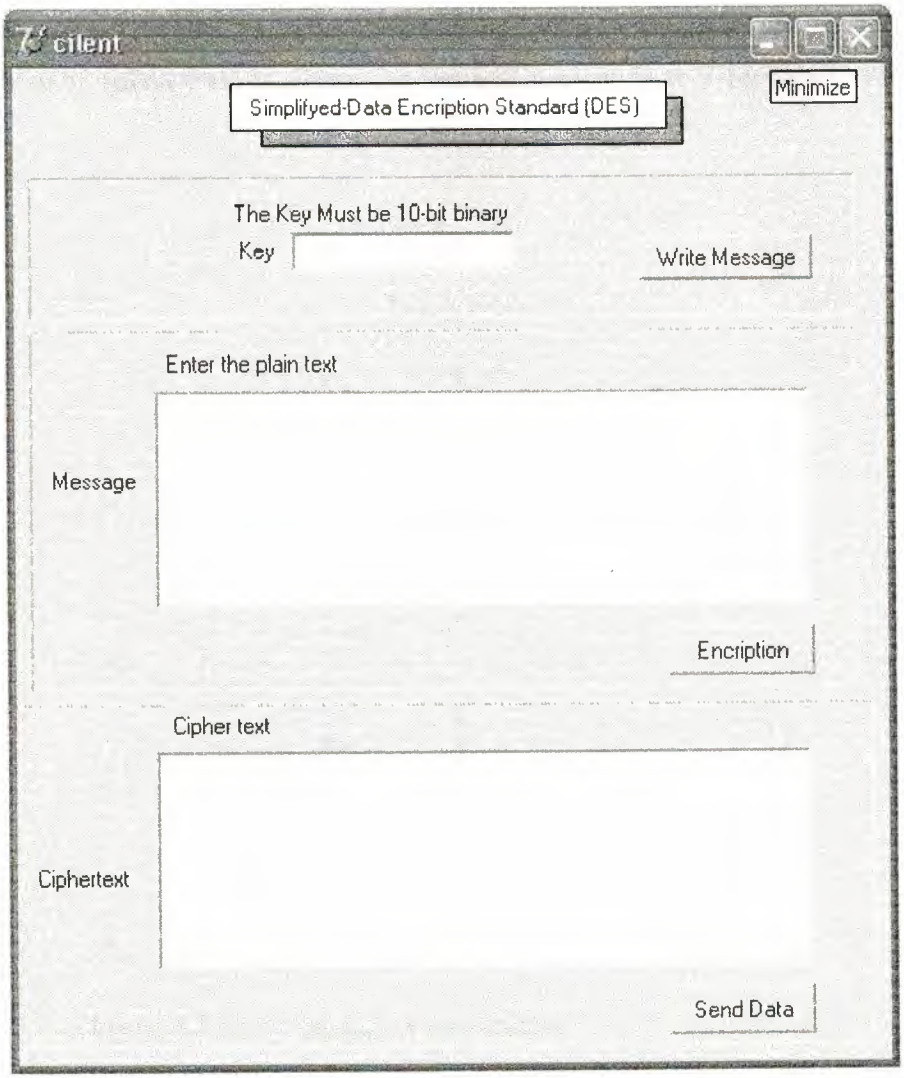


Figure 4.1 Client interface

4.3.1. Write Message

First, the user should enter the pre-shared key used in the encryption process then the user should click on (write message) button. The 10-bit key shared between sender and receiver, from this key, two 8-bit subkeys are produced for use in particular stages of the encryption and decryption algorithm.

Figure 4.2 shown the stages followed to produce the subkeys. In detail suppose we want to encrypt number 51 (plain text) to cipher text. we will convert 51 to 8-bit binary. We will get 01010001.

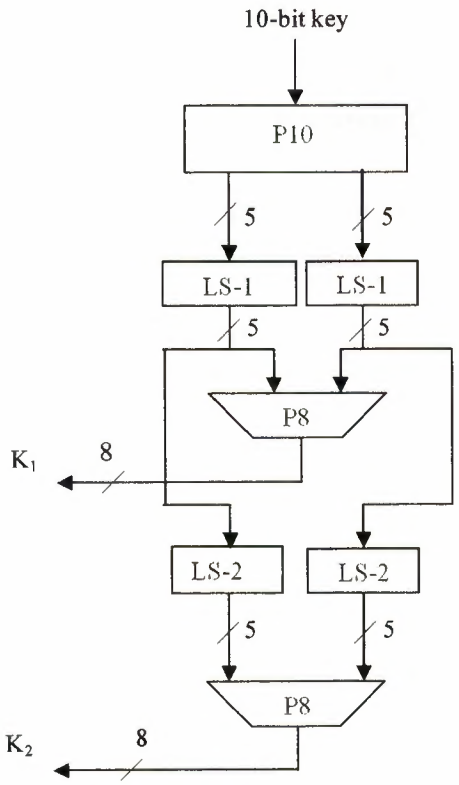


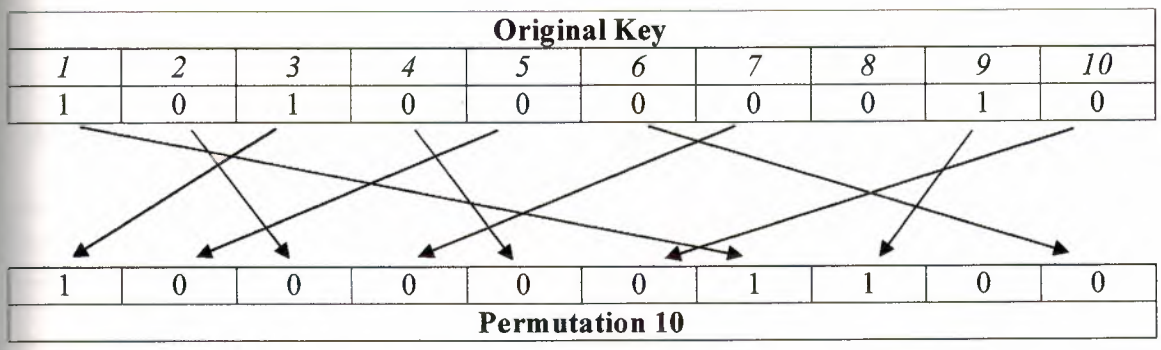
Figure 4.2 Key Generation for subkeys

First, permute the key in the following fashion. Let the 10-bit key be designated as $(k_1k_2k_3k_4k_5k_6k_7k_8k_9k_{10})$. Then the permutation P10 is defined as

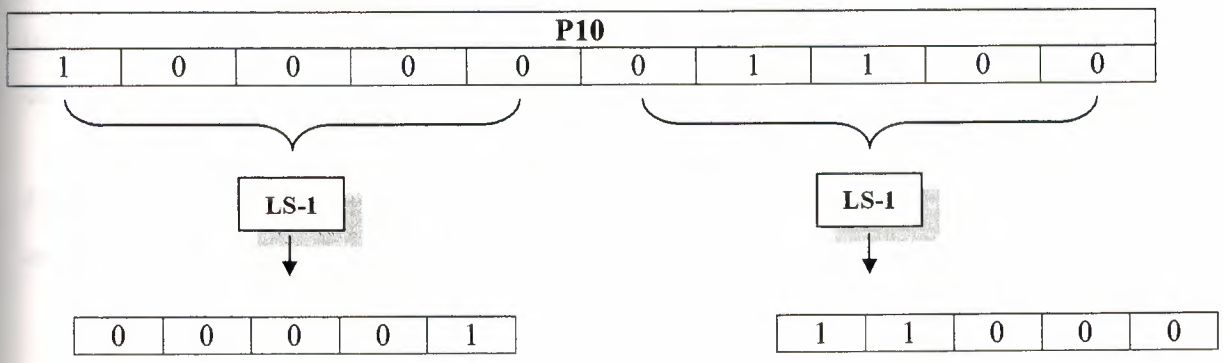
$P10(k_1k_2k_3k_4k_5k_6k_7k_8k_9k_{10}) = (k_3k_5k_2k_7k_4k_{10}k_1k_9k_8k_6)$

P10									
3	5	2	7	4	10	1	9	8	6

For example the key is (1010000010) is permuted to (1000001100)



After permutation (P10) Left Shift Operation will occur. We will divide 10 – bit key from center in two 5 –Bits.



