B.E. [COMPUTER SCIENCE AND ENGINEERING]

VI SEMESTER

08PE604 - Professional Elective – IV

[NETWORK SECURITY]

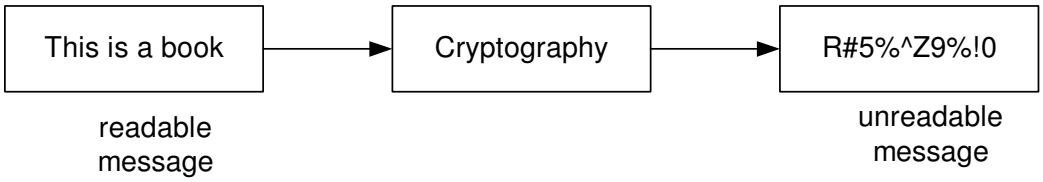
UNIT – I

BASIC TERMINOLOGY

1) Cryptography
2) Plain text
3) Cipher text
4) Encryption
5) Decryption

Cryptography

It is the art and science of achieving security by encoding messages to make them non-readable.



Plain text

Plain text or clear text signifies a message that can be understood by the sender, the recipient and also by anyone else who gets an access to that message.

Cipher text

When a plain text message is codified using any suitable scheme, the resulting message is called as cipher text.

Encryption

The process of encoding plain text messages into cipher text messages is called as encryption.

Decryption

The reverse process of transforming cipher text messages back to plain text messages is called as decryption.

1.1 SECURITY PROBLEM IN COMPUTER NETWORKS

Network Security measures are needed to protect data during their transmission. The security of computer based information systems is concerned with methods of providing cost effective and operationally effective protection of information system from undesirable future events.

There are three aspects of Information Security. They are

1) Security Attack
2) Security Mechanism
3) Security Service

Security Attack

Any action that compromises the security of information owned by an organization.

Security Mechanism

A mechanism that is designed to detect, prevent or recover from a security attack.

Security Service

A Service that enhances a security of the data processing systems and the information transfers of an organization.

Security Attack

4 kinds of security attack.

1. Interruption
2. Interception
3. Modification
4. Fabrication

Interruption

An assert of the system is destroyed or unavailable or unusable. This is an attack on availability.

For example: Destruction of a hardware device, cutting of a common line.

Interception

An unauthorized party gains access to an assert. This is an attack on confidentiality. The unauthorized party would be a person, a programme or a computer.

For eg: Copying of files or programs.

Modification

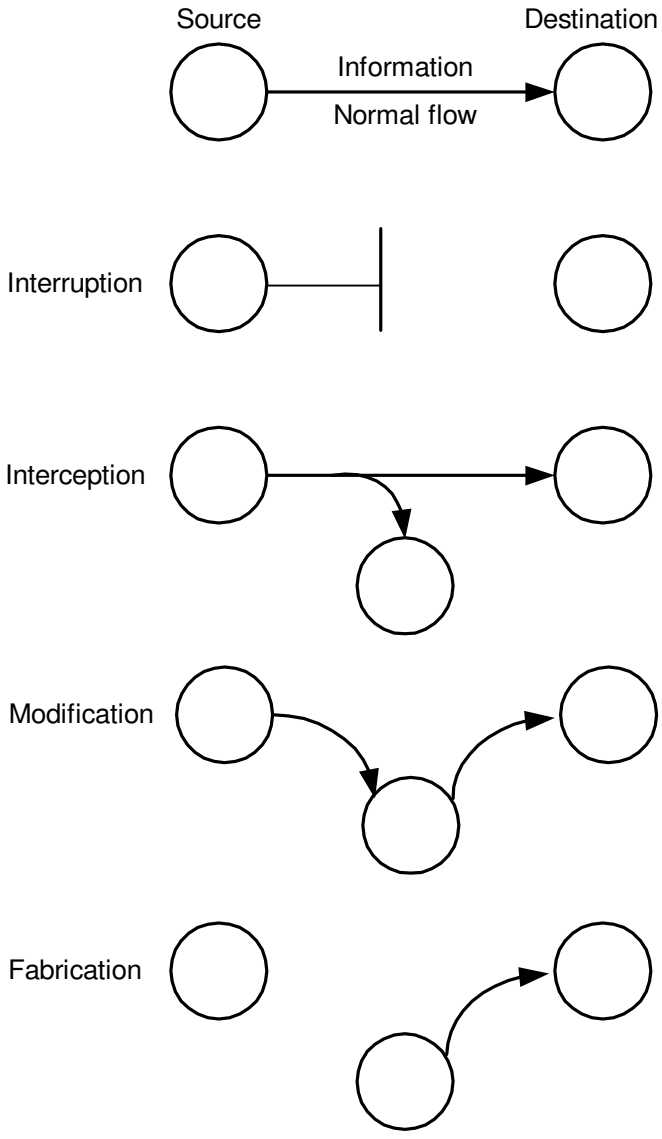
An unauthorized party not only accesses but tampers with an assert. This is an attack on integrity.

