References:

[1]Charles P. Pfleeger, Security In Computing, Prentice Hall,2006.

[2]www.comms.scitech.susx.ac.uk/fft/crypto/cryptography.pdf.

[3]Mark Stamp, Information: Security Principles and Practice, John Whiley & Sons, Hoboken, New Jersy,2006.

[4]B.Scneier,Applied Cryptography, John Whiley & Sons, New York ,1996.

ABSTRACT

As a general definition, encryption means a technique to convert the plain-text into a secret form, usually this process is controlled by a key. this key is similar to the mechanical key used to control the entry to a certain room. the security of the real-world encryption algorithms lies solely on the key.

Encryption algorithms can be classified into two kinds, classical and modern ,the classical algorithms are simple to implement without using computers, while the modern algorithms are designed to be used on computers.

Monoalphabetic, Polyalphabetic, Transposition…etc. are classical algorithms, DES is a modern algorithm.

With today’s computing abilities ,DES has shown the signs of aging. it has been the target for many attacks for almost quarter a century .this makes the need for stronger standard more serious.

This project is aimed to design and implement an encryption technique that is derived from the DES and provides more security. this is estimated to be 2^4 times the already available security factor. the proposed development was directed toward the sub-key generation algorithm.

All the algorithms available in this project were programmed using Microsoft Visual BASIC 6.0 and tested successfully.

List of Contents

List of Tables

List of Figures

Chapter One:Introduction

1.1 Introduction [1]

Suppose we have a message to communicate, but our message might fall into the wrong hands. by using encryption we distinguish the message so that even if the transmission is diverted, the message will not be revealed. encryption is a means of maintaining secure data in an insecure environment.

Suppose S wants to send a message to T, who will deliver it to R, T then becomes the transmission medium. if an outsider ,O, wants the message and tries to obtain access to the message ,we call O an interceptor or intruder. any time after S transmits it via T, the message is exposed, so O might try to access the message in any of the following ways:

- **interrupt it:**by preventing its reaching R, thereby affecting the availability of the message.
- **intercept it:**by being able to see or listen to the message, thereby affecting the secrecy of the message.
- **modify it:**by seizing the message and changing it in some way.
- **fabricate an authentic-looking message**, arranging for it to be delivered as if it came from S ,thereby affecting the integrity of the message.

The original form of the message is known as the plain text, and the encrypted form is called ciphertext. this situation is shown in Fig. 1-1 below.for convenience in explanation we may denote a plaintext message *P* as a sequence of individual characters *P=[p1,p2,…,pn]*,similarly ciphertext can be written as *C=[c1,c2,….,cn]*.formally,the transformations between plain text and cipher text are denoted *C=E(P)* and *P=D(C)* ,where *C* represents the ciphertext,*E* is the encryption algorithm,*P* is the plain text and *D* is the decryption algorithm.obviously we want a crypto system for which *P=D(E(P)).*

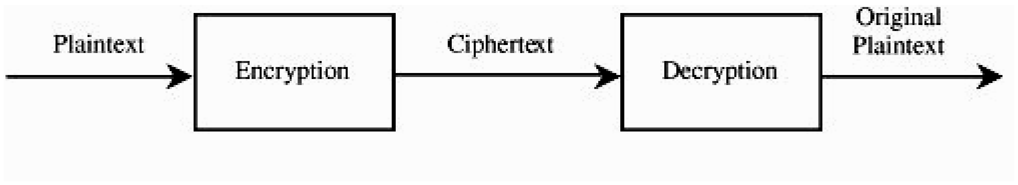


Figure 1-1 The encryption and decryption process

Some encryptions use a key *K*,so that the ciphertext message depends on both the original plaintext message and the key value.denoted *C=E(K,P)*.essentially,*E* is a set of encryption algorithms,and the key *K* selects one specific algorithm.sometimes the encryption and decryption keys are the same,so that *P=D(K,E(K,P))* other times encryption and decryption keys come in pairs.then a decryption key,K_D,inverts the encryption of key K_E,so that $P=D(K_D,E(K_E,P))$.these two cases are shown in fig.1-2.

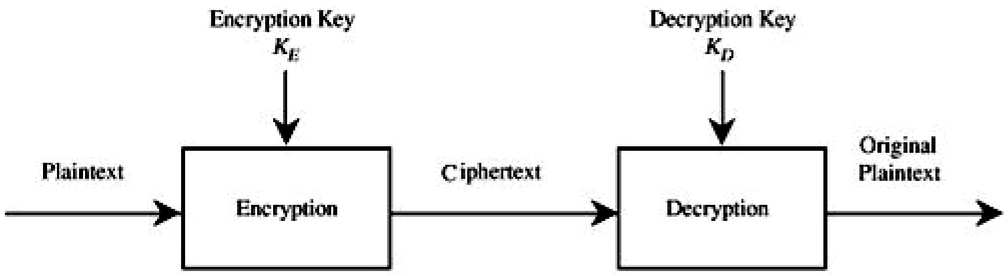


Figure 1-2 Asymmetric Cryptosystem

A key allows different encryptions of one plaintext message just by changing the key.use of a key provides additional security.if the encryption algorithm should fall into the interceptors hands,future messages can still be kept secret because the interceptor will not know the key value.a cipher that does not require the use of a key is called a keyless cipher.

1.2 Cryptography - (Definition) [2]

We may wonder why cryptography is so important and the people need to study more about it. We will study in this project about the cryptography and the awareness that every information specialist should know about cryptography and its importance.

Cryptography - is the study of mathematical techniques related to the aspects of information security such as confidentiality, data integrity, authentication and data origination.

We are going to see the history and science of cryptography. The science behind every form of security, authentication mechanisms, information or data safety, in our words, *information security* and what not. The reason is, whichever operating system you use, whatever programs or authentication systems you deploy, the actual strength of the system to withstand attacks against possible ways depends highly on the underlying cryptosystem.

The importance for information security has an old history. The story begins when *Julius Caesar* send messages to his trusted acquaintances, he didn't trust the messengers or likely to know that message can be intercepted while on the way. So he replaced every 'A' with 'D' and 'B' with 'E' and so on through the alphabets. Only someone who knew '**SHIFT BY 3**' could ***decipher*** his message.

Some of the common terms that are used in cryptosystems are explained here. The original message is called as the ***plaintext***. The disguised message is called as the ***ciphertext***.The method of producing *ciphertext* from *plaintext* using the key is called as ***encryption*** or ***enciphering*** or ***encoding*** and the reverse procedure of producing the *plaintext* from *ciphertext* using the key is called as ***decryption*** or ***deciphering*** or ***decoding***. The people who are supposed to receive the disguised message are called the ***recipients***, and other people are ***enemies, oppenents, interloppers, eavesdroppers, tappers*** or ***third parties***. A ***cryptosystem*** means a collection of algorithms. Algorithms are labeled and the labels are called the keys. For example, if caesar used, '*SHIFT BY n*', algorithm means, *n* is the ***key***. Key plays and important role the cryptosystem because the whole strength of the algorithm used depends on how the key is choosed. For a very strong algorithm, and an easily guessable key means, the encoding of the message goes as a waste.

The science of breaking ***cryptosystems*** is called the ***Cryptanalysis***. Though cryptanalysis may not be the easier to learn, because of it complexity involved in breaking the ciphered messages, but to decipher it ***without*** the knowledge of the cryptoalgorithm and the key used. The study of both cryptography and cryptanalysis collectively is called as the cryptology. It may be possible to find the key used or to find the plaintext from the ciphertext, by a cryptoanalyst by using various attack techniques.

Cryptanalysis plays an important role in the cryptography because, it attacks the encoded message to produce the relevant plaintext. For example, if a function $f(x)$ gives, y and you know the function and the result y, all you need to find is the key x, which can be found using many methods. One such method is called as ***brute-forcing.*** It is nothing but substituting all possible values of x in the function f and match the output with y, the one that matches, gives the x which is also the key.

The all possible values of *x* means the ***keyspace*** where the key is searched. For every known algorithm today, there exists a *keyspace* and the technique of searching called as the ***keysearch.*** You may think then, every known algorithm can be broken, but the complexity of the algorithm and the keystrength, makes the task tougher. The massive amount of computing power and time required to break the message and to find the key makes the algorithm stronger.

A good cryptosystem, should be able to withstand different kind of attacks, depending upon the algorithm used. Various other methods used to break the encrypted messages are classified attacks, interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, computing power and offcourse not the least but luck. Also the computational number theory posses the threat to what so called as the public-key crypto systems, which are widely used in all formats in modern secure transactions, authentication and digital certificates. Though the keyspace and amount of computational power makes it impractical for such attacks.

1.3 Why Cryptography

The main use of cryptography is to provide the following as mentioned earlier.

(1) privacy or confidentiality

(2) data integrity

(3) authentication and

(4) non-repudiation.

1. ***Confidentiality*** is a service used to keep the content of information from all but those authorized to posses it. *Secrecy* is a term synonymous with confidentiality and privacy. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible.

2. ***Data integrity*** is a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution.

3. ***Authentication*** is a service related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other.

Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons this aspect of cryptography is usually subdivided into two major classes: *entityauthentication* and *dataorigin authenticatio*n. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).

4. ***Non-repudiation*** is a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted. A procedure involving a trusted third party is needed to resolve the dispute.
A fundamental goal of cryptography is to adequately address these four areas in both theory and practice. Cryptography is about the prevention and detection of cheating and other malicious activities and to secure what you have as sensitive information.

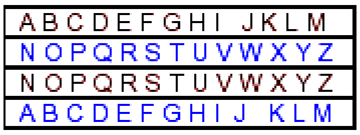
1.4 History and Classification of Encryption & Decryption

The various methods that are devised with relevant to time starting from historic events. It is very tough to spot the origin of cryptography, because it started when men, want to hide or protect the information, to provide confidentiality, it became a practice.

Julius caesar, used the old method of shifting for producing ciphertext, which is called as the ***simple monoalphabetic substitution cipher***.

Then various modified versions of the same shifting became a practice in *usenets,* where the message if shifted by 13 (**rot13**), which on encoding twice produces the plaintext.It was the one of trivial encodings used for a long time on message boards and news servers. So the news readers have builtin **rot13** functions to decode or encode plaintexts.
For your assistance We include the relevant substitution table.

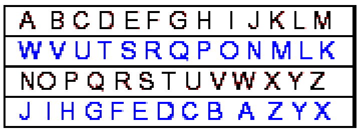
Table 1-1 rot13 substitution table



A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M

The next was the ***atbash*** system, which is a modified version of the caesar and even easier to solve. The only difference was *Caesars* was Roman in origin and *atbash* was Jewish in origin. In the atbash system, the last letter represents the first letter and first letter represents the last letter and so on. So there was practically only one solutions and quickly decipherable. The table that was used in atbash is The combination of caesars and atbash was used in some encodings schemes.

Table 1-2 Atbash substitution table



A	B	C	D	E	F	G	H	I	J	K	L	M
W	V	U	T	S	R	Q	P	O	N	M	L	K
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
J	I	H	G	F	E	D	C	B	A	Z	Y	X

Ancient Babylonian merchants used *intaglio*, a piece of flat stone carved into a collage of images and some writing to identify themselves in trading transactions. Using this mechanism, they are producing what today we know as 'digital signature.' The public knew that a particular 'signature' belonged to this trader, but only he had the intaglio to produce that signature.

The other methods are the monoalphabetic substitution where a letter of plaintext always produces the same cipher text. The usage is similar to atbash and caesars, but the order of the letters is user defined. So you can produce a ciphering table for a monoalphabetic cipher. It was cracked by the methods known as frequency search, which searches for vowels which occur more in a paragraph and concluding the keys for that alphabet. The famous phrase used was ***'pen and paper techniques',*** which required a little common sense and analytical reasoning for breaking the substitution ciphers.

Various other methods like ***Polybius Chequerboard, Vigenegravere or the famous autokey cipher, Playfair, Transpositional cipher****,* etc, came into picture. They collectively used key to manipulate the plaintext, which was broken easily in many cases. But there are some historic ciphers which remains still undeciphered.