After this we will apply Permutation 8 according to the following scheme.

P8							
6	3	7	4	8	5	10	9

The result is subkey (K1)

So now K1 is:

K_1							
1	0	1	0	0	1	0	0

Now we will apply LS-2 to the pair of 5-bit strings produced by the two LS-1 function.

0	0	1	0	0

0	0	0	1	1

After this we will again apply Permutation 8 to get 8-Bit key. So the subkey K2 will be.

K_2							
0	1	0	0	0	0	1	1

4.3.2. Encryption

By clicking one the "Encrypt" button, the plain text will convert into cipher text; actually, the pre-shared key entered before will be used to generate the two subkeys used in the encryption of the text have been entered by the user , the application uses S-DES algorithm to encrypt the data.

Figure 4.3 shows the S-DES encryption algorithm in greater detail. As was mentioned, encryption involves the sequential application of five functions. We examine each of these.

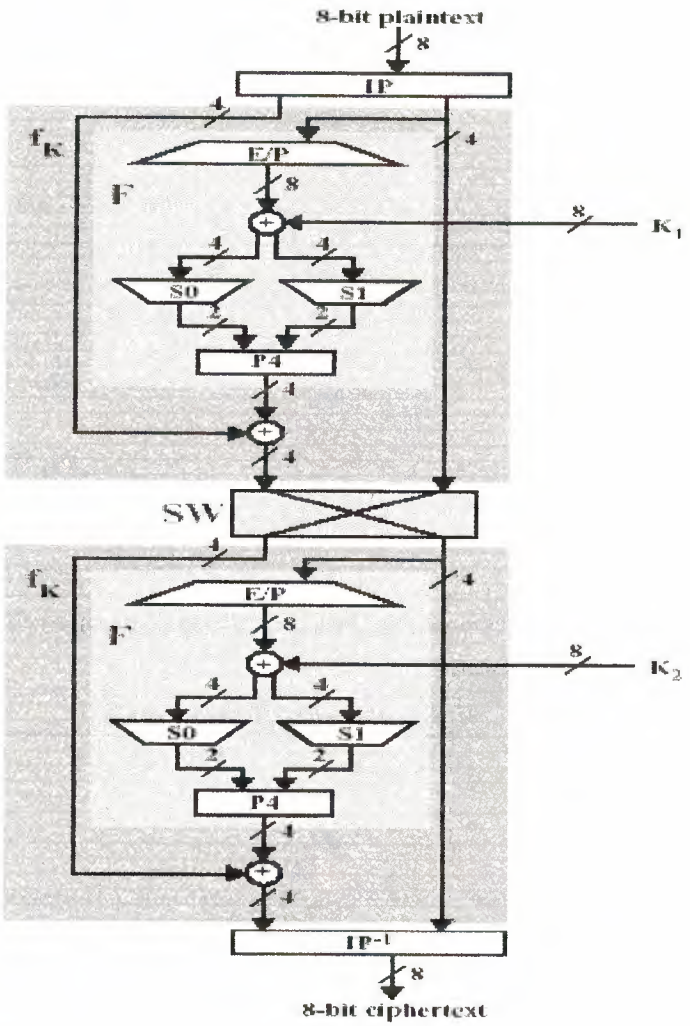


Figure 4.3 Simplified DES Scheme Encryption Detail [1].

Suppose that we have now:

K_1							
1	0	1	0	0	1	0	0
K_2							
0	1	0	0	0	0	1	1
Plain Text							
0	1	0	1	0	0	0	1

The input of the algorithm is an 8-bit block of plaintext, which we first permute using the IP function:

IP							
2	6	3	1	4	8	5	7

We will get,

1	0	0	0	1	1	0	0

L R

Now we will consider **R** part.

1	1	0	0

On this we will apply Expansion Permutation, and the manner will be:

Expansion Permutation E/P							
4	1	2	3	2	3	4	1

We will get,

After E/P							
0	1	1	0	1	0	0	1

Now we will apply **Exclusive-OR** (XOR) operation with the input $\mathbf{K_1}$

E/P	0	1	1	0	1	0	0	1	
$\mathbf{K_1}$	1	0	1	0	0	1	0	0	**XOR**
	1	**1**	**0**	**0**	**1**	**1**	**0**	**1**	

Now we will feed this data in our Shift S-boxes (S0 and S1). For this we will divide 8 bit into two halves.

1	1	0	0

S0

1	1	0	1

S1

S0		0	1	2	3	
0		1	0	3	2	
1		3	2	1	0	
2		0	2	1	3	
3		3	1	3	2	

S1		0	1	2	3	
0		0	1	2	3	
1		2	0	1	3	
2		3	0	1	0	
3		2	1	0	2	

To read **S0** matrix we will apply operation in following manner:

The S-box operate as follows: the first and fourth input bits are treated as

2bit numbers that specify a row of the s-box, and the second and third input bits specify a column of the s-box.

For Column (Bit 2, Bit 3) = (10) = **2**

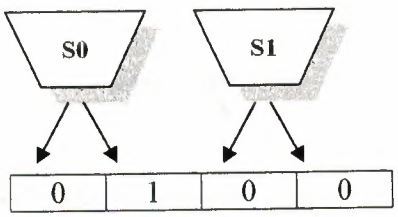
So now we have **1** which is in binary equal to **01** (2 bit).

And for **S1**,

For Row (Bit 1, Bit 4) = (11) = **3** (in binary) and

For Column (Bit 2, Bit 3) = (10) = **2**

So now we have **0** which is in binary equal to **00** (2 bit).



Now we will apply **Permutation 4** (P4),

P4			
2	4	3	1

Result will be,

1	0	0	0

The output of **P4** is the output of the function **F**

After this we have to apply Exclusive-OR operation with the left side.

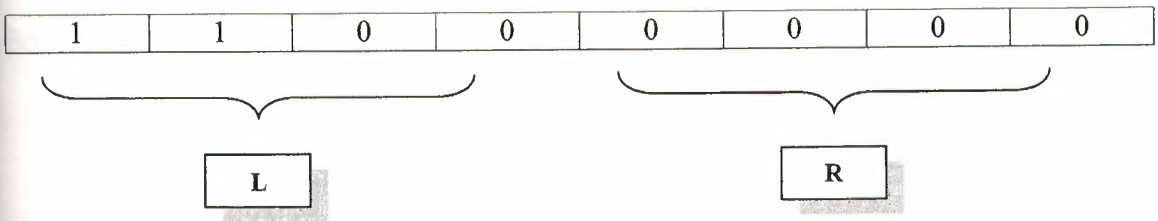
1	0	0	0	
1	0	0	0	**XOR**
0	**0**	**0**	**0**	

Now we will combine this with right part (R).

0	0	0	0	1	1	0	0

After applying the **switch function**

The function f_k only alters the leftmost 4 bits of the input. The switch function (SW) Interchanges the left and right 4 bits. In the second instance, the E/P, S0,S1, and P4 functions are the same. The key input is K2.