For eg: Changing values in a data file, altering a programme so that it performs differently and modifying the content of messages being transmitted in a network.

Fabrication

An unauthorized party inserts counterfeit objects into the system. This is an attack on authenticity.

For eg: Addition of records to an existing database.



Security Services

1) Confidentiality
2) Integrity
3) Availability

Confidentiality: (Secrecy or Privacy)

Confidentiality ensures that computer related asserts are accessed only by authorized parties.

Confidentiality is the protection of transmitted data from passive attacks with respect to the release of message contents, several levels of protection can be identified.

The service protects all user data transmitted between two users over a period of time.

For eg: Traffic flow analysis.

Integrity

Integrity means that asserts can be modified only by authorized parties.

Integrity is active attack.

Integrity means different things in different context.

Modification includes writing, changing, deleting and creating.

Characteristics

1) Concise
2) Accurate
3) Modified only by authorized parties
4) Modified only by authorized process.
5) Modified only by acceptable way.
6) Consistent
7) Easy to understand.

Availability

Availability means that asserts are accessible to authorized parties at appropriate times.

Sometimes called Denial of Service.

Availability applies both data and services.

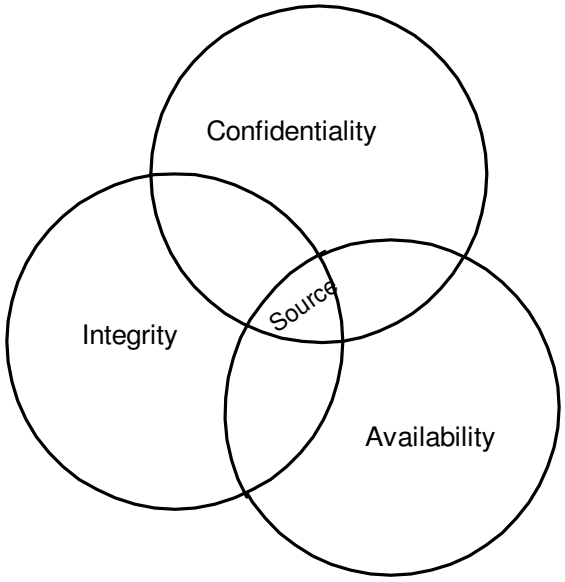
For eg: Service is available if

i) it is present in a usable form

ii) It is making clear progress

iii) The service is completed in an acceptable period of time.

Relationship between Confidentiality and Integrity and Availability



Threats, Vulnerabilities and Controls

A computer based system has 3 separate but valuable components.

1. Hardware

2. Software

3. Data

Vulnerability

Vulnerability is a weakness in the security system.

For eg: Procedures, design or implementation.

Threat

Threat is a set of circumstances that has the potential to cause loss or harm.

Control

Control is an action, device, procedure or technique that removes or reduces vulnerability.

1.2 BREACHES OF SECURITY

Abuse of computer based information system such as theft of resources within the computer installation, illegal access to or use of data, disruption of computing resources or theft of proprietary software are referred to as breaches of security. They are illustrated in table 1.1.

i) Loss of availability of services (the service is not available at the scheduled time)

ii) Loss of integrity (the system does something not intended or does not do something that is intended; or data values are incorrect)

iii) Loss of confidentiality (data are revealed to unauthorized people)

Availability and integrity apply to all information systems but confidentiality does not. Security of information systems seeks to eliminate or reduce these risks of deterioration of services and is a vast subject dealing with hardware, system software, application software, people and organizations. This is not to imply that security has only a technical dimension. It is more than a technical subject and includes the psychological and sociological behaviour of people (Parker, 1981) In fact, the behaviour of people is a major and central factor in security.