With the invention of crypto machine called ***enigma*** by the allies and Germans, during the World War I and World War II for the transfer of commands and orders to the troops and about the actual situation. Some people intercepted the messages but failed to decrypt due to the constant improvement and change done with the machine architecture.

1.5 Modern Cryptography

The classification divides itself as the method of enciphering the plaintext to ciphertext differs into various methods that are devised. The cyptography branches itself into ***Symmetric Key cryptography often known as conventional cryptogrphy or secret-key cryptography*** *and* ***public-key cryptography (asymmetric key cryptography) or public/secret pair cryptography***. The *symmetric key* technique further classifies itself into ***Block ciphers*** *and* ***Stream ciphers.***

Symmetric key cryptography is the method where the plaintext is converted to cipher text based on the ***unique key*** and the function used. The actual strength of the symmetric key cryptography lies in choosing the nonreversible function that uses the key to produce the cipher text. Like a function $E_K(P) = C$,where *E* is the function and ***K*** is the key used and ***P*** is the plaintext and ***C*** is the ciphertext produced. Then again the use of function $D_K(C) = P$,where *D* is the function used to decrypt the messages using the above components. It is possible to produce the function $E=D$, and such functions uses the total key length as strength. Though 64 bits may offer a very good security, a 128 bit or 256 bit will definitely use for any mission critical applications. It also aids in improving security because of the increase in the key space, which makes it less prone to brute-force key search and other kind of attacks like plaintext attack,choosed plaintext attack, differential plaintext attack etc. The Stream ciphers are the algorithms that uses the function to encrypt the plaintext in a stream more or less like reading a file character by character and encoding it. The block ciphers are used to encrypt the plaintext in a predefined size of blocks. Say 32 bytes, 64 bytes or 128 bytes. It is done to achieve speed.

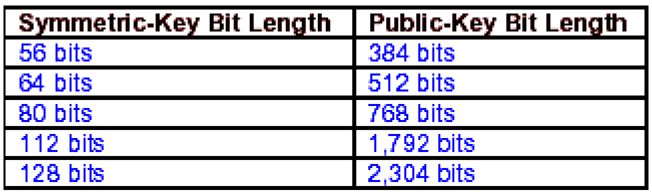
Popular block ciphers are ***DES***, ***Blowfish*** etc, and stream ciphers are ***vernam ciphers*** or also known as ***one time pad (Considered to be very secure but practically almost impossible, aka OTP)*** ciphers.

In ***block Ciphers*** mode when we encode the message, the message size was not altered. Whereas in the ***stream cipher***, the message length gets altered. Thus these two are the major classification of the symmetric key cryptography. Various algorithms like DES,Blowfish come under the block cipher category.

The various attack methods used in the cryptanalysis against symmetric key cryptography are differential cryptanalysis, linear cryptanalysis and algebric attacks.

Here is a comparison between the ***Symmetric key*** algorithms and ***Public key*** algorithms based on their key sizes.

Table 1-3 a comparison between symmetric and public key algorithms



Symmetric-Key Bit Length	Public-Key Bit Length
56 bits	384 bits
64 bits	512 bits
80 bits	768 bits
112 bits	1,792 bits
128 bits	2,304 bits

Public-key cryptography - as the name indicates, uses two kinds of functions and two different keys. The keys are terms as the ***private key*** and ***public key***. The *public key* is the one which is kept *'visible'*, (i.e.), commonly transmitted over networks, etc. and the other one which is kept *'secret'* the private key, which is never revealed to anybody. Thus the system, uses both keys, it's commonly know as *'public/secret pair algorithms'* or *'private/public key algorithms'* though the name *public key cryptography* exists.The functions used in the public key cryptography are like the following.

Consider the function ***E*** **,$E_K(P)$ =*C*,** which uses the ***K***, the public key to encrypt plaintext (***P***) to produce ciphertext (***C***) can only be decrypted using another function ***D***, **$D_S(C)=P$** Which uses ***S***, which is the private key.

In some, (but not all) public/private algorithms, you can also use ***S*** to encrypt the plaintext that can be later decrypted using ***K***, which is in use by some programs. The major advantage of the public/private pair cryptography is that it solves the issue of key distribution. But since, the functioning is concerned, the pair of functions deployed makes it less secure and there exists a list of attacks against the system. So, even the key strength of 768 bits pose a question threat to information security. Normally a key strength of 1024 and above is secure. But there are programs, which give about 4096 bits of key size which is an uncommon, but providing with a very good security. Such programs are often controlled by the U.S. government. due to export restrictions to be used outside U.S.

Public-key algorithms are slower compared to the speed of symmetric key algorithms or secret-key algorithms. Also, public-key algorithms are prone to attack than the secret-key algorithms. Another main use of public-key algorithms is to produce ***digital signatures***, which plays the lead role in identifying the origin of the message. The *digital signature* thus provides the way to authenticate which is known as message non-repudiation as explained before.Public-key cryptography may be vulnerable to impersonation, but it is the sole responsibility of the user to protect his/her private key securely.

Digital signatures can be used to identify the message source and the integrity of the message using the digital fingerprint like conventional fingerprint matching. Slightest modification in the message will cause the fingerprinting mechanisms to fail and the whole message is discarded or reported back to the sender about the incident. It also provides the authentication mechanisms that requires users to sign on to a public terminal or access a network resource, by using the various implementations of the public-key cryptography. One such implementation is ***kerberos***, which was developed at **MIT**, can be used for a network wide authentication procedure for a list from pre-qualified users using their digital signatures.

The real power can be brought out by using the combination of secret-key and public-key cryptosystems. Like using a secret-key algorithm to encrypt the data/file/message and then signing it with the public-key algorithm can be the highest possible security that can be given to a data/file/message. Also compressing the data/file/message with/without encryption (which uses some patented symmetric key algorithms!) will be more secure because the integrity is maintained to the lowest possible level.

The other mechanism, which is used along with above methods is called as ***hashing or message digesting.*** It means to produce a very small value, say ***H1*** from a function ***M*** for the plaintext ***P1***. It should be hard to find any other plaintext ***P2 ,*** which satisfies ***M(P2)=H2***, which gives ***H1=H2*** proves that the plaintext **P1=P2**.

For Example, the function *M*,$M(P_X)=H_X$, Where ***X*** stands for the different plaintexts, that produces the hashed values.The function *M* is chosen, such that it is impossible to reverse the function to produce the plaintext from the hashed value (also called as *'hash'* or *'checksum'*).The common use of the hashing is to check the validity of the password. The hash thus produced should be greater that 128 bits to prevent a malicious user to break it. The only known attack against hash is the bruteforce attack which is almost impossible because of the combination of the plaintext (actually the password) chosen. So by choosing a complicated password with the mixed symbols and alphanumerice characters, adds strength to the hashing algorithm by increasing the available keyspace to search for the key. The commonly used algorithms are ***MD4, MD5, SHA*** and ***SHA1***.As ***MD*** stands for ***Message Digest*** and ***SHA*** stands for ***Secure Hash Algorithm***.

In unix systems, a hashing algorithm called **MD4** was used to generate the checksum or the hash and nowadays, it is replaced with a more stronger algorithm called **MD5** which was a derivative from **MD4** after fixing an unpublished flaw in one of the hashing rounds.

1.6 Latest Cryptographic Advancements

Due to the modern methods of attacks and advancement in the computational speeds ,
have taken over the well known secure secret-key algorithm, ***DES***, ***D****ata* ***E****ncryption* ***S****tandard,* which dominated the world of cryptography for decades. A 40 bit version of the DES algorithm can be cracked by any available modern computer and a 56 bit can be broken by any cheaply available Super Computer or by a simple beo-wulf cluster. That pose a security threat and people have developed, developing various other algorithms like ***3DES*** also known as ***Triple DES*** etc,.

The ***NIST*** *and* ***NSA*** plays an important role in deciding the algorithms. They analyze and recommend the algorithm for usage.

The latest trends have opened for the ***AES Advanced Encryptions Standards*** and many algorithms have been submitted for the review. Some of them approved and under analysis. They are IDEA, RC5, RC6 Rjindael, MARS, Two-fish, Blowfish, Diamond, Sapphire etc.

The inventions grow as the need increases. As per the truth, the importance of cryptography as given has a big response from scientific and legal bodies. Today, by the invention of high speed machines and high bandwidth transmission rates, people are planning to go further. The famous ***e=mc2*** has stepped into cryptography also. The Japanese have undertaken the improvement and implementation of ***quantum cryptography*** which uses the properties of the atoms and its inner particles for data integrity, which off course has a long way to go. The first commercially available quantum-cryptographic product is expected to be release around 2040's.

1.7 Applications

1.7.1 Web-Commerce

Web-commerce has grown into one of the fastest-growing sector of industry in the past two years. Billions of dollars have passed hands in the process and each entrepreneur wants a slice of the dough. To make this possible, data encryption plays a very central role in ensuring customers that paying for anything online is secure.

The reason beyond why does e-commerce need encryption is In order to enable secure online transaction, data encryption plays four important functions:

- **Digital authentication** allows both the customers and the merchant to be sure that they are dealing with whom the other party claims to be. This is absolutely necessary before sending credit card details to the merchant and also allow merchants to verify that the customer is the real owner of the credit card being used.
- **Integrity** ensures that the messages received are not changed during transmission by any third party.
- **Non-repudiation** prevents customers or merchants denying they ever received or sent a particular message or order.
- **In the event that information is intercepted**, encryption ensures privacy that prevents third parties from reading and/or using the information to their own advantage.

There are two methods of encryption employed in e-commerce:

- **Private-key encryption** (secret-key or symmetric encryption) in which users share a common key.
- **Public-key encryption** (also known as asymmetric encryption) where different keys are used for encryption and decryption.

These systems have their advantages and disadvantages and so secure transaction protocols such as Netscape's Secure Sockets Layer and Secure Electronic Transaction use a combination of both.

1.7.2:Networks Securiy :

Networks,their design, development, and usage are critical to our style of computing. We interact with networks daily, when we perform banking transactions, make telephone calls, or ride trains and planes. The utility companies use networks to track electricity or water usage and bill for it. When we pay for groceries or gasoline, networks enable our credit or debit card transactions and billing. Life without networks would be considerably less convenient, and many activities would be impossible. Not surprisingly, then, computing networks are attackers' targets of choice. Because of their actual and potential impact, network attacks attract the attention of journalists, managers, auditors, and the general public. For example, when you read the daily newspapers, you are likely to find a story about a network-based attack at least every month. The coverage itself evokes a sense of evil, using terms such as hijacking, distributed denial of service, and our familiar friends viruses, worms, and Trojan horses. Because any large-scale attack is likely to put thousands of computing systems at risk, with potential losses well into the millions of dollars, network attacks make good copy.

The media coverage is more than hype; network attacks are critical problems. Fortunately, your bank, your utility company, and even your Internet service provider take network security very seriously. Because they do, they are vigilant about applying the most current and most effective controls to their systems. Of equal importance, these organizations continually check their risks and learn about the latest attack types and defense mechanisms so that they can maintain the protection of their networks.

Chapter Two:Cryptography Algorithms

2.1 Substitution Ciphers [2]

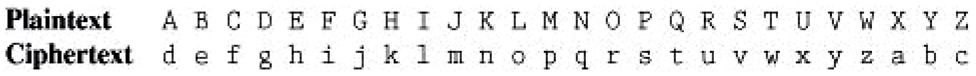
Children sometimes devise "secret codes" that use a correspondence table with which to substitute a character or symbol for each character of the original message. This technique is called a monoalphabetic cipher or **simple substitution**. A substitution is an acceptable way of encrypting text. In this section of the project, we will demonstrate several kinds of substitution ciphers.

2.1.1 The Caesar Cipher

The **Caesar cipher** has an important place in history. Julius Caesar is said to have been the first to use this scheme, in which each letter is translated to the letter a fixed number of places after it in the alphabet. Caesar used a shift of 3, so plaintext letter p_i was enciphered as ciphertext letter c_i by the rule

$$c_i = E(p_i) = p_i + 3$$

A full translation chart of the Caesar cipher is shown here.