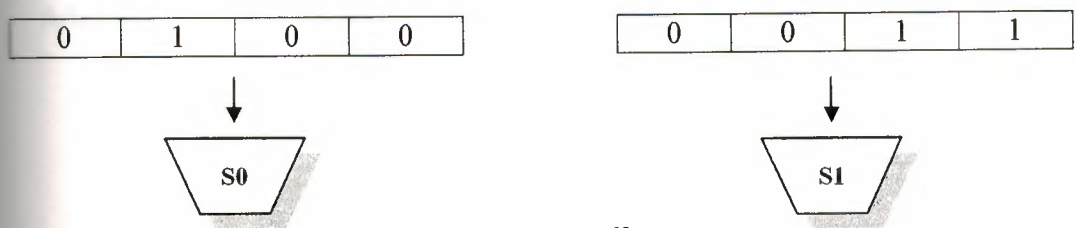


Now on right part **(R)** we will again apply Expansion Permutation **(E/P)**. We will get,

After E/P							
0	0	0	0	0	0	0	0

Now we will apply Exclusive-OR (**XOR**) operation using Key 2 ($\mathbf{K_2}$).

E/P	0	0	0	0	0	0	0	0	
$\mathbf{K_2}$	0	1	0	0	0	0	1	1	**XOR**
	0	**1**	**0**	**0**	**0**	**0**	**1**	**1**	



After this we have to apply Exclusive-OR operation with the left side.

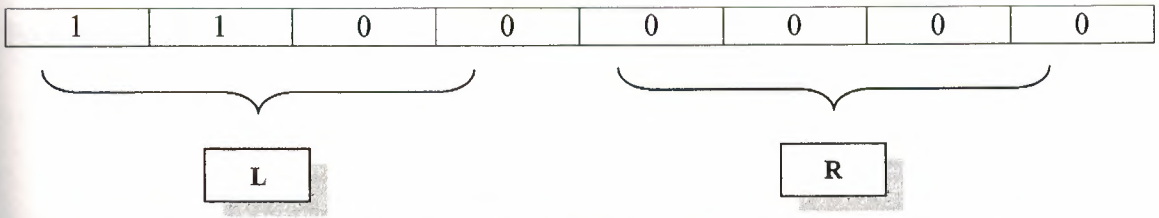
1	0	0	0	
1	0	0	0	**XOR**
0	**0**	**0**	**0**	

Now we will combine this with right part (R).

0	0	0	0	1	1	0	0

After applying the **switch function**

The function f_k only alters the leftmost 4 bits of the input. The switch function (SW) Interchanges the left and right 4 bits. In the second instance, the E/P, S0,S1, and P4 functions are the same. The key input is K2.

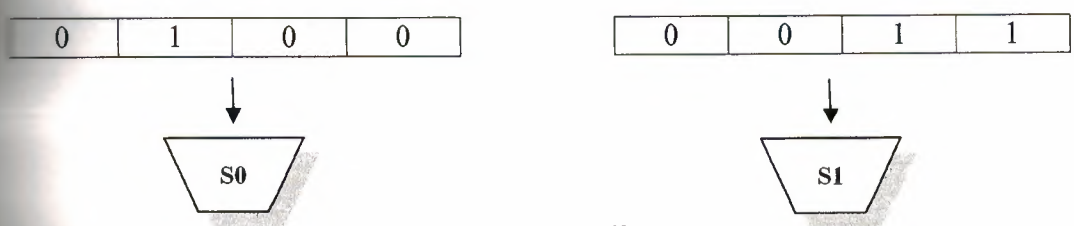


Now on right part **(R)** we will again apply Expansion Permutation **(E/P)**. We will get,

After E/P							
0	0	0	0	0	0	0	0

Now we will apply Exclusive-OR (**XOR**) operation using Key 2 ($\mathbf{K_2}$).

E/P	0	0	0	0	0	0	0	0	
K_2	0	1	0	0	0	0	1	1	**XOR**
	0	**1**	**0**	**0**	**0**	**0**	**1**	**1**	



S0		0	1	2	3	
0		1	0	3	2	
1		3	2	1	0	
2		0	2	1	3	
3		3	1	3	2	

S1		0	1	2	3	
0		0	1	2	3	
1		2	0	1	3	
2		3	0	1	0	
3		2	1	0	2	

To read **S0** matrix we will apply operation in following manner:

For Row (Bit 1, Bit 4) = (00) = **0** (in binary) and

For Column (Bit 2, Bit 3) = (10) = **2**

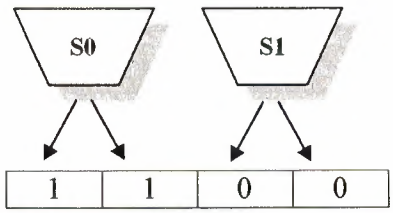
So now we have **3** which is in binary equal to **11** (2 bit).

And for **S1**,

For Row (Bit 1, Bit 4) = (01) = **1** (in binary) and

For Column (Bit 2, Bit 3) = (00) = **1**

So now we have **0** which is in binary equal to **00** (2 bit).



Now we will apply **Permutation 4** (P4),

P4			
2	4	3	1

Result will be,

1	0	0	1

After this we have to apply Exclusive-OR operation with the left side.

1	0	0	1	
1	1	0	0	**XOR**
0	**1**	**0**	**1**	

Now we will combine this with right part (R).

0	1	0	1	0	0	0	0

After this we will apply Initial Permutation^{-1} (IP^{-1})

IP^{-1}							
4	1	3	5	7	2	8	6

So we will get the cipher text,

1	0	0	0	0	1	0	0

4.3.3. Send Data

The "send data" button is responsible for sending data from client to the server using CORBA technology; this technology using broadcast technique to send data to all networked computers and the server only will response to the message and be able to decrypt the sent data. At this point he decryption will be done in the sever side in the next step.

4.3.4. Enter Secret Shared Key

After receiving the cipher text from the client, the user should enter the 10-bit secrete sh key shown in Figure 4.4 used in the decryption process. The 10-bit key shared key between sender and receiver, from this key, two 8-bit subkeys are produced for use in particular stages of the encryption and decryption algorithm.