Plaintext	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Ciphertext	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Using this encryption, the message

TREATY IMPOSSIBLE

would be encoded as

```
T R E A T Y   I M P O S S I B L E
w u h d w b   l p s r v v l e o h
```

2.1.2 Advantages and Disadvantages of the Caesar Cipher

Most ciphers, and especially the early ones, had to be easy to perform in the field. In particular, it was dangerous to have the cryptosystem algorithms written down for the soldiers or spies to follow. Any cipher that was so complicated that its algorithm had to be written out was at risk of being revealed if the interceptor caught a sender with the written instructions. Then, the interceptor could readily decode any ciphertext messages intercepted (until the encryption algorithm could be changed).

The Caesar cipher is quite simple. During Caesar's lifetime, the simplicity did not dramatically compromise the safety of the encryption because anything written, even in plaintext, was rather well protected; few people knew how to read! The pattern $p_i + 3$ was easy to memorize and implement. A sender in the field could write out a plaintext and a ciphertext alphabet, encode a message to be sent, and then destroy the paper containing the alphabets.

Its obvious pattern is also the major weakness of the Caesar cipher. A secure encryption should not allow an interceptor to use a small piece of the ciphertext to predict the entire pattern of the encryption.

2.1.3 Other Substitutions

In substitutions, the alphabet is scrambled, and each plaintext letter maps to a unique ciphertext letter. We can describe this technique in a more mathematical way. Formally, we say that a **permutation** is a reordering of the elements of a sequence. For instance, we can permute the numbers l to 10 in many ways, including the permutations $\pi_1 = 1, 3, 5, 7, 9, 10, 8, 6, 4, 2$; and $\pi_2 = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1$. A permutation is a function, so we can write expressions such as $\pi_1(3) = 5$ meaning that the letter in position 3 is to be replaced by the fifth letter. If the set is the first ten letters of the alphabet, $\pi_1(3) = 5$ means that `c` is transformed into `E`.

One way to scramble an alphabet is to use a key, a word that controls the permutation. For instance, if the key is `word`, the sender or receiver first writes the alphabet and then writes the key under the first few letters of the alphabet.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
word
```

The sender or receiver then fills in the remaining letters of the alphabet, in some easy-to-remember order, after the keyword.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
wordabcefghijklmnpqstuvxyz
```

In this example, the key is short, so most plaintext letters are only one or two positions off from their ciphertext equivalents. With a longer keyword, the distance is greater and less predictable, as shown below. Because π must map one plaintext letter to exactly one ciphertext letter, duplicate letters in a keyword, such as the second *s* and *o* in `professional`, are dropped.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
profesinalbcdghjkmqtuvwxyz
```

Notice that near the end of the alphabet replacements are rather close, and the last seven characters map to themselves. Conveniently, the last characters of the alphabet are among the least frequently used, so this vulnerability would give little help to an interceptor.

Still, since regularity helps an interceptor, it is desirable to have a less regular rearrangement of the letters. One possibility is to count by threes (or fives or sevens or nines) and rearrange the letters in that order. For example, one encryption uses a table that starts with

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
adgj
```

Using every third letter. At the end of the alphabet, the pattern continues mod 26, as shown below.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
adgjmpsvybehknqtwzcfilorux
```

We can use $c(a)=(3*a) \bmod 26$ for odd place characters in a message ,and $c(a)=((5*a)+13) \bmod 26$ for even place characters.
All substitution ciphers above were implemented successfully as a computer program that is shown in AppendixA.

2.2 One-Time Pads

A **one-time pad** is sometimes considered the perfect cipher. The name comes from an encryption method in which a large, non-repeating set of keys is written on sheets of paper, glued together into a pad. For example, if the keys are 20 characters long and a sender must transmit a message 300 characters in length, the sender would tear off the next 15 pages of keys. The sender would write the keys one at a time above the letters of the plaintext and encipher the plaintext with a prearranged chart (called a **Vigenère tableau**) that has all 26 letters in each column, in some scrambled order. The sender would then destroy the used keys.

For the encryption to work, the receiver needs a pad identical to that of the sender. Upon receiving a message, the receiver takes the appropriate number of keys and deciphers the message as if it were a plain substitution with a long key. Essentially, this algorithm gives the effect of a key as long as the number of characters in the pad.

The one-time pad method has two problems: the need for absolute synchronization between sender and receiver, and the need for an unlimited number of keys. Although generating a large number of random keys is no problem, printing, distributing, storing, and accounting for such keys are problems.

2.2.1 Long Random Number Sequences

A close approximation of a one-time pad for use on computers is a random number generator. In fact, computer random numbers are not random; they really form a sequence with a very long period (that is, they go for a long time before repeating the sequence). In practice, a generator with a long period can be acceptable for a limited amount of time or plaintext.

To use a random number generator, the sender with a 300-character message would interrogate the computer for the next 300 random numbers, scale them to lie between 0 and 25, and use one number to encipher each character of the plaintext message.

2.2.2 The Vernam Cipher

The **Vernam cipher** is a type of one-time pad devised by Gilbert Vernam for AT&T. The Vernam cipher is immune to most cryptanalytic attacks. The basic encryption involves an arbitrarily long non-repeating sequence of numbers that are combined with the plaintext.Vernam's invention used an arbitrarily long punched paper tape that fed into a teletype machine. The tape contained random numbers that were combined with characters typed into the teletype. The sequence of random numbers had no repeats, and each tape was used only once. As long as the key tape does not repeat or is not reused, this type of cipher is immune to cryptanalytic attack because the available ciphertext does not display the pattern of the key. A model of this process is shown in Fig. 2-1.

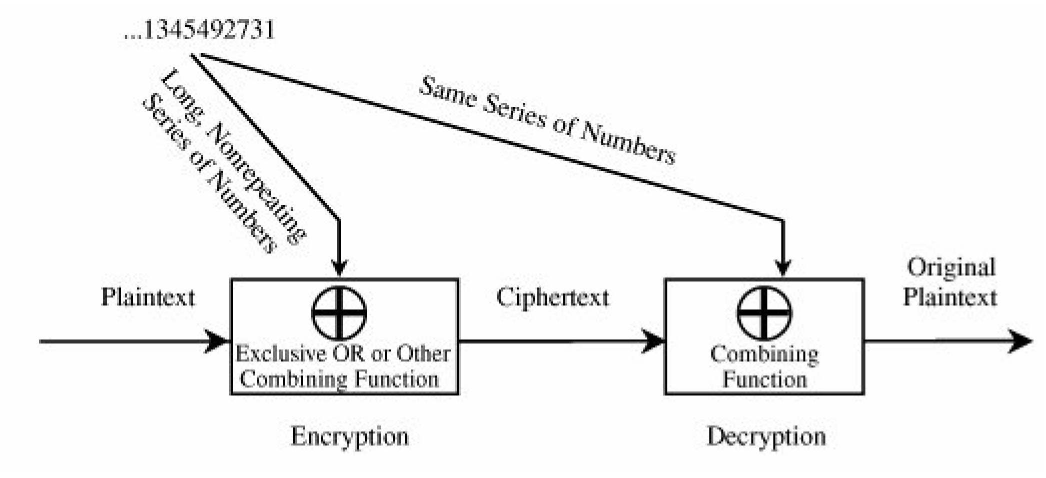


Figure 2-1 Vernam Cipher

To see how this method works, we perform a simple Vernam encryption. Assume that the alphabetic letters correspond to their counterparts in arithmetic notation mod 26. That is, the letters are represented with numbers 0 through 25. To use the Vernam cipher, we sum this numerical representation with a stream of random two-digit numbers. For instance, if the message is

VERNAM CIPHER

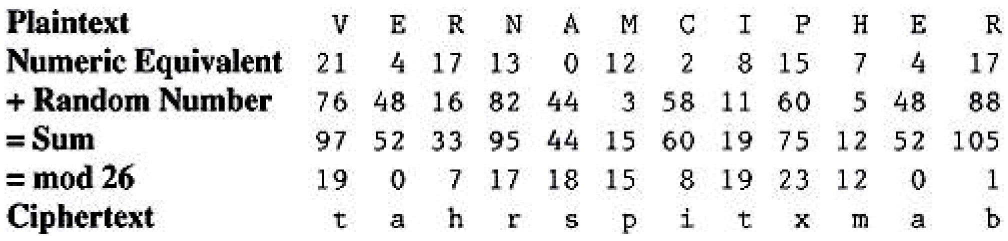
the letters would first be converted to their numeric equivalents, as shown here.

V	E	R	N	A	M	C	I	P	H	E	R
21	4	17	13	0	12	2	8	15	7	4	17

Next, we generate random numbers to combine with the letter codes. Suppose the following series of random two-digit numbers is generated.

76 48 16 82 44 03 58 11 60 05 48 88

The encoded form of the message is the sum mod 26 of each coded letter with the corresponding random number. The result is then encoded in the usual base-26 alphabet representation.



Plaintext	V	E	R	N	A	M	C	I	P	H	E	R
Numeric Equivalent	21	4	17	13	0	12	2	8	15	7	4	17
+ Random Number	76	48	16	82	44	3	58	11	60	5	48	88
= Sum	97	52	33	95	44	15	60	19	75	12	52	105
= mod 26	19	0	7	17	18	15	8	19	23	12	0	1
Ciphertext	t	a	h	r	s	p	i	t	x	m	a	b

Thus, the message

VERNAM CIPHER

is encoded as

tahrsp itxmab

In this example, the repeated random number 48 happened to fall at the places of repeated letters, accounting for the repeated ciphertext letter a; such a repetition is highly unlikely. The repeated letter t comes from different plaintext letters, a much more likely occurrence. Duplicate ciphertext letters are generally unrelated when this encryption algorithm is used.

2.2.3 Book Ciphers

Another source of supposedly "random" numbers is any book, piece of music, or other object of which the structure can be analyzed. Both the sender and receiver need access to identical objects. For example, a possible one-time pad can be based on a telephone book. The sender and receiver might agree to start at page 35 and use two middle digits (ddd-DDdd) of each seven-digit phone number, mod 26, as a key letter for a substitution cipher.

They use an already agreed-on table (a Vigenère tableau) that has all 26 letters in each column, in some scrambled order.

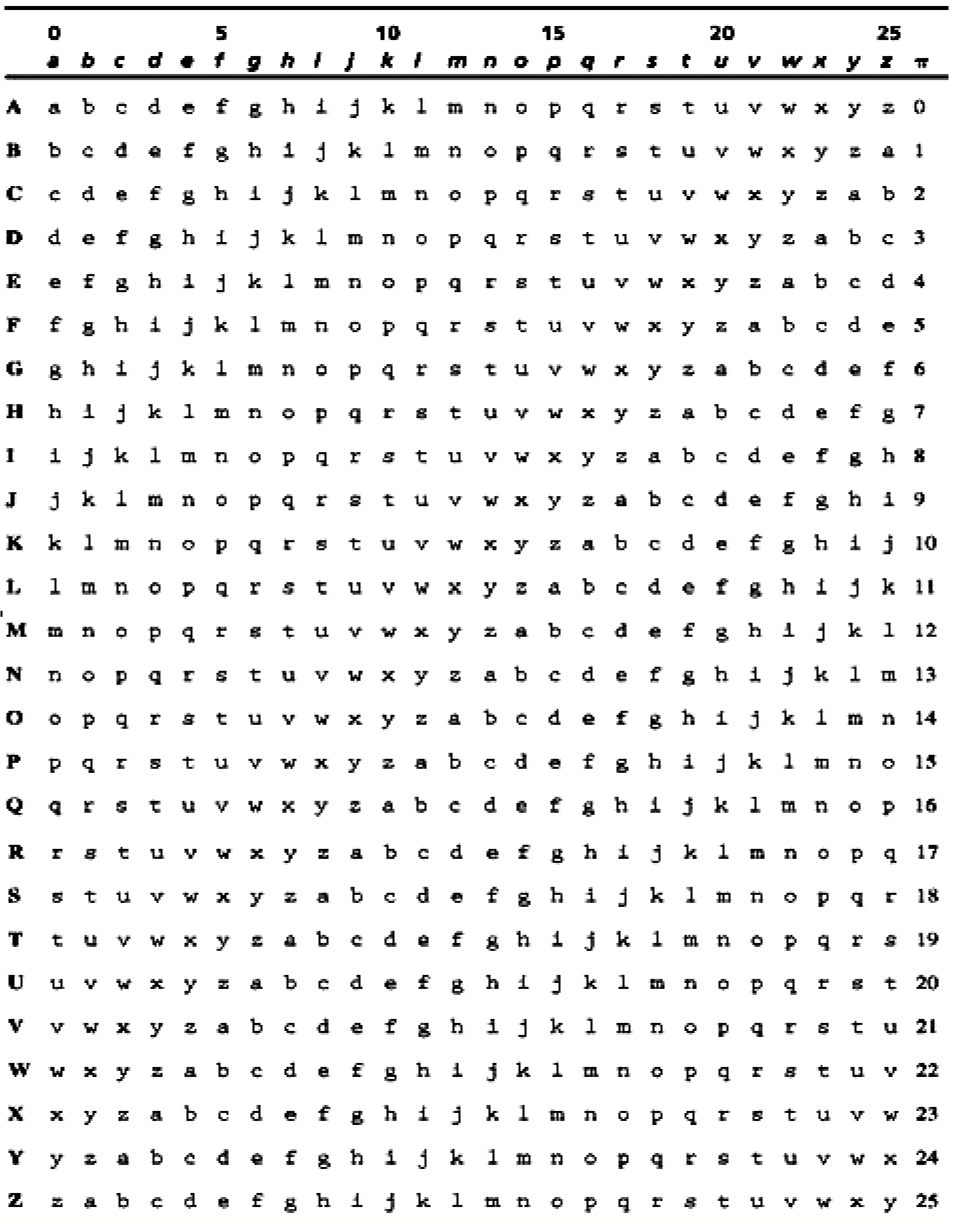
Any book can provide a key. The key is formed from the letters of the text, in order. This type of encryption was the basis for Ken Follett's novel, The Key to Rebecca, in which Daphne du Maurier's famous thriller acted as the source of keys for spies in World War II. Were the sender and receiver known to be using a popular book, such as The Key to Rebecca, the bible, or Security in Computing, it would be easier for the cryptanalyst to try books against the ciphertext, rather than look for patterns and use sophisticated tools.

As an example of a book cipher, we might select a passage from Descarte's meditation: What of thinking? I am, I exist, that is certain. The meditation goes on for great length, certainly long enough to encipher many very long messages. To encipher the message MACHINES CANNOT THINK by using the Descartes key, we would write the message under enough of the key and encode the message by selecting the substitution in row p_i, column k_i.

```
iamie xistt hatis cert
MACHI NESCA NNOTT HINK
```

If we use the substitution table shown as Table 2-1, this message would be encrypted as `uaopm kmkvt unhbl jmed` because row *M* column *i* is *u*, row *A* column *a* is *a*, and so on.

Table 2-1 Vigenère Tableau



	0					5					10					15					20					25	
	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*	*π*
A	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	0
B	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	1
C	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	2
D	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	3
E	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	4
F	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	5
G	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	6
H	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	7
I	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	8
J	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	9
K	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	10
L	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	11
M	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	12
N	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	13
O	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	14
P	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	15
Q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	16
R	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	17
S	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	18
T	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	19
U	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	20
V	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	21
W	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	22
X	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	23
Y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	24
Z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	25

The flaw in this cipher lies in the fact that neither the message nor the key text is evenly distributed; in fact, the distributions of both cluster around high-frequency letters. For example, the four letters *A, E, O,* and *T* account for approximately 40 percent of all letters used in standard English text. Each ciphertext letter is really the intersection of a plaintext letter and a key letter. But if the probability of the plaintext or the key letter's being *A*, *E*, *O*, or *T* is *0.4*, the probability of both being one of the four is *0.4 * 0.4 = 0.16*, or nearly one in six. Using the top six letters (adding *N* and *I*) increases the sum of the frequencies to *50* percent and thus increases the probability for a pair to *0.25*, or one in four.

We look for frequent letter pairs that could have generated each ciphertext letter. The encrypted version of the message MACHINES CANNOT THINK is

uaopm kmkvt unhbl jmed

To break the cipher, assume that each letter of the ciphertext comes from a situation in which the plaintext letter (row selector) and the key letter (column selector) are both one of the six most frequent letters. (As we calculated before, this guess will be correct approximately 25 percent of the time.) The trick is to work the cipher inside out. For a ciphertext letter, look in the body of the table for the letter to appear at the intersection of one of the six rows with one of the six columns. Find combinations in the Vigenère tableau that could yield each ciphertext letter as the result of two high-frequency letters.

Searching through this table for possibilities, we transform the cryptogram.

Ciphertext	u	a	o	p	m	k	m	k	v	t	u	n	h	b	l	j	m	e	d
Possible	?	A	A	?	E	?	E	?	?	A	?	A	O	N	?	?	E	A	?
Plaintext			O		I		I			T		N	T	T			I	E	
					T		T										T		

This technique does not reveal the entire message, or even enough of it to make the message `MACHI NESCA NNOTT HINK` easy to identify. The technique did, however, make predictions in ten letter positions, and there was a correct prediction in seven of those ten positions. The algorithm made 20 assertions about probable letters, and eight of those 20 were correct. (A score of 8 out of 20 is 40 percent, even better than the 25 percent expected.) The algorithm does not come close to solving the cryptogram, but it substantially reduces the 26^{19} possibilities for the analyst to consider. Giving this much help to the cryptanalyst is significant. A similar technique can be used even if the order of the rows is permuted.

Also, we want to stress that a one-time pad cannot repeat. If there is any repetition, the interceptor gets two streams of ciphertext: one for one block of plaintext, the other for a different plaintext, but both encrypted using the same key. The interceptor combines the two ciphertexts in such a way that the keys cancel each other out, leaving a combination of the two plaintexts. The interceptor can then do other analysis to expose patterns in the underlying plaintexts and give some likely plaintext elements. The worst case is when the user simply starts the pad over for a new message, for the interceptor may then be able to determine how to split the plaintexts and unzip the two plaintexts intact.

2.3 Transpositions (Permutations)

The goal of substitution is confusion; the encryption method is an attempt to make it difficult for a cryptanalyst or intruder to determine how a message and key were transformed into ciphertext. In this section, we look at a different kind of scrambling with the similar goal. A **transposition** is an encryption in which the letters of the message are rearranged. With transposition, the cryptography aims for diffusion, widely spreading the information from the message or the key across the ciphertext. Transpositions try to break established patterns. Because a transposition is a rearrangement of the symbols of a message, it is also known as a permutation.

2.3.1 Columnar Transpositions

As with substitutions, we begin this study of transpositions by examining a simple example. The **columnar transposition** is a rearrangement of the characters of the plaintext into columns.

The following set of characters is a five-column transposition. The plaintext characters are written in rows of five and arranged one row after another, as shown here.

c_1	c_2	c_3	c_4	c_5
c_6	c_7	c_8	c_9	c_{10}
c_{11}	c_{12}	etc.		

Figure 2-2 Arrangement of characters

We form the resulting ciphertext by reading down the columns, as shown in Fig. 2-3.

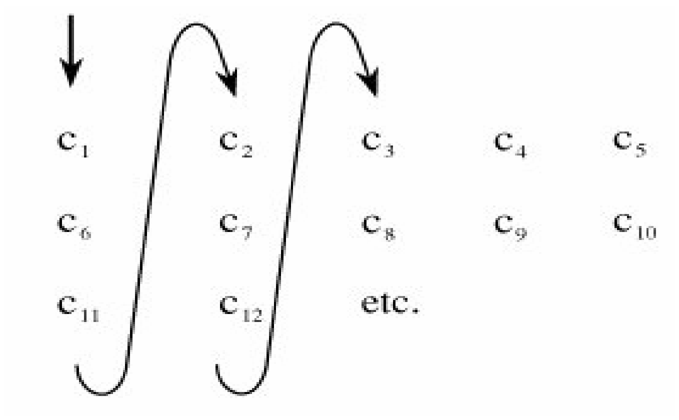

c_1 c_2 c_3 c_4 c_5
c_6 c_7 c_8 c_9 c_{10}
c_{11} c_{12} etc.

Figure 2-3 Columnar Transposition

For instance, suppose you want to write the plaintext message `THIS IS A MESSAGE TO SHOW HOW A COLUMNAR TRANSPOSITION WORKS`. We arrange the letters in five columns as

```
T  H  I  S  I
S  A  M  E  S
S  A  G  E  T
O  S  H  O  W
H  O  W  A  C
O  L  U  M  N
A  R  T  R  A
N  S  P  O  S
I  T  I  O  N
W  O  R  K  S
```

The resulting ciphertext would then be read down the columns as