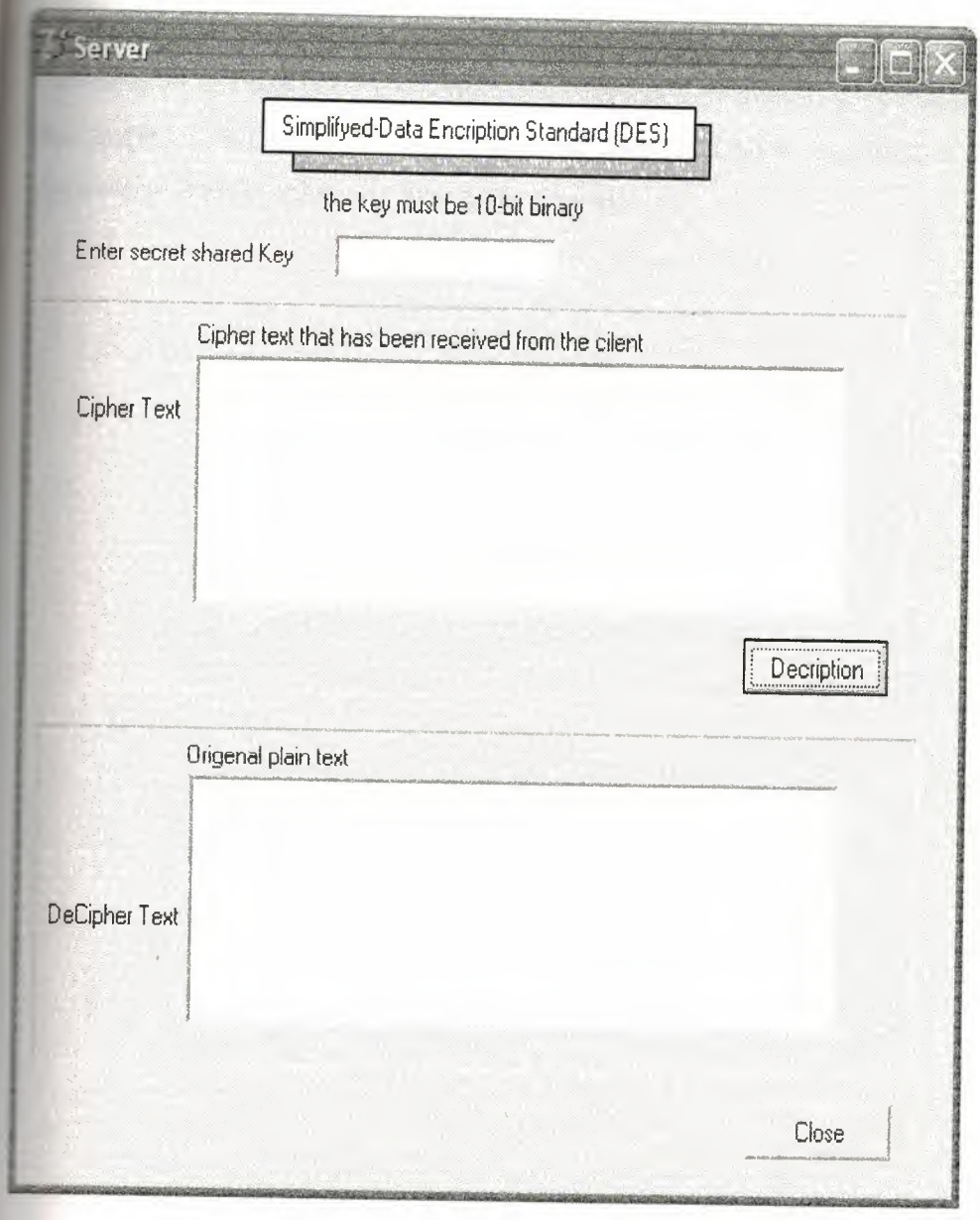


Figure 4.4 Server interface

4.3.5. Decryption

After entering the secrete key shared and by clicking on "decryption" button, the cipher text will be converted into the original text. The same operation happens as any encryption operation, where decryption uses the same algorithm as encryption, except changes in the subkeys position in the algorithm.

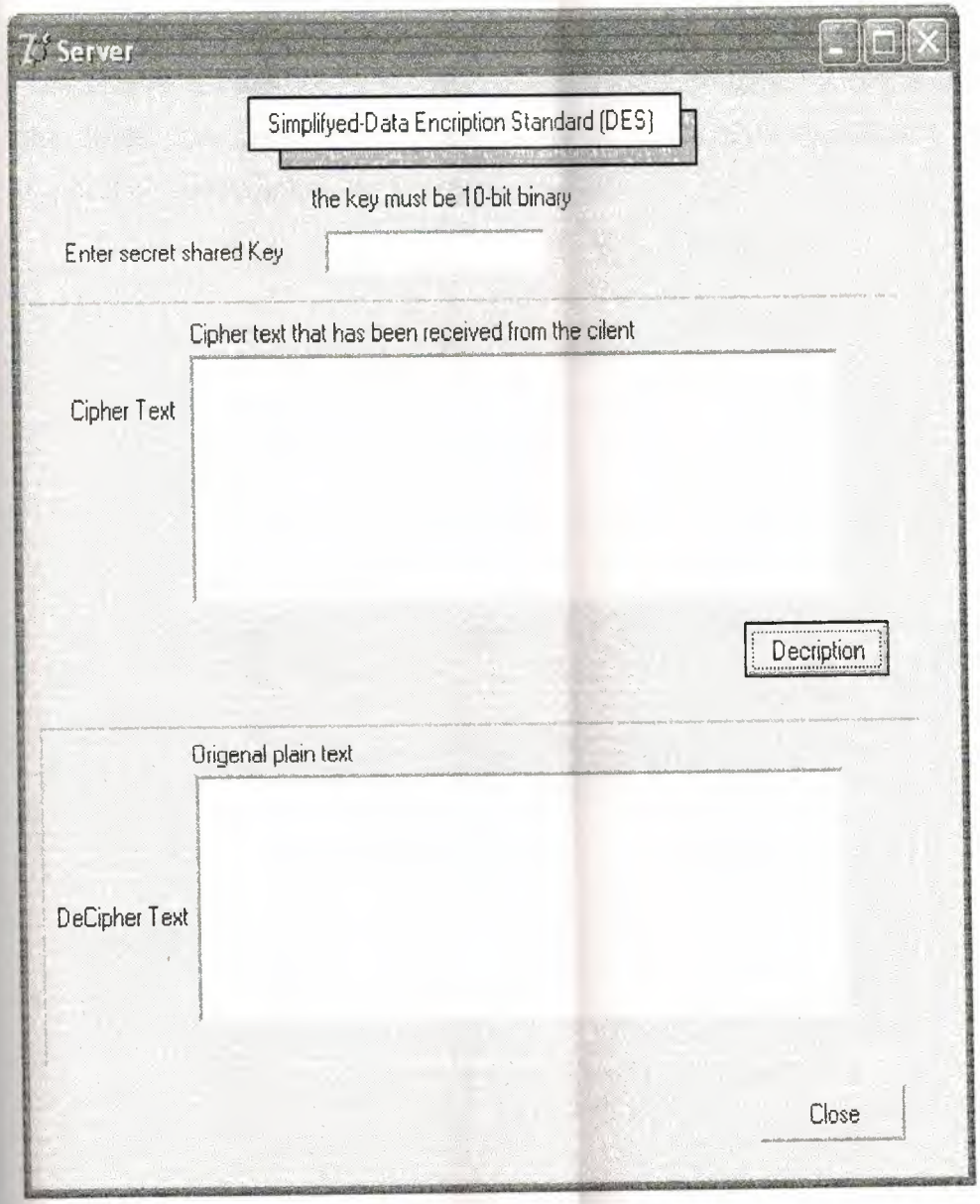


Figure 4.4 Server interface

4.3.5. Decryption

After entering the secrete key shared and by clicking on "decryption" button, the cipher text will be converted into the original text. The same operation happens as any encryption operation, where decryption uses the same algorithm as encryption, except changes in the subkeys position in the algorithm.

4.4. Summary

This chapter has described the application developed by author, which is an implementation of the single key cryptography concepts using simplified DES algorithm, it also help using the application and describes the interfaces used in.

CONCLUSION

In this thesis, it has given a detailed view on the single key cryptography and its application; it also has provided a sample program using DES protocol a practical implementation of the pervious theoretical background.

Next paragraph the important result obtained from this thesis:

- Cryptography and the applications of cryptography and where it is used, the basics of the cryptography system and its types, techniques and algorithms used by any cryptography system were discussed in more details, and finally we talked about the key management and the certificate authorities.

- Data standard encryption as a one of the modern and effective techniques of cryptography systems, describe block cipher principles and Feistel block cipher, about the origins of data standard encryption, DES algorithm and 3DES, it also discussed the modes of operation, hash function and finally the message access authentication.