```
tssoh oaniw haaso lrsto imghw
utpir seeoa mrook istwc nasns
```

In this example, the length of this message happens to be a multiple of five, so all columns are the same length. However, if the message length is not a multiple of the length of a row, the last columns will be one or more letters short. When this happens, we sometimes use an infrequent letter, such as `X`, to fill in any short columns.

The One-time pads and transposition algorithms were implemented successfully as a computer program that is shown in AppendixA.

2.4 Combinations of Approaches

Substitution and transposition can be considered as building blocks for encryption. Other techniques can be based on each of them, both of them, or a combination with yet another approach. each technique is only one piece of the larger puzzle. Just as we could have a locked car inside a locked garage, we could also combine various approaches to encryption to strengthen the overall security of our system. A combination of two ciphers is called a **product cipher**. Product ciphers are typically performed one after another, as in $E_2(E_1(P,k_1), k_2)$. Just because of applying two ciphers does not necessarily mean the result is any stronger than, or even as strong as, either individual cipher.

2.5 Data Encryption Standard [2]

The symmetric systems provide a two-way channel to their users: A and B share a secret key, and they can both encrypt information to send to the other as well as decrypt information from the other. The symmetry of this situation is a major advantage.

As long as the key remains secret, the system also provides authentication, proof that a message received was not fabricated by someone other than the declared sender.Authenticity is ensured because only the legitimate sender can produce a message that will decrypt properly with the shared key.

The Data Encryption Standard (DES) is a system developed for the U.S. government for use by the general public. It has been officially accepted as a cryptographic standard both in the United States and abroad. Many hardware and software systems use the DES. However, its adequacy has recently been questioned.

2.5.1 Overview of the DES Algorithm

Recall that the strength of the DES algorithm derives from repeated application of substitution and permutation, one on top of the other, for a total of 16 cycles. That is, plaintext is affected by a series of cycles of a substitution then a permutation. The iterative substitutions and permutations are performed as outlined in Fig. 2-4

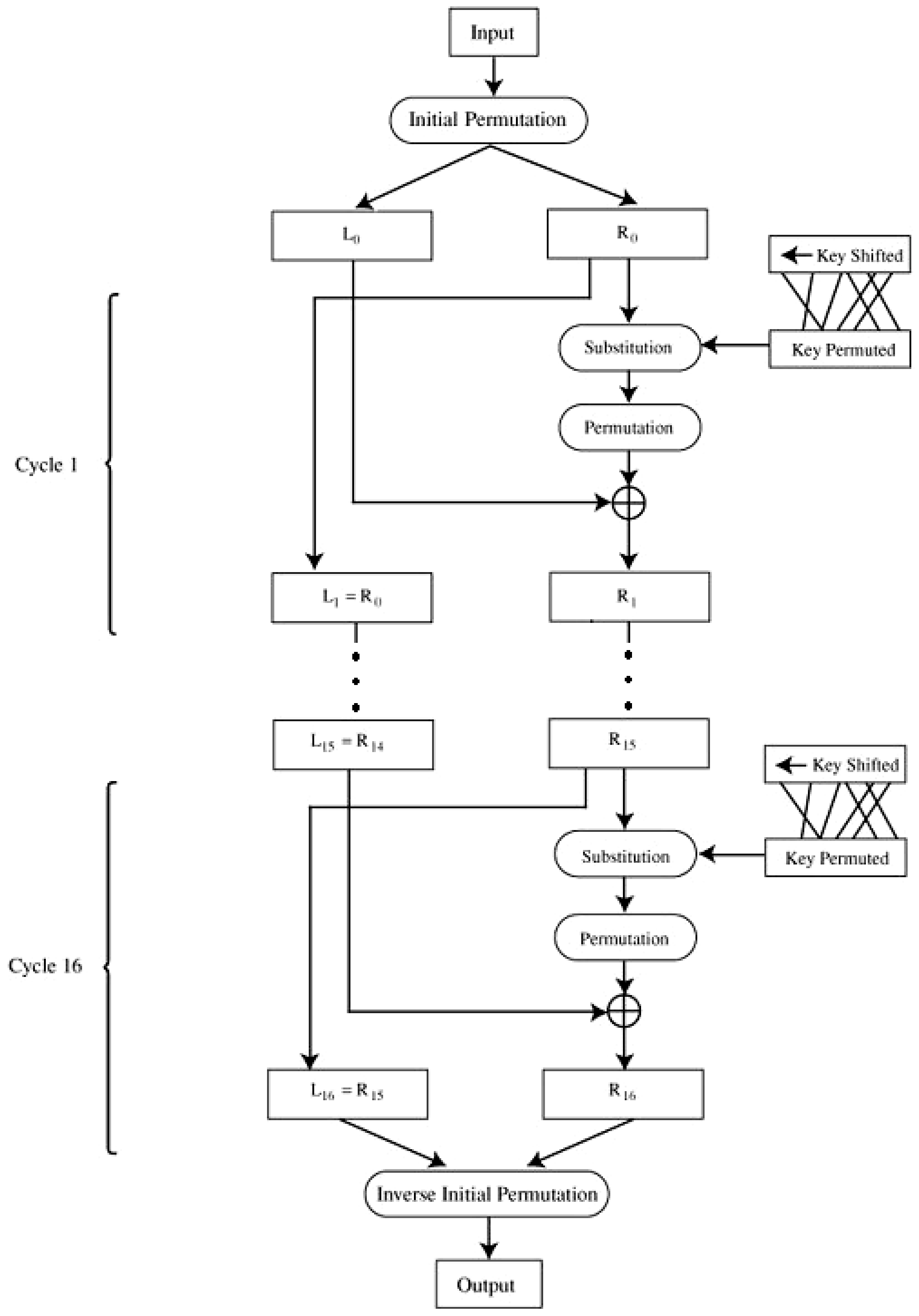


Figure 2-4 Cycles of Substitution and Permutation

The algorithm uses only standard arithmetic and logical operations on up to 64-bit numbers, so it is suitable for implementation in software on most current computers. Although complex, the algorithm is repetitive, making it suitable for implementation on a single-purpose chip. In fact, several such chips are available on the market for use as basic components in devices that use DES encryption in an application.

2.5.2 Details of the Encryption Algorithm

The basis of the DES is two different ciphers, applied alternately. Shannon noted that two weak but complementary ciphers can be made more secure by being applied together (called the "product" of the two ciphers) alternately, in a structure called a product cipher. The product of two ciphers is depicted in Fig. 2-5

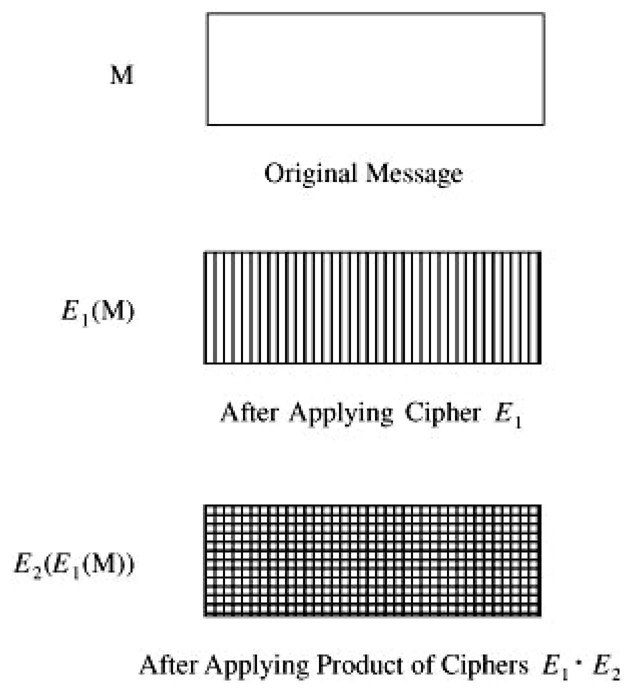

M
Original Message
E_1(M)
After Applying Cipher E_1
$E_2(E_1$(M))
After Applying Product of Ciphers $E_1 \cdot E_2$

Figure 2-5 Product Ciphers

After initialization, the DES algorithm operates on blocks of data. It splits a data block in half, scrambles each half independently, combines the key with one half, and swaps the two halves. This process is repeated 16 times. It is an iterative algorithm using just table lookups and simple bit operations. Although the bit-level manipulations of the algorithm are complex, the algorithm itself can be implemented quite efficiently.

Input to the DES is divided into blocks of 64 bits. The 64 data bits are permuted by a so-called initial permutation. The data bits are transformed by a 64-bit key (of which only 56 bits are used). The key is reduced from 64 bits to 56 bits by dropping bits 8, 16, 24, … 64 (where the most significant bit is named bit "1"). These bits are assumed to be parity bits that carry no information in the key.

Next begins the sequence of operations known as a cycle. The 64 permuted data bits are broken into a left half and a right half of 32 bits each. The key is shifted left by a number of bits and permuted. The key is combined with the right half, which is then combined with the left half. The result of these combinations becomes the new right half; the old right half becomes the new left half. This sequence of activities, which constitutes a cycle, is shown in Fig. 2-6. The cycles are repeated 16 times. After the last cycle is a final permutation, which is the inverse of the initial permutation.

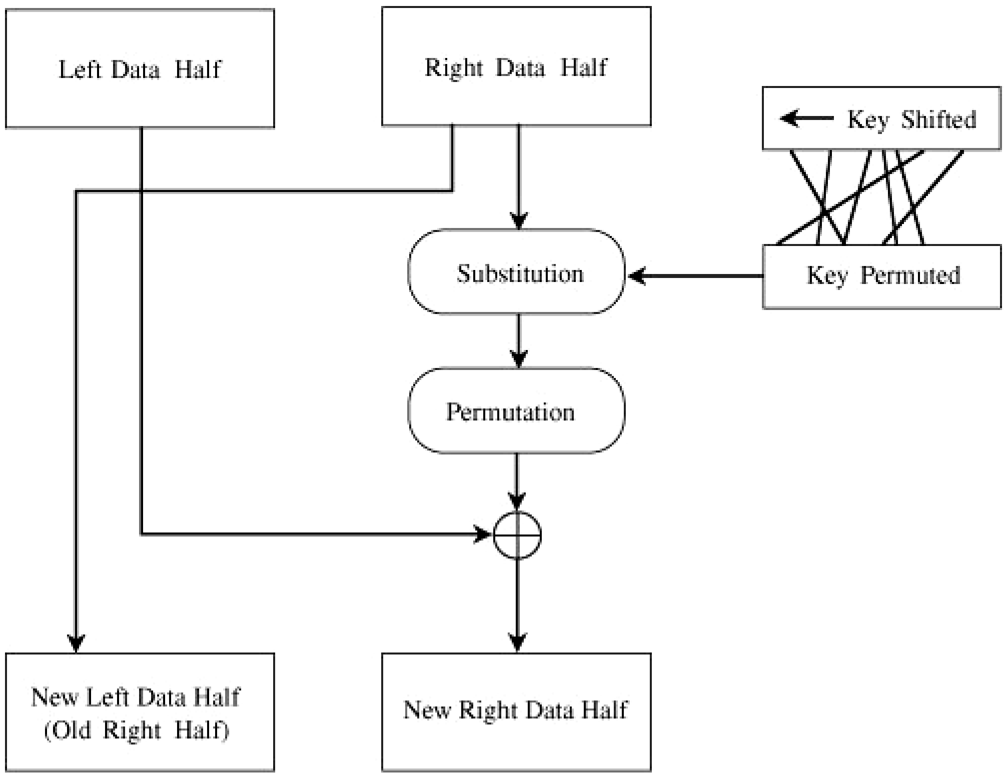


Figure 2-6 A Cycle in the DES

For a 32-bit right half to be combined with a 64-bit key, two changes are needed. First, the algorithm expands the 32-bit half to 48 bits by repeating certain bits, while reducing the 56-bit key to 48 bits by choosing only certain bits. These last two operations, called **expansion permutations** and permuted choices, are shown in the diagram of Fig. 2-7.

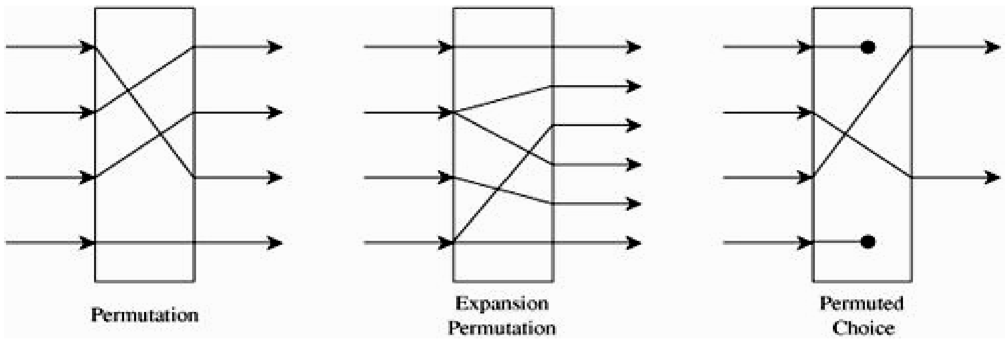


Figure 2-7 Types of Permutations

2.5.3 Details of Each Cycle of the Algorithm

Each cycle of the algorithm is really four separate operations. First, a right half is expanded from 32 bits to 48. Then, it is combined with a form of the key. The result of this operation is then substituted for another result and condensed to 32 bits at the same time. The 32 bits are permuted and then combined with the left half to yield a new right half. This whole process is shown in Fig. 2-8.

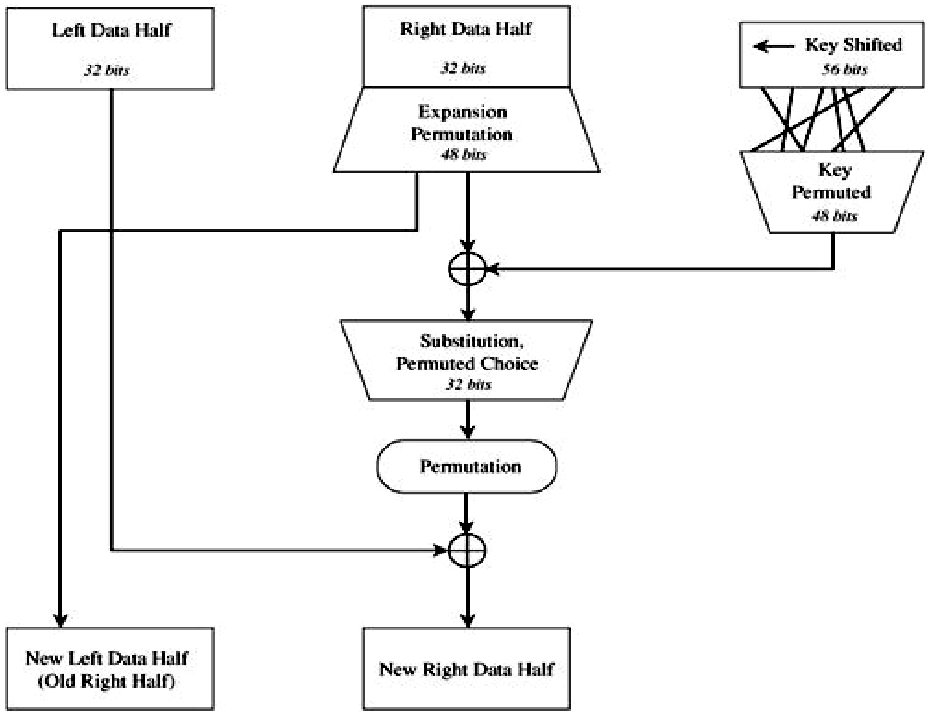


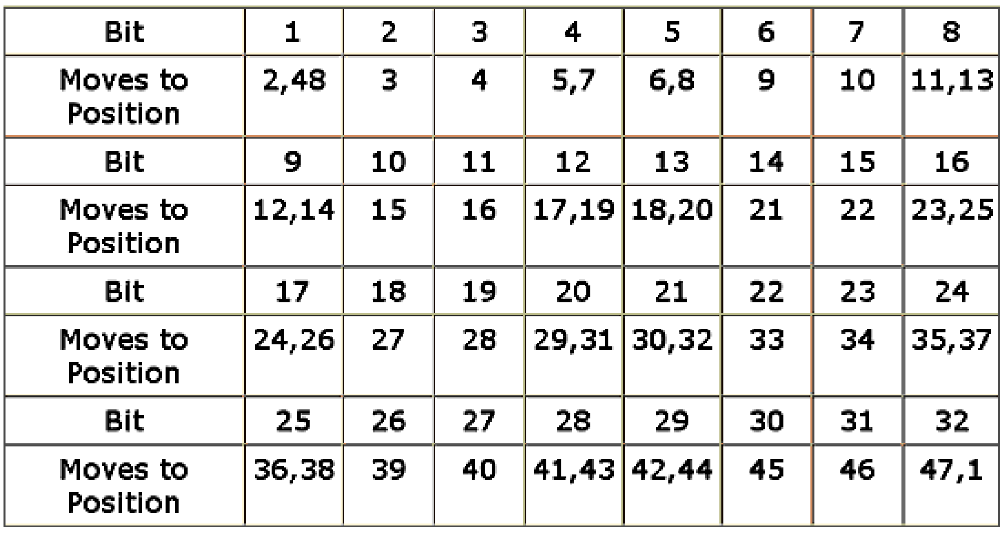
Figure 2-8 Details of a Cycle

2.5.4 Expansion Permutation

Each right half is expanded from 32 to 48 bits by means of the expansion permutation. The expansion permutes the order of the bits and also repeats certain bits. The expansion has two purposes: to make the intermediate halves of the ciphertext comparable in size to the key and to provide a longer result that can later be compressed.

The expansion permutation is defined by Table 2-2. For each 4-bit block, the first and fourth bits are duplicated, while the second and third are used only once. This table shows to which output position(s) the input bits move. Since this is an expansion permutation, some bits move to more than one position. Each row of the table shows the movement of eight bits. The interpretation of this table is that bit 1 moves to positions 2 and 48 of the output, while bit 10 moves to position 15. A portion of the pattern is also shown in Fig. 2-9.

Table 2-2 Expansion Permutation.



Bit	1	2	3	4	5	6	7	8
Moves to Position	2,48	3	4	5,7	6,8	9	10	11,13
Bit	9	10	11	12	13	14	15	16
Moves to Position	12,14	15	16	17,19	18,20	21	22	23,25
Bit	17	18	19	20	21	22	23	24
Moves to Position	24,26	27	28	29,31	30,32	33	34	35,37
Bit	25	26	27	28	29	30	31	32
Moves to Position	36,38	39	40	41,43	42,44	45	46	47,1

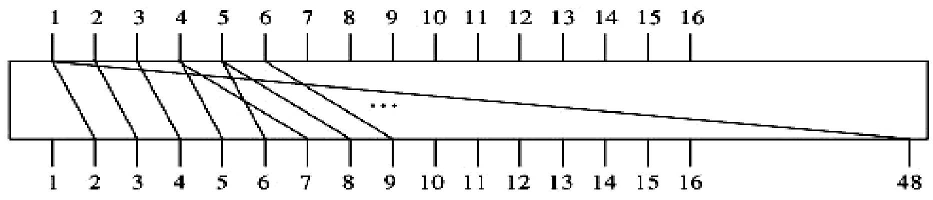


Figure 2-9 Pattern of Expansion Permutation

2.5.5 Key Transformation

As described above, the 64-bit key immediately becomes a 56-bit key by deletion of every eighth bit. At each step in the cycle, the key is split into two 28-bit halves, the halves are shifted left by a specified number of digits, the halves are then pasted together again, and 48 of these 56 bits are permuted to use as a key during this cycle.

Next, the key for the cycle is combined by an exclusive OR function with the expanded right half. That result moves into the S-boxes we are about to describe.

At each cycle, the halves of the key are independently shifted left circularly by a specified number of bit positions. The number of bits shifted is given in Table 2-3.

Table 2-3 Bits Shifted by Cycle Number

Cycle Number	Bits Shifted
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

After being shifted, 48 of the 56 bits are extracted for the exclusive OR combination with the expanded right half. The choice permutation that selects these 48 bits is shown in Table 2-4. For example, from this table we see that bit 1 of the shifted key goes to output position 5, and bit 9 is ignored in this cycle.

Table 2-4 Choice Permutation to Select 48 Key Bits

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

2.5.6 S-Boxes

Substitutions are performed by eight **S-boxes.** An S-box is a permuted choice function by which six bits of data are replaced by four bits. The 48-bit input is divided into eight 6-bit blocks, identified as $B_1B_2... B_8$; block B_j is operated on by S-box S_j.

The S-boxes are substitutions based on a table of 4 rows and 16 columns. Suppose that block B_j is the six bits $b_1b_2b_3b_4b_5b_6$. Bits b_1 and b_6, taken together, form a two-bit binary number b_1b_6, having a decimal value from 0 to 3. Call this value r. Bits b_2, b_3, b_4, and b_5 taken together form a 4-bit binary number $b_2b_3b_4b_5$, having a decimal value from 0 to 15. Call this value c. The substitutions from the S-boxes transform each 6-bit block B_j into the 4-bit result shown in row r, column c of section S_i of Table 2-5. For example, assume that block B_7 in binary is 010011. Then, r = 01 = 1 and c = 1001 = 9. The transformation of block B_7 is found in row 1, column 9 of section 7 of Table 2-5. The value 3 = 0011 is substituted for the value 010011.

Table 2-5 S-Boxes of DES

		Column															
Box	**Row**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
S_1																	
	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2																	
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																	
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																	
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																	
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																	
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

2.5.7 P-Boxes

After an S-box substitution, all 32 bits of a result are permuted by a straight permutation, P. Table 2-6 shows the position to which bits are moved. Eight bits are shown on each row. For example, bit 1 of the output of the substitution moves to bit 9, and bit 10 moves to position 16.

Table 2-6 Permutation Box P

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

2.5.8 Initial and Final Permutations

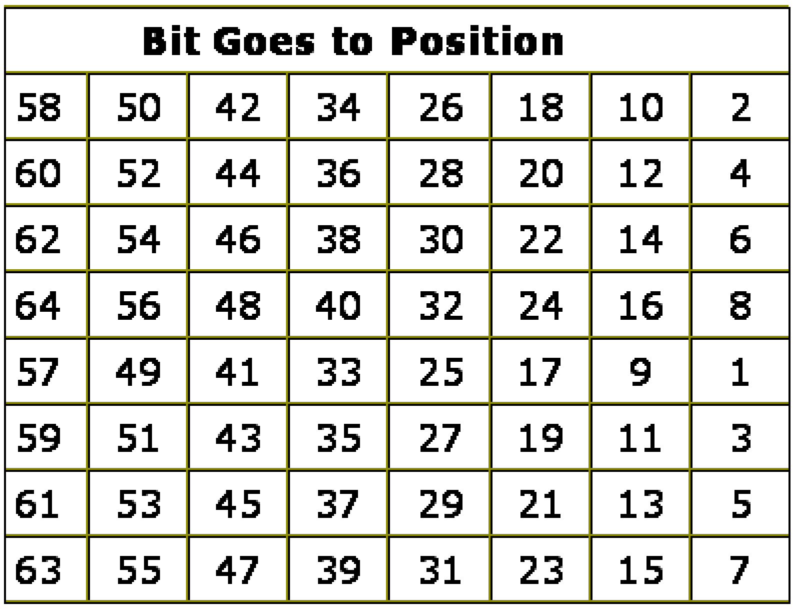
The DES algorithm begins with an initial permutation that reorders the 64 bits of each input block. The initial permutation is shown in Table 2-7.

Table 2-7 Initial Permutation

Bit Goes to Position							
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

At the conclusion of the 16 substitution-permutation rounds, the DES algorithm finishes with a final permutation (or inverse initial permutation), which is shown in Table 2-8.

Table 2-8 Final Permutation (Inverse Initial Permutation)



Bit Goes to Position							
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
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

2.5.9 Complete DES

Now we can put all the pieces back together. First, the key is reduced to 56 bits. Then, a block of 64 data bits is permuted by the initial permutation. Following are 16 cycles in which the key is shifted and permuted, half of the data block is transformed with the substitution and permutation functions, and the result is combined with the remaining half of the data block. After the last cycle, the data block is permuted with the final permutation.

2.5.10 The Decryption Process

Decryption in DES is exactly the same as encryption save that 16 iterations of the key are used in reverse order from K_{16} to K_1 on the last. This means that the algorithm acts as its own inverse.

2.5.11 Security of the DES

The cryptanalytic attacks described here have not exposed any significant, exploitable vulnerabilities in the design of DES. But the weakness of the 56-bit key is now apparent. Although the amount of computing power or time needed is still significant enough to deter casual DES key browsing, a dedicated adversary could succeed against a specific DES ciphertext of significant interest.

Does this mean the DES is insecure? No, not yet. Nobody has yet shown serious flaws in the DES. With a triple DES approach, the effective key length is raised from 56 bits to 112 or 168 bits, increasing the difficulty of attack exponentially. In the near term (years, probably decades) triple DES is strong enough to protect even significant commercial data (such as financial data or patient medical records). Still, DES is nearing the end of its useful lifetime, and a replacement is in order. With millions of computers in the world, clearly DES is inadequate to protect sensitive information with a modest time value. Similarly, algorithms with key lengths of 64 and 80 bits may be strong enough for a while, but an improvement in processor speeds and number of parallel computers threatens those, too.

2.5.12 Weaknesses of the DES

The DES algorithm also has known weaknesses, but these weaknesses are not believed to be serious limitations of the algorithm's effectiveness.

2.5.12.1 Complements

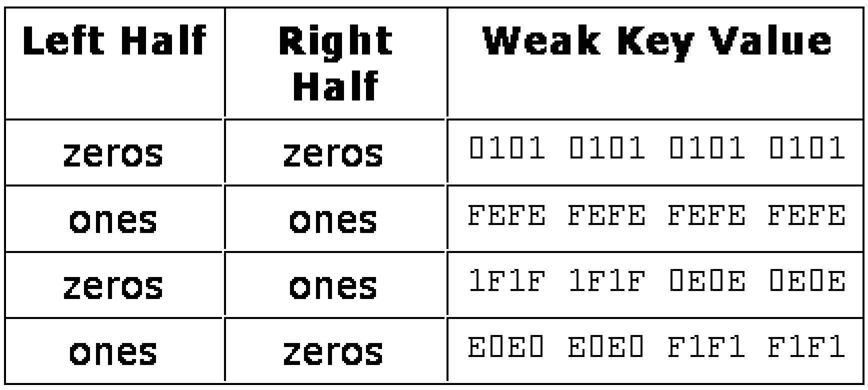
The first known weakness concerns complements. (Throughout this discussion, "complement" means "ones complement," the result obtained by replacing all 1s by 0s and 0s by 1s in a binary number.) If a message is encrypted with a particular key, the complement of that encryption will be the encryption of the complement message under the complement key. Stated formally, let *p* represent a plaintext message and *k* a key, and let the symbol $\neg x$ mean the complement of the binary string *x*. If $c = DES(p, k)$ (meaning *c* is the DES encryption of *p* using key *k*), then $\neg c = DES(\neg p, \neg k)$. Since most applications of encryption do not deal with complement messages and since users can be warned not to use complement keys, this problem is not serious.

2.5.12.2 Weak Keys

A second known weakness concerns choice of keys. Because the initial key is split into two halves and the two halves are independently shifted circularly, if the value being shifted is all 0s or all 1s, then the key used for encryption in each cycle is the same as for all other cycles. Remember that the difference between encryption and decryption is that the key shifts are applied in reverse. Key shifts are right shifts, and the number of positions shifted is taken from the bottom of the table up, instead of top down. But if the keys are all 0s or all 1s anyway, right or left shifts by 0, 1, or 2 positions are all the same. For these keys, encryption is the same as decryption: $c = DES(p, k)$, and $p = DES(c, k)$. These keys are called "weak keys." The same thing happens if one half of the key is all 0s and the other half is all 1s. Since these keys are known, they can simply be avoided, so this, too, is not a serious problem.

The four weak keys are shown in hexadecimal notation in Table 2-9. (The initial key permutation extracts every eighth bit as a parity bit and scrambles the key order slightly. Therefore, the "half zeros, half ones" keys are not just split in the middle.)

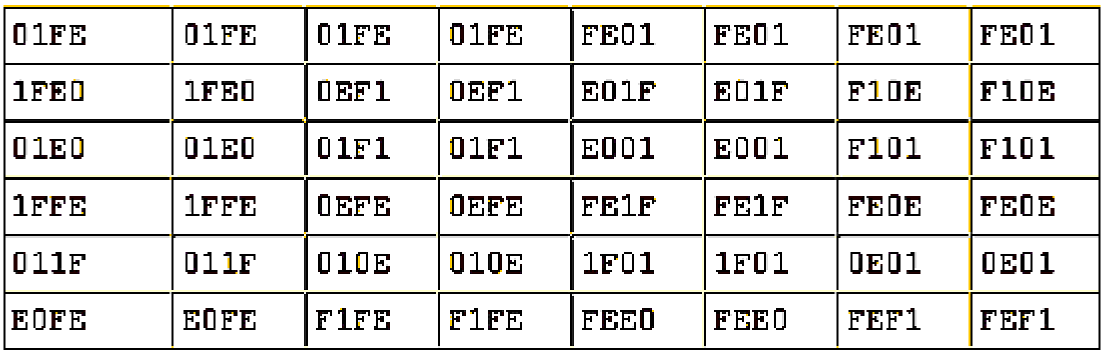
Table 2-9 Weak DES Keys



Left Half	Right Half	Weak Key Value
zeros	zeros	0101 0101 0101 0101
ones	ones	FEFE FEFE FEFE FEFE
zeros	ones	1F1F 1F1F 0E0E 0E0E
ones	zeros	E0E0 E0E0 F1F1 F1F1

2.5.12.3 Semiweak Keys

A third difficulty is similar: Specific pairs of keys have identical decryption. That is, there are two different keys, k_1 and k_2, for which $c = DES(p, k_1)$ and $c = DES(p, k_2)$. This similarity implies that k_1 can decrypt a message encrypted under k_2. These so called semiweak keys are shown in Table 2-10. Other key patterns have been investigated with no additional weaknesses found to date. We should, however, avoid any key having an obvious pattern such as these.



01FE	01FE	01FE	01FE	FE01	FE01	FE01	FE01
1FE0	1FE0	0EF1	0EF1	E01F	E01F	F10E	F10E
01E0	01E0	01F1	01F1	E001	E001	F101	F101
1FFE	1FFE	0EFE	0EFE	FE1F	FE1F	FE0E	FE0E
011F	011F	010E	010E	1F01	1F01	0E01	0E01
E0FE	E0FE	F1FE	F1FE	FEE0	FEE0	FEF1	FEF1

Table 2-10 Semiweak DES Key Pairs

2.5.12.4 Key Length

IBM's original submission to NBS had a 112-bit key. By the time the DES became a standard ,that was reduced to a 56-bit key. many cryptographers argued for the longer key. Their arguments centered on the possibility of a brute-force attack. but (2 to power 56) possible keys were satisfying to some extent.

2.5.12.5 Number of Iterations

Many analysts wonder whether 16 iterations are sufficient. Since each iteration diffuses the information of the plaintext throughout the ciphertext, it is not clear that 16 cycles diffuse the information sufficiently. For example, with only one cycle, a single ciphertext bit is affected only by a few bits of plaintext. With more cycles, the diffusion becomes greater, so ideally no one ciphertext bit depends on any subset of plaintext bits.

The DES was implemented successfully as a computer program that is shown in AppendixA.

.

2.6 KNAPSACK (Public/Private Key Encryption) [3]

The Merkle-Hellman knapsack cryptosystem is based on a problem that is known to be NP-complete. This seems to make it an ideal candidate for a secure public key cryptosystem.

The knapsack problem can be stated as follows. Given a set of n weights

$$W_0, W_1, \ldots, W_{n-1}$$