- Web security considerations, and some of the famous web threats, how to protect it and finally the main standards of web security used in electronic business and commerce.

- An application developed by the author to secure data transmission over communication channels using single key cryptography technology (Data Encryption Standard algorithm) implemented using Delphi language.

But we have observed that:

- The usage of this technology is going down by time to the interest of the public key cryptography due to the growing of client/server application.
- To provide more security, the key length of DES should be increased.

REFERENCES

[1] Cryptography and Network Security, principles and practice Second Edition

[2] Bruce Schneier, Applied Cryptography, Second Edition, John Wiley & Sons, New York, 1996.

[3] Bellare, M., Canetti, R., Krawczyk, H., "Keying Hash Functions for Message Authentication", Preprint.

[4] "Secure Sockets Layers Prorocol and Application" Allan Schiffman: Tersia Systems.Inc{http://www.tersia.com}.

[5] H. Krawczyk, IETF Draft: Keyed-MD5 for Message Authentication, November 1995.

[6] ANSI X3.106, "American National Standard for Information Systems-Data Link Encryption," American National Standards Institute, 1983.

[7] NIST FIPS PUB 186, "Digital Signature Standard," National Institute of Standards and Technology, U.S Department of Commerce, 18 May 1994.

[8] National Institute of Standards and Technology, Advance Encryption Standard Development Effort Webpage. http://csrc.nist.gov/encryption/aes/aes_home.htm.

[9] B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C, Published by John Wiley & Sons, Inc. 1996.

[10] RSA Security, http://www.rsasecurity.com/

[11] SET Secure Electronic Transaction LLC, http://www.setco.org/setmark/.

[12] The Open Group, Architecture for Public-Key Infrastructure (APKI), 1999, http://www.opengroup.org/publications/catalog/g801.htm.

[13] [NS99] National Institute of Standards and Technology (1999), Data Encryption Standard (DES), U.S. Department of Commerce, U.S.A.

[14] Record-Breaking DES Key Search Completed. July, 1998 http://www.cryptography.com/resources/whitepapers/DES.html.

[15] Data Encryption Standard, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C. (January 1977).

[16] Carl H. Meyer and Stephen M. Matyas, Cryptography: A New Dimension in Computer Data Security, John Wiley & Sons, New York, 1982.

[17] Amoroso,E.: Fundamentals of Computer Security Technology, Prentice Hall, 1994.

[18] Kent, Stephen and Randall Atkinson. IP Authentication Header (RFC 2402) Internet standards track protocol, November 1998.

[19] RSA Laboratories, "PKCS #7: RSA Cryptographic Message Syntax Standard," version 1.5, November 1993.

[20] Netscape Corporation, How SSL works, http://www.netscape.com/assist/security/ssl/howitworks.html.

[21] Netscape Corporation, The SSL Protocol, http://www.netscape.com/newsref/std/SSL.html.

[22] Microsoft technical support network, http://www.microsoft.com/technet.

[23] American National Standards Institute, *ANSI X9.17: Financial Institution Key Management (Wholesale)*, 1995.

[24] E. Biham and A. Shamir, Differential cryptanalysis of the full 16-round DES, *Advances in Cryptology - Crypto '92*, Springer-Verlag (1993), 487-496.

GLOSSARY

AES The Advanced Encryption Standard that will replace DES (The Data Encryption Standard) around the turn of the century.

Algorithm A series of steps used to complete a task.

Alice The name traditionally used for the first user of cryptography in a system; Bob's friend.

ANSI American National Standards Institute.

Attack Either a successful or unsuccessful attempt at breaking part or all of a cryptosystem. See algebraic attack, birthday attack, brute force attack, chosen ciphertext attack, chosen plaintext attack, differential cryptanalysis, known plaintext attack, linear cryptanalysis, middleperson attack.

Authentication The action of verifying information such as identity, ownership or authorization.

Bit A binary digit, either 1 or 0.

Block Cipher A symmetric cipher which encrypts a message by breaking it down into blocks and encrypting each block.

Bob The name traditionally used for the second user of cryptography in a system; Alice's friend.

Certificate In cryptography, an electronic document binding some pieces of information together, such as a user's identity and public-key. Certifying Authorities (CA's) provide certificates.

Certifying Authority (CA) A person or organization that creates certificates.

Checksum Used in error detection, a checksum is a computation done on the message and transmitted with the message; similar to using parity bits.

Cipher An encryption-decryption algorithm.

Ciphertext Encrypted data.

Clipper Clipper is an encryption chip developed and sponsored by the U.S. government as part of the Capstone project.

compromise The unintended disclosure or discovery of a cryptographic key or secret.

CRL Certificate Revocation List.

Cryptanalysis The art and science of breaking encryption or any form of cryptography.

Cryptography The art and science of using mathematics to secure information and create a high degree of trust in the electronic realm. See also public key, secret key, symmetric-key, and threshold cryptography.

Cryptosystem An encryption-decryption algorithm (cipher), together with all possible plaintexts, ciphertexts and keys.

Decryption The inverse (reverse) of encryption.

DES Data Encryption Standard, a block cipher developed by IBM and the U.S. government in the 1970's as an official standard. See also block cipher.

Diffie-Hellman Key Exchange A key exchange protocol allowing the participants to agree on a key over an insecure channel.

Digital Signature The encryption of a message digest with a private key.

Distributed Key A key that is split up into many parts and shared (distributed) among different participants.

DSA Digital Signature Algorithm. DSA is a public-key method based on the discrete logarithm problem.

DSS Digital Signature Standard. DSA is the Digital Signature Standard.

ECC Elliptic Curve Cryptosystem; A public-key cryptosystem based on the properties of elliptic curves.

Electronic Commerce (*e*-commerce) Business transactions conducted over the Internet.

Electronic Mail (*e*-mail) Messages sent electronically from one person to another via the Internet.

Encryption The transformation of plaintext into an apparently less readable form (called ciphertext) through a mathematical process. The ciphertext may be read by anyone who has the key that decrypts (undoes the encryption) the ciphertext.

Feistel Cipher A special class of iterated block ciphers where the ciphertext is calculated from the plaintext by repeated application of the same transformation called a round function.

FIPS Federal Information Processing Standards.

function A mathematical relationship between two values called the input and the output, such that for each input there is precisely one output. For example, f defined on the set of real numbers as $f(x) = x^2$ is a function with input any real number x and with output the square of x.

Handshake A protocol two computers use to initiate a communication session.

Identification A process through which one ascertains the identity of another person or entity.

Internet The connection of computer networks from all over the world forming a worldwide network.

Kerberos An authentication service developed by the Project Athena team at MIT.

Key A string of bits used widely in cryptography, allowing people to encrypt and decrypt data; a key can be used to perform other mathematical operations as well. Given a cipher, a key determines the mapping of the plaintext to the ciphertext. See also distributed key, private key, public key, secret key, session key, shared key, sub key, symmetric key, weak key.

Key Exchange A process used by two more parties to exchange keys in cryptosystems.

Key Management The various processes that deal with the creation, distribution, authentication, and storage of keys.

Key Pair The full key information in a public-key cryptosystem, consisting of the public key and private key.

Key Recovery A special feature of a key management scheme that allows messages to be decrypted even if the original key is lost.

Key Schedule An algorithm that generates the subkeys in a block cipher.

Life Cycle The length of time a key can be kept in use and still provide an appropriate level of security.

Meet-In-The-Middle Attack A known plaintext attack against double encryption with two separate keys where the attacker encrypts a plaintext with a key and ``decrypts'' the original ciphertext with another key and hopes to get the same value.

Message Digest The result of applying a hash function to a message.