and sum S, and $a_0, a_1, \ldots, a_{n-1}$, where each a_i *is either* 0 *or* 1, so that

$$S = a_0W_0 + a_1W_1 + \cdots + a_{n-1}W_{n-1}$$

provided this is possible. For example, suppose the weights are

85, 13, 9, 7, 47, 27, 99, 86

and the desired sum is $S = 172$. Then a solution to the problem exists and is given by:

$$a = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (11001100)$$

since 85 + 13 + 47 + 27 = 172. Although the general knapsack problem is known to be NP-complete, there is a special case that can be solved in linear time.

A *superincreasing knapsack*, is similar to the general knapsack except that when the weights are arranged from least to greatest, each weight is greater than sum of all previous weights. For example,

3, 6, 11, 25, 46, 95, 200, 411

is a superincreasing knapsack. Solving a superincreasing knapsack problem is easy.

Suppose we are given the set of weights in the equation above and the sum $S = 309$.

To solve this, we simply start with the largest weight and work toward the smallest to recover the a_i in linear time. Since $S < 411$, we have $a_7 = 0$. Then since $S > 200$, we have $a_6 = 1$ and we let $S = S - 200 = 109$. Then since $S > 95$, we have $a_5 = 1$ and we let $S = S - 95 = 14$. Continuing in this manner, we find $a = 10100110$, which we can easily verify solves the problem since 3 + 11 + 95 + 200 = 309.

Next, we'll list the steps in the procedure used to construct a knapsack cryptosystem.

We'll then illustrate each of these steps with a specific example.

1. Generate a superincreasing knapsack.
2. Convert the superincreasing knapsack into a general knapsack.
3. The public key is the general knapsack.
4. The private key is the superincreasing knapsack together with the conversion factors.

Below, we'll see that it's easy to encrypt using the general knapsack and, with the private key, that it's easy to decrypt the ciphertext. Without the private key, it appears that a very difficult problem must be solved to recover the plaintext from the ciphertext.

Now we'll present a specific example. For this example, we'll follow the numbering in the outline above.

1. We'll choose the superincreasing knapsack

$(2, 3, 7, 14, 30, 57, 120, 251)$.

2. To convert the superincreasing knapsack into a general knapsack, we choose a multiplier m and a modulus n so that m and n are relatively prime and n is greater than the sum of all elements in the superincreasing knapsack. For this example, we choose $m = 41$ and $n = 491$. Then the general knapsack is computed from the superincreasing knapsack by modular multiplication:

$2m = 2 \cdot 41 = 82 \bmod 491$

$3m = 3 \cdot 41 = 123 \bmod 491$

$7m = 7 \cdot 41 = 287 \bmod 491$

$14m = 14 \cdot 41 = 83 \bmod 491$

$30m = 30 \cdot 41 = 248 \bmod 491$

$57m = 57 \cdot 41 = 373 \bmod 491$

$120m = 120 \cdot 41 = 10 \bmod 491$

$251m = 251 \cdot 41 = 471 \bmod 491$.

The resulting general knapsack is $(82, 123, 287, 83, 248, 373, 10, 471)$, which appears to be a general (non-superincreasing) knapsack.

3. The public key is the general knapsack

Public key: $(82, 123, 287, 83, 248, 373, 10, 471)$.

4. The private key is the superincreasing knapsack together with the modular inverse of the conversion factor m, that is,

Private key: $(2, 3, 7, 14, 30, 57, 120, 251)$ and

$M^{-1} \bmod n = 41^{-1} \bmod 491 = 12$.

Suppose A's public and private key pair are given in *step* 3 and *step* 4, above.

Then if B wants to encrypt the message M = 150 for A, B first converts 150 to binary, that is, 10010110. Then B uses the 1 bits to select the elements of the general knapsack that are summed to give the ciphertext. In this case, B finds

C = 82 + 83 + 373 + 10 = 548.

To decrypt this ciphertext, A uses his private key to find

$(C * m^{-1})$ mod n = 548 · 12 mod 491 = 193

A then solves the superincreasing knapsack for 193. This is an easy (linear time) problem from which A recovers the message in binary 10010110 or, in decimal, M = 150.

In words, multiplying by m^{-1} transforms the problem into the superincreasing realm, where it's easy to solve for the weights. Proving that the decryption formula works in general is equally straightforward.

Without the private key, it appears that the attacker, T, must find a subset of the elements of the public key that sum to the ciphertext value C. In this example, T must find a subset of (82, 123, 287, 83, 248, 373, 10, 471) that sums precisely to 548.

This appears to be a general knapsack problem, which is known to be a very difficult problem.

The trapdoor in the knapsack occurs when we convert the superincreasing knapsack into the general knapsack using modular arithmetic. The conversion factors are unavailable to an attacker. The one-way feature results from the fact that it is easy to encrypt with the general knapsack, but it's apparently hard to decrypt without the private key.

But with the private key, we can convert the problem into a superincreasing knapsack problem, which is easy to solve.

Unfortunately, this clever knapsack public key cryptosystem is insecure. It was broken (by Shamir) in 1983 using an Apple II computer . The attack relies on a technique known as "lattice reduction".The bottom line is that the "general knapsack" that is derived from the superincreasing knapsack is not really a general knapsack. In fact, it's a very special and highly structured case of the knapsack, and the lattice reduction attack is able to take advantage of this structure to easily solve for the plaintext (with a high probability).

Much research has been done on the knapsack problem since the Merkle-Hellman knapsack was broken. Today, there are knapsack variants that appear to be secure, but people are reluctant to use them since the name "knapsack" is forever tainted.

Typically, we will choose the value of n to be 100 to 200 binary digits long. If n is 200 bits long, the s_i (super increasing knapsack) are usually chosen to be about 2^{200} apart.

That is, there are about 200 terms in the knapsacks, and each term of the simple knapsack is between 200 and 400 binary digits long.

To find inverses [2] we use Fermat's theorem:

In number theory, Fermat's theorem states that for any prime p and any element $a < p$,

$$x = a^{p-2} \bmod p$$

The Knapsack algorithm was implemented successfully as a computer program that is shown in AppendixA.

Chapter Three:The DES Proposed System

3.1 Important Design Concepts

It is easy to design a block cipher. if it is thought of a 64-bit block cipher as a permutation of the 64-bit numbers, it is clear that almost all of those permutations are secure. What is difficult is to design a block cipher that is not only secure, but can also be easily described and simply implemented.

It is easy to design a block cipher if we have sufficient memory for 48-inputs-32-outputs S-boxes. it is hard to design an insecure DES variant if it is iterated for 128 rounds. if the length of the key is 512 bits, it does not matter if there are key-complementation properties.

The difficulty with real-world block cipher design is to design a highly secure block cipher with the smallest possible key, the smallest possible memory, and the fastest possible running time.

3.2 Differential and Linear Cryptoanalysis [4]

The DES was the world standard for a very long time ,which made it a target for new types of attacks ,as the differential cryptoanalysis and linear cryptoanalysis.

3.2.1 Differential Cryptoanalysis

In 1990,Eli Biham and Adi Shamir introduced Differential cryptoanalysis this new method of cryptoanalysis is unknown to the public yet. using this method Biham and Shamir found a chosen-plaintext attack against DES that was more efficient than Brute Force.

Differential cryptoanalysis looks specifically at ciphertext pairs: pairs of ciphertexts whose plaintexts have particular differences. it analyzes the evolution of these differences as the plaintext propagates through the rounds of DES when they are encrypted with the same key.

Simply, choose pairs of plaintexts with a fixed difference. the two plaintexts can be chosen at random ,as long as they satisfy particular difference conditions. the cryptoanalyst does not even have to know their values.(for DES, the term “difference” is defined using

XOR. this can differ for different algorithms.) Then, using the differences in the resulting ciphertexts, assign different probabilities to different keys. as you analyze more and more ciphertext pairs, one key will emerge as the most probable. This is the correct key.

Differential cryptoanalysis works against DES and other similar algorithms with constant S-Boxes. the attack is heavily dependent on the structure of S-Boxes; the ones in DES just happen to be optimized against differential cryptoanalysis.