MIME Multipurpose Internet Mail Extensions.

NIST National Institute of Standards and Technology, a United States agency that produces security and cryptography related standards (as well as others); these standards are published as FIPS documents.

Non-Repudiation A property of a cryptosystem. Non-repudiation cryptosystems are those in which the users cannot deny actions they performed.

NSA National Security Agency. A security-conscious U. S. government agency whose mission is to decipher and monitor foreign communications.

PKI Public-key Infrastructure. PKIs are designed to solve the key management problem. See also key management.

Plaintext The data to be encrypted.

Private Key In public-key cryptography, this key is the secret key. It is primarily used for decryption but is also used for encryption with digital signatures.

Protocol A series of steps that two or more parties agree upon to complete a task.

Public Key In public-key cryptography this key is made public to all, it is primarily used for encryption but can be used for verifying signatures.

Public-Key Cryptography cryptography based on methods involving a public key and a private key.

RSA Algorithm A public-key cryptosystem based on the factoring problem. RSA stands for Rivest, Shamir and Adleman, the developers of the RSA public-key cryptosystem and the founders of RSA Data Security (now RSA Security).

S/MIME Secure Multipurpose Internet Mail Extensions.

SSL Secure Socket Layer. A protocol used for secure Internet communications.

Secret Key In secret-key cryptography, this is the key used both for encryption and decryption.

Secret Sharing Splitting a secret (e.g. a private key) into many pieces such that any specified subset of k pieces may be combined to form the secret, but k-1 pieces are not enough.

Session Key A key for symmetric-key cryptosystems which is used for the duration of one message or communication session

SET Secure Electronic Transaction. MasterCard and Visa developed (with some help from industry) this standard jointly to insure secure electronic transactions.

Shared Key The secret key two (or more) users share in a symmetric-key cryptosystem.

Stream Cipher A secret-key encryption algorithm that operates on a bit at a time.

Subkey A value generated during the key scheduling of the key used during a round in a block cipher.

Symmetric Cipher An encryption algorithm that uses the same key is used for encryption as decryption.

XOR A binary bitwise operator yielding the result one if the two values are different and zero otherwise. XOR is an abbreviation for *exclusive-OR*.

APPENDIX A

CLIENT PROGRAM CODE

```
unit cilentx;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,math,
  Corba, cypher_i, cypher_c, StdCtrls, ComCtrls, Buttons, ExtCtrls;
type
  Tcilent = class(TForm)
    Label6: TLabel;
    Label10: TLabel;
    edit1: TEdit;
    Label14: TLabel;
    Label16: TLabel;
    key1: TEdit;
    key2: TEdit;
    BitBtn5: TBitBtn;
    Label19: TLabel;
    Label20: TLabel;
    KeyGen: TEdit;
    BitBtn6: TBitBtn;
    Label21: TLabel;
    Memo2: TMemo;
    BitBtn7: TBitBtn;
    Label22: TLabel;
    Label23: TLabel;
    Memo3: TMemo;
    Label24: TLabel;
    BitBtn8: TBitBtn;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Bevel3: TBevel;
```

```
    Shape1: TShape;
    Shape2: TShape;
    Label1: TLabel;
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
  private
  { private declarations }
  protected
    // declare your Corba interface variables like this
    ciph_msg:cipheredmsg;
    procedure InitCorba;
  { protected declarations }
  public
  { public declarations }
  end;
  type array8=array[1..8]of integer;
var
  cilent: Tcilent;
  str,ch:string;
  temp:integer;
  P10,p10two          :array[1..10]of integer;
  k1,k2,m1,jP,ip,result  :array8;
  ConvTemp: array[1..4]of integer;
procedure Tcilent.InitCorba;
begin
  CorbaInitialize;
  // Bind to the Corba server like this
  ciph_msg:= TcipheredmsgHelper.bind;
end;
```

```
  //**** convert integer into byte*****
function inttobyte(x :integer):array8;
var
 count:integer;
 ar:array8;
begin
  count:=8;
  while x<>0 do
  begin
    if (x mod 2)<>0 then
    begin
      ar[count]:=1;
      x:=round((x-1)/2);
    end
    else
    begin
      ar[count]:=0;
      x:=round(x/2);
    end;
    dec(count);
  end;
  while count>0 do
  begin
    ar[count]:=0;
    dec(count);
  end;
  result:=ar;
end;
//*****************************
function bytetoint(ar:array8):integer;
var
  i:integer;
begin
```

```
 result:=0;
 for i:=1 to 8 do
 begin
   result:=round(result+ power(2,i-1)*ar[9-i]);
 end;
end;
function F1(ip:array8;key1:array8):array8;
var
 i,r,c: integer;
 s0,s1:array[0..3,0..3]of integer;
 ep,xr1,xr2:array8;
 p4,p4final :array[1..4] of integer;