DES’s resistance can be improved by increasing the number of rounds .Chosen-plaintext differential cryptoanalysis DES with 17 or 18 rounds takes about the same time as a brute-force search. at 19 rounds or more, differential cryptoanalysis becomes impossible because it requires 2^{64} chosen plaintexts:DES has a 64-bit block size, so it has only 2^{64} possible plaintext blocks. in general, you can prove that an algorithm is resistant to differential cryptoanalysis by showing that the amount of plaintext required to mount such an attack is greater than the amount of plaintext possible.

3.2.2 Linear Cryptoanalysis

Linear cryptoanalysis is another type of cryptoanalytic attack, invented by Misuru Matsui. This attack uses linear approximation to describe the action of a block cipher (in our case, DES) This means that if some of the plaintext bits are XORed together, some ciphertext bits are XORed together, and then the result is XORed, the single bit result is the XOR of some of the key bits.

Against full 16-round DES ,this attack requires 2^{43} known plaintexts. a software implementation of this attack recovered a DES key in 50 days using 12 HP9735 workstations. That is the most effective attack against DES at this time.

Linear cryptoanalysis is heavily dependent on the structure of the S-Boxes and the S-Boxes in DES are not optimized against this attack. in fact, the ordering of the S-Boxes chosen for DES lies among the 9 percent to 16 percent that offer the least protection against linear cryptoanalysis. According to Don Coppersmith, one of the designers of DES, resistance to linear cryptoanalysis “was not part of the design criteria of DES.” Either they didn’t know about linear cryptoanalysis or they knew about something else more powerful whose resistance criteria took precedence.

3.3 The DES Proposed System

The modification to the DES algorithm will be on the key-processing algorithm, so that the arrangement of the S-Boxes will be dependent on the sub-keys, this will create a different way for choosing S-Boxes for each round.

3.3.1 Steps of the Algorithm

The key-processing algorithm has an 80-bit key length, which becomes 70 bits after removing the parity bits,7-bit left shift each round, a part to indicate the arrangement of S-Boxes for each round and a stage of S-boxes inside the key-processing algorithm itself, in addition to linear permutations to provide more diffusion.

The proposed key generation algorithm is described in the following steps:

1.The 80 bits enters initial permutations to cancel 10 parity bits,The permutation is shown below.

Table 3-1 The proposed system main permutations

57	43	47	25	62	2	61	66	45	4
34	70	30	77	22	5	76	20	42	55
75	38	41	50	52	79	23	67	51	21
73	19	53	17	9	44	13	74	10	27
29	1	31	35	14	65	33	11	28	6
3	58	12	78	7	37	18	15	63	26
54	39	71	36	59	60	68	49	69	46

2.The 70 bits are seperated into three parts:

A.48 bits enters the S-boxes to generate 32 bit.

B.16 bits enters a permutation box to produce 16 bits permuted output, this permutation is shown below:

C.6 bits, first two bits are XORed, second two bits are XORed to produce four bits.

Table 3-2 The proposed system secondary permutations

12	9	10	5	1	13	16	11
14	4	6	3	8	2	7	15

3.The leftmost 16 bits of the 32 bits output (step 2,A) are swapped with the permuted 16 bits (step 2,B) and all the output of (step 2) are combined to produce 48-bit block to be sent to the main algorithm as K1.

4.The 4 bit output from (step 2,c) is used twice after adding (1) to the two least significant bits and discarding the carry, first the 4 bits are sent to the main algorithm to control the arrangement of the S-Boxes. The first bit determines whether to switch the boxes 2 and 3, The second bit determines whether to switch the boxes 1 and 7, The third bit determines whether to switch the boxes 4 and 6, The fourth bit determines whether to switch the boxes 5 and 8,Then the four bits are recombined with the 48 bits to prepare for the next subkey of the next round.

5.for the next round the 52 bits are shifted 7-bits to the left ,the 48 bits are sent to the main algorithm, the left four bits are processed like the (step 2,c),and so on for 16 rounds.
the proposed system was implemented as a computer program that can be seen in AppendixB.

3.3.2 System Description

The proposed system has the following description:
1.The 80 bits instead of 64 bits are aimed to resist the brute-force attack. this increases the required iterations by a factor of 2^{14} .that is when using the most basic attack of trying every possible key the needed iterations will be increased from $2^{56}=7.2*10^{16}$ to $2^{70}=1.18*10^{21}$.

2.The S-Boxes inside the key generation algorithm are aimed to reduce linearity. this is to resist the linear cryptoanalysis providing a non-linear operation. the non-linear operation was chosen to be the same S-Boxes of the main algorithm to reduce the needed components (in hardware) or disk space(in software).it is done only once to increase the speed of generation of sub-keys, and it is convenient for key generation.

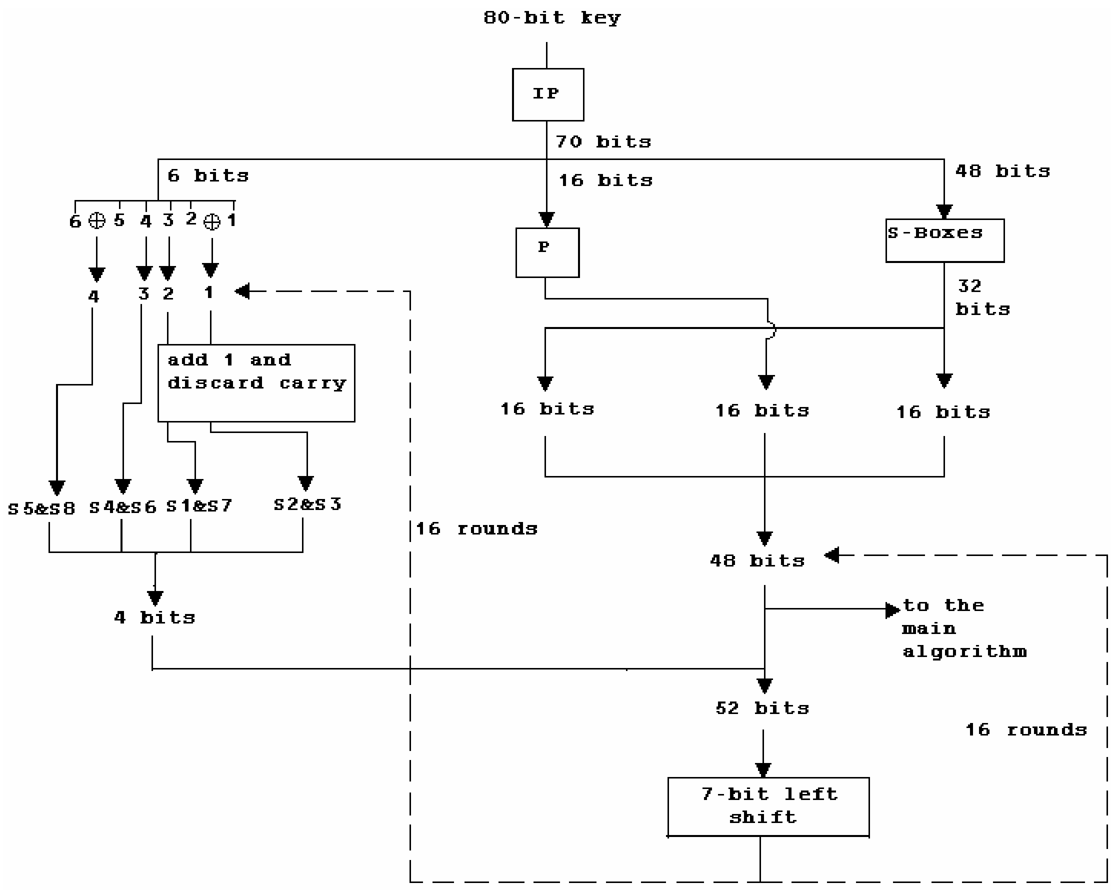


Figure 3-1 The proposed key-generation algorithm

3.the permutation inside the key generation algorithm is to provide more diffusion. diffusion dissipates the redundancy of the plaintext by spreading it out over the ciphertext. a cryptoanalyst looking for those redundancies will have a harder time finding them.

4.controlling the S-Boxes of the main algorithm by the key is a very important feature that is aimed to resist differential cryptoanalysis. the key dependent swapping of the S-Boxes in each iteration makes the differential cryptoanalysis inconvenient to apply because it will require more time than the brute-force attack.

5.the addition of 1 and neglecting the carry provides two benefits; first, there will be no weak ,semi-weak, or possibly-weak keys. second, it will cancel the complement keys property, i.e., when $E_K(M)=C,E_{k'}(M')<>C'$.

6.the 7-bit shift left will provide at least,15 different sub-keys for each key. any number less than 7 will provide more similar sub-keys. it is clear that the number of shifts should not be a factor of 52,because it will cause alike patterns.

3.4 Discussions

The study of statistical properties of the original and proposed systems will show a clearer view of the characteristics of both, and will show which is stronger. such a study needs more time and resources than they are available for this project.

One simple and clear calculation is the operation of effective-key length calculation. for the original DES it can be calculated by:
48 bits, that gives 2^{48} possible keys and for the proposed system:
48 bits+4 bits controlling S-Boxes,that give 2^{52} possible keys.

This shows that the proposed system is more secured than the original DES with a factor of 2^4. this means that if all the other features of the proposed system are put aside, it will take 16 times the time required to break the DES.

For example, the 50 days needed to break the DES with linear cryptoanalysis will become 800 days for the proposed system. neglecting that the proposed system is aimed against cryptoanalysis.

Chapter Four:Conclusion and Future Work

4.1 Conclusion

From the study of the previous chapters we see that all the classical methods of cryptography can not be used in the field in present time because of their simplicity, but they can be used as building blocks for the modern methods of cryptography.

The DES has recently shown noticeable weakness towards the new techniques of cryptoanalysis such as linear cryptoanalysis. so a need for a new stronger algorithm became more important.

The proposed system has some advantages over the old DES ,such as the longer key ,more resistivity toward linear and differential cryptoanalysis, and the elimination of the weak keys and complement keys property. this does not provide an unbreakable cipher, because all ciphers are breakable, theoretically. but the proposed system will provide more security. a further mathematical analysis to both systems will provide more information about who is stronger.

Although Knapsack is a public key algorithm which use a high number of bits, it can be cracked easily using a technique called 'Lattice reduction'.

Much research has been done on the knapsack problem since the Merkle-Hellman knapsack was broken. Today, there are knapsack variants that appear to be secure, but people are reluctant to use them since the name “knapsack” is forever tainted.

4.2 Suggestions for Future Studies

In the field of cryptography, there are many new directions. The most recent is quantum cryptography. with it ,it is possible to create a communication channel where it is impossible to eavesdrop without disturbing the transmission.

The new cryptographic systems, such as Rijndael, use a longer key that is between 128 and 256 bits. this, probably ,will hold long enough toward any of the attacks available today.