begin
 s0[0,0]:=1; s0[0,1]:=0; s0[0,2]:=3; s0[0,3]:=2;
 s0[1,0]:=3; s0[1,1]:=2; s0[1,2]:=1; s0[1,3]:=0;
 s0[2,0]:=0; s0[2,1]:=2; s0[2,2]:=1; s0[2,3]:=3;
 s0[3,0]:=3; s0[3,1]:=1; s0[3,2]:=3; s0[3,3]:=2;

 s1[0,0]:=0; s1[0,1]:=1; s1[0,2]:=2; s1[0,3]:=3;
 s1[1,0]:=2; s1[1,1]:=0; s1[1,2]:=1; s1[1,3]:=3;
 s1[2,0]:=3; s1[2,1]:=0; s1[2,2]:=1; s1[2,3]:=0;
 s1[3,0]:=2; s1[3,1]:=1; s1[3,2]:=0; s1[3,3]:=3;

 ep[1]:=ip[8];
 ep[2]:=ip[5];
 ep[3]:=ip[6];
 ep[4]:=ip[7];
 ep[5]:=ip[6];
 ep[6]:=ip[7];
 ep[7]:=ip[8];
 ep[8]:=ip[5];

 //--make xor--//
```

```
for i:=1 to 8 do
 if ep[i]=key1[i] then
   xr1[i]:=0
 else
   xr1[i]:=1;
  //-get p4 part1---//
r:=xr1[1]*2+xr1[4];
c:=xr1[2]*2+xr1[3];

if s0[r,c] =0 then
begin p4[1]:=0;p4[2]:=0; end

else if s0[r,c] =1 then
begin p4[1]:=0;p4[2]:=1; end

else if s0[r,c] =2 then
begin p4[1]:=1;p4[2]:=0; end

else if s0[r,c] =3 then
begin p4[1]:=1;p4[2]:=1; end;
   //----//
  //---get p4 part2 --//
r:=xr1[5]*2+xr1[8];
c:=xr1[6]*2+xr1[7];
if s1[r,c] =0 then
begin p4[3]:=0;p4[4]:=0; end
else if s1[r,c] =1 then
begin p4[3]:=0;p4[4]:=1; end

else if s1[r,c] =2 then
begin p4[3]:=1;p4[4]:=0; end

else if s1[r,c] =3 then
```

```
begin p4[3]:=1;p4[4]:=1; end;
  //--applay p4--//
p4final[1]:=p4[2];
p4final[2]:=p4[4];
p4final[3]:=p4[3];
p4final[4]:=p4[1];
//--make xor--//
for i:=1 to 4 do
 if p4final[i]=ip[i] then
   xr2[i]:=0
 else
   xr2[i]:=1;
for i:=5 to 8 do
  xr2[i]:=ip[i];
f1:=xr2;
end;
//------------end function----------------
procedure Tcilent.BitBtn1Click(Sender: TObject);
var
 i:integer;
begin
 str:=KeyGen.text;
 if length(str)<>10 then
 begin
   showmessage('You have to enter 10 bit binary');
   exit;
 end;
 for i:=1 to 10 do
 begin
  if (str[i]='1') or (str[i]='0') then
    p10[i]:=strtoint(str[i])
  else
  begin
```

```
    showmessage('The Key must be 10 bit binary');
    exit;
  end;
end;
  p10two[1] :=p10[3];
  p10two[2] :=p10[5];
  p10two[3] :=p10[2];
  p10two[4] :=p10[7];
  p10two[5] :=p10[4];
  p10two[6] :=p10[10];
  p10two[7] :=p10[1];
  p10two[8] :=p10[9];
  p10two[9] :=p10[8];
  p10two[10]:=p10[6];

  temp:=p10two[1];
  for i:=1 to 4 do
    p10two[i]:=p10two[i+1];
  p10two[5]:=temp;
  temp:=p10two[6];
  for i:=6 to 9 do
    p10two[i]:=p10two[i+1];
  p10two[10]:=temp;
//---------- generate K1 -------
  k1[1]:=p10two[6];
  k1[2]:=p10two[3];
  k1[3]:=p10two[7];
  k1[4]:=p10two[4];
  k1[5]:=p10two[8];
  k1[6]:=p10two[5];
  k1[7]:=p10two[10];
  k1[8]:=p10two[9];
  str:='';
```

```
    for i:=1 to 8 do
    begin
     ch:=inttostr(k1[i]);
     str:=str+ch;
    end;
    key1.text:=str;
//------------------------------
//---------- generate K2 -------
    p10[1] :=p10two[3];
    p10[2] :=p10two[4];
    p10[3] :=p10two[5];
    p10[4] :=p10two[1];
    p10[5] :=p10two[2];
    p10[6] :=p10two[8];
    p10[7] :=p10two[9];
    p10[8] :=p10two[10];
    p10[9] :=p10two[6];
    p10[10]:=p10two[7];

    k2[1]:=p10[6];
    k2[2]:=p10[3];
    k2[3]:=p10[7];
    k2[4]:=p10[4];
    k2[5]:=p10[8];
    k2[6]:=p10[5];
    k2[7]:=p10[10];
    k2[8]:=p10[9];
    str:='' ;
    for i:=1 to 8 do
    begin
     ch:=inttostr(k2[i]);
     str:=str+ch;
    end;
```

```
    key2.text:=str;
    edit1.Text:= KeyGen.Text;
Memo2.Enabled:=true;
Memo2.Color:=clwindow;
Memo2.SetFocus;
end;
procedure Tcilent.BitBtn2Click(Sender: TObject);
begin
 //  PageControl1.SelectNextPage(true);
end;
procedure Tcilent.BitBtn3Click(Sender: TObject);
 var
  chara:char;
  i,j,va:integer;
  text:string;
begin
  str:='';
  text:=Memo2.Lines.Text;//msg.text;
  for j:=1 to length(text) do
  begin
    chara:=text[j];
    m1:=inttobyte(ord(chara));
    //-- get IP--
    ip[1]:=m1[2];
    ip[2]:=m1[6];
    ip[3]:=m1[3];
    ip[4]:=m1[1];
    ip[5]:=m1[4];
    ip[6]:=m1[8];
    ip[7]:=m1[5];
    ip[8]:=m1[7];
 //*********************************************************
//*********************************************************
```

```
  result:=f1(ip,k1);
  for i:=1 to 4 do
    ConvTemp[i]:=result[i];
  for i:=1 to 4 do
   result[i]:=result[i+4];
  for i:=5 to 8 do
    result[i]:=ConvTemp[i-4];
  result:=f1(result,k2);
  jp[1]:=result[4];
  jp[2]:=result[1];
  jp[3]:=result[3];
  jp[4]:=result[5];
  jp[5]:=result[7];
  jp[6]:=result[2];
  jp[7]:=result[8];
  jp[8]:=result[6];

   va:= bytetoint(jp);
   str:=str + chr(va);
 end;
// Ciphertext.text:=str;
 Memo3.Lines.Text:=str;
{  va:= bytetoint(jp);
 str:='';
 str:=chr(va);
 Ciphertext.text:=str;
}
{   str:='' ;
   for i:=1 to 8 do
   begin
     ch:=inttostr(jp[i]);
     str:=str+ch;
   end;
```

```
    Ciphertext.text:=str;}
//   PageControl1.SelectNextPage(true);
end;
procedure Tcilent.FormCreate(Sender: TObject);
begin
 InitCorba;
end;

procedure Tcilent.BitBtn4Click(Sender: TObject);
begin
 ciph_msg.receivemsg(Memo3.Text, Key1.Text, key2.Text);
end;
end.
```

SERVER PROGRAM CODE

```
unit serverx;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,Math,
  Corba, cypher_i, cypher_c, cypher_s, cypher_impl, StdCtrls, Buttons,
  ComCtrls, ExtCtrls;

type
  TServer = class(TForm)
    Label1: TLabel;
    Memo3: TMemo;
    Label2: TLabel;
    BitBtn2: TBitBtn;
    Memo1: TMemo;
    Label3: TLabel;
    Label4: TLabel;
    Bevel1: TBevel;
    Bevel2: TBevel;
```

```
    Shape1: TShape;
    Shape2: TShape;
    Label5: TLabel;
    KeyGen: TEdit;
    BitBtn6: TBitBtn;
    Label6: TLabel;
    Label7: TLabel;
    Button1: TButton;
    procedure BitBtn4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Memo1KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure Memo1KeyPress(Sender: TObject; var Key: Char);
    procedure KeyGenKeyPress(Sender: TObject; var Key: Char);
    procedure Button1Click(Sender: TObject);
  private
  { private declarations }
  protected
  { protected declarations }
   // Add Corba interface variables here like this
   ciph_msg :cipheredmsg; // skeleton object
   procedure InitCorba;
  public
  { public declarations }
  strKey1, StrKey2:string;
  end;

type array8=array[1..8]of integer;

var
  Server: TServer;
  str,ch:string;
```

```
 temp:integer;
 P10,p10two          :array[1..10]of integer;
 k1,k2,m1,jP,ip,result  :array8;
 ConvTemp: array[1..4]of integer;

implementation
procedure TServer.InitCorba;
begin
 CorbaInitialize;

 // Add CORBA server code here like this
  ciph_msg:= TcipheredmsgSkeleton.Create('ready', tcipheredmsg.Create);
  BOA.ObjIsReady(ciph_msg as _Object);
end;

function bytetoint(ar:array8):integer;
var
 i:integer;
begin
 result:=0;
 for i:=1 to 8 do
 begin
   result:=round(result+ power(2,i-1)*ar[9-i]);
 end;
end;

function F1(ip:array8;key1:array8):array8;
var
 i,r,c: integer;
 s0,s1:array[0..3,0..3]of integer;
 ep,xr1,xr2:array8;
 p4,p4final :array[1..4] of integer;
begin
```

```
s0[0,0]:=1; s0[0,1]:=0; s0[0,2]:=3; s0[0,3]:=2;
s0[1,0]:=3; s0[1,1]:=2; s0[1,2]:=1; s0[1,3]:=0;
s0[2,0]:=0; s0[2,1]:=2; s0[2,2]:=1; s0[2,3]:=3;
s0[3,0]:=3; s0[3,1]:=1; s0[3,2]:=3; s0[3,3]:=2;


s1[0,0]:=0; s1[0,1]:=1; s1[0,2]:=2; s1[0,3]:=3;
s1[1,0]:=2; s1[1,1]:=0; s1[1,2]:=1; s1[1,3]:=3;
s1[2,0]:=3; s1[2,1]:=0; s1[2,2]:=1; s1[2,3]:=0;
s1[3,0]:=2; s1[3,1]:=1; s1[3,2]:=0; s1[3,3]:=3;

ep[1]:=ip[8];
ep[2]:=ip[5];
ep[3]:=ip[6];
ep[4]:=ip[7];
ep[5]:=ip[6];
ep[6]:=ip[7];
ep[7]:=ip[8];
ep[8]:=ip[5];

//--make xor--//
for i:=1 to 8 do
 if ep[i]=key1[i] then
    xr1[i]:=0
 else
    xr1[i]:=1;

   //-get p4 part1---//
r:=xr1[1]*2+xr1[4];
c:=xr1[2]*2+xr1[3];

if s0[r,c] =0 then
begin p4[1]:=0;p4[2]:=0; end
```

```
else if s0[r,c] =1 then
begin p4[1]:=0;p4[2]:=1; end

else if s0[r,c] =2 then
begin p4[1]:=1;p4[2]:=0; end

else if s0[r,c] =3 then
begin p4[1]:=1;p4[2]:=1; end;
  //----//
  //---get p4 part2 --//
r:=xr1[5]*2+xr1[8];
c:=xr1[6]*2+xr1[7];

if s1[r,c] =0 then
begin p4[3]:=0;p4[4]:=0; end

else if s1[r,c] =1 then
begin p4[3]:=0;p4[4]:=1; end

else if s1[r,c] =2 then
begin p4[3]:=1;p4[4]:=0; end

else if s1[r,c] =3 then
begin p4[3]:=1;p4[4]:=1; end;
  //--applay p4--//
p4final[1]:=p4[2];
p4final[2]:=p4[4];
p4final[3]:=p4[3];
p4final[4]:=p4[1];

//--make xor--//
for i:=1 to 4 do
```

```
   if p4final[i]=ip[i] then
     xr2[i]:=0
   else
     xr2[i]:=1;
  for i:=5 to 8 do
    xr2[i]:=ip[i];
  f1:=xr2;
end;
//------------end function----------------

   //**** convert integer into byte*****
function inttobyte(x :integer):array8;
var
 count:integer;
 ar:array8;
begin
  count:=8;
  while x<>0 do
  begin
    if (x mod 2)<>0 then
    begin
      ar[count]:=1;
      x:=round((x-1)/2);
    end
    else
    begin
      ar[count]:=0;
      x:=round(x/2);
    end;
    dec(count);
  end;

  while count>0 do
```

```
 begin
   ar[count]:=0;
   dec(count);
 end;
 result:=ar;
end;
procedure TServer.BitBtn4Click(Sender: TObject);
var
 va:integer;
 i,j:integer;
 text:string;
 chara:char;
 str:string;
begin
  BitBtn1Click(Sender);

  { for i:=1 to 8 do
   begin
     k1[i]:=strtoint(StrKey1[i]);
     k2[i]:=strtoint(StrKey2[i]);
   end;
   }
   text:=Memo3.Lines.Text;

   for j:=1 to length(text) do
   begin
     chara:=text[j];
     Jp:=inttobyte(ord(chara));
     ip[1]:=jp[2];
     ip[2]:=jp[6];
     ip[3]:=jp[3];
     ip[4]:=jp[1];
     ip[5]:=jp[4];
```

```
    ip[6]:=jp[8];
    ip[7]:=jp[5];
    ip[8]:=jp[7];
    result:=f1(ip,k2);

    for i:=1 to 4 do
     ConvTemp[i]:=result[i];

    for i:=1 to 4 do
     result[i]:=result[i+4];

    for i:=5 to 8 do
     result[i]:=ConvTemp[i-4];

    result:=f1(result,k1);
    ip[1]:=result[4];
    ip[2]:=result[1];
    ip[3]:=result[3];
    ip[4]:=result[5];
    ip[5]:=result[7];
    ip[6]:=result[2];
    ip[7]:=result[8];
    ip[8]:=result[6];

    va:= bytetoint(ip);
    str:=str+chr(va);
  end;

  Memo1.Lines.Text:=str;

 // PageControl1.SelectNextPage(true);
end;
procedure TServer.FormCreate(Sender: TObject);
```

```
begin
  InitCorba;
end;



procedure TServer.BitBtn1Click(Sender: TObject);
var
 i:integer;
begin
  str:=KeyGen.text;
  if length(str)<>10 then
  begin
    showmessage('The Key must be 10 bit binary');
    exit;
  end;

  for i:=1 to 10 do
  begin
   if (str[i]='1') or (str[i]='0') then
     p10[i]:=strtoint(str[i])
   else
   begin
     showmessage('The Key must be 10 bit binary');
     exit;
   end;
  end;

    p10two[1] :=p10[3];
    p10two[2] :=p10[5];
    p10two[3] :=p10[2];
    p10two[4] :=p10[7];
    p10two[5] :=p10[4];
```

```
p10two[6] :=p10[10];
p10two[7] :=p10[1];
p10two[8] :=p10[9];
p10two[9] :=p10[8];
p10two[10]:=p10[6];

temp:=p10two[1];
for i:=1 to 4 do
 p10two[i]:=p10two[i+1];
p10two[5]:=temp;

temp:=p10two[6];
for i:=6 to 9 do
 p10two[i]:=p10two[i+1];
p10two[10]:=temp;
//---------- generate K1 -------
k1[1]:=p10two[6];
k1[2]:=p10two[3];
k1[3]:=p10two[7];
k1[4]:=p10two[4];
k1[5]:=p10two[8];
k1[6]:=p10two[5];
k1[7]:=p10two[10];
k1[8]:=p10two[9];

str:=" ;
for i:=1 to 8 do
begin
 ch:=inttostr(k1[i]);
 str:=str+ch;
end;
//key1.text:=str;
//-------------------------------
```

```
//---------- generate K2 -------
   p10[1] :=p10two[3];
   p10[2] :=p10two[4];
   p10[3] :=p10two[5];
   p10[4] :=p10two[1];
   p10[5] :=p10two[2];
   p10[6] :=p10two[8];
   p10[7] :=p10two[9];
   p10[8] :=p10two[10];
   p10[9] :=p10two[6];
   p10[10]:=p10two[7];

   k2[1]:=p10[6];
   k2[2]:=p10[3];
   k2[3]:=p10[7];
   k2[4]:=p10[4];
   k2[5]:=p10[8];
   k2[6]:=p10[5];
   k2[7]:=p10[10];
   k2[8]:=p10[9];

   str:='' ;
   for i:=1 to 8 do
   begin
     ch:=inttostr(k2[i]);
     str:=str+ch;
   end;
  // key2.text:=str;
   //edit1.Text:= KeyGen.Text;


  // strKey1:=k1;
```

```
  // strKey2:=k2

end;

procedure TServer.Memo1KeyDown(Sender: TObject; var Key: Word;
 Shift: TShiftState);
begin
 { if key in [13,38,40,37,39] then
  else
   key:=0;
 showmessage(inttostr(key));}
end;

procedure TServer.Memo1KeyPress(Sender: TObject; var Key: Char);
begin
  if key in [#13,#38,#40,#37,#39] then
  else
   key:=#0;
end;

procedure TServer.KeyGenKeyPress(Sender: TObject; var Key: Char);
begin

  if (key <> '1') and (key <> '0') then
    key:=#0;
end;
procedure TServer.Button1Click(Sender: TObject);
begin
close;
end;
end.
```