Table 1.1 Threats, Breaches and Countermeasures

Threat	Loss of	Breach of security	Countermeasures
Fire	Availability of computing services	Destruction of data and hardware	Fire and smoke detectors
Insertion of forged input data	Integrity	Financial fraud and corruption of data file	Sound clerical procedures and good administrative practices
Unauthorised perusal of computer reports	Confidentiality	Unauthorised access to sensitive data	Good administrative practices
Failure of computer terminal	Availability	Disruption of computing services	A reserve terminal
Noise affecting	Integrity and	Loss of data	Message

communications and so data transmission	availability		sequences numbering
Theft of data file by computer personnel	Availability	Removal and hence non-availability of company data	Vetting of personnel and good operational procedures.

1.3 CLASSICAL ENCRYPTION TECHNIQUES

Encryption

The process of converting plaintext to cipher text is known as enciphering or encryption.

There are two basic encryption techniques: They are

1) Substitution
2) Transposition

Substitution

It is the process in which each element in the plain text (bit, letter, group of bits or letters) is mapped into another element.

Transposition

It is the process in which elements in the plain text are rearranged.

SUBSTITUTION TECHNIQUES

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

Some of the substitution techniques are:

1) Caesar cipher
2) Mono-alphabetic cipher
3) Playfair cipher
4) Hill cipher
5) Poly-alphabetic cipher

Caesar cipher

The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet.

Example

Plain text:	M	E	E	T	M	E
Cipher text:	P	H	H	W	P	H

The algorithm can be explained as

$$C = E(P) = (P+3) \bmod 26$$

$$P = D(C) = (C-3) \bmod 26$$

Where,

C= Cipher text, P= Plain text

If we assign numerical equivalent to each letter of the alphabet, then

a → 1

b → 2

c → 3

.

.

.

z → 26

We can solve the above example by using numerical values:

Plain text:	M	E	E	T	M	E
	↓	↓	↓	↓	↓	↓
	13	5	5	20	13	5

For finding Cipher text (Take P = M)

$C = (P+3) \bmod 26$

$= (13+3) \bmod 26$

$= 16 \bmod 26$

$C = 16$

Solving in the above manner, we get

Cipher text:	16	8	8	23	16	8
	↓	↓	↓	↓	↓	↓
	P	H	H	W	P	H

For Decryption,

$$P = (C-3) \bmod 26$$

$$C = 16 \quad = (16-3) \bmod 26$$

$$= 13$$

Now, we get the plain text as

P:	13	5	5	20		13	5
	↓	↓	↓	↓		↓	↓
	M	E	E	T		M	E

Example:

Plain text: meet me at nine

A sample key might be:

Plaintext letter:	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
Keyword	y	n	l	k	x	b	s	h	m	i	w	d	p	j	r	o	q	v	f	e	a	u	g	t	z	c

To encrypt the message “meet me at nine”, start by taking the first letter of the message, ‘m’, and look up the corresponding keyword ‘p’. Repeat by looking up the next plaintext letter ‘e’, and noting it becomes ‘x’. Continue this process for the rest of the message. Typically spaces, numbers, and punctuation are left alone. In this case “meet me at nine”. Would become “pxxe px ye jmjx”.

Plain text:	meet	me	at	nine
Cipher text:	pxxe	px	ye	jmjx

Decryption is similar. Start with the first cipher text letter ‘p’, and look at the table to find the corresponding plaintext letter ‘m’. Continue with the next letter ‘x’, and find it maps to ‘e’.

Cipher text:	pxxe	px	ye	jmjx
Plain text:	meet	me	at	nine

Playfair cipher

The playfair cipher is a cryptographic technique that is used for the manual encryption of data.

Playfair is quite fast to use and does not demand any equipment to use.

The playfair cipher is based on the use of a 5×5 matrix of letters constructed using a keyword.

Encryption is done based on the following rules:

1) Repeating plaintext letters that would fall in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
2) Plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right.
3) Plaintext letters that fall in the same column of the matrix are each replaced by the letter beneath.
4) Each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

Example

Pick a keyword that does not contain any letter more than once.

Keyword: KEYWORD

Now write the letters of that word in the first squares of a five by five matrix:

K	E	Y	W	O
R	D			

Then finish filling up the remaining squares of the matrix with the remaining letters of the alphabet, in an alphabetical order. Since there are 26 letters and only 25 squares, we assign I and J to the same square.

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

To encipher a message, divide it into pairs of letters. Pay no attention to punctuation or to spaces between words. For example, the sentence “Why, don’t you?” becomes

WH YD ON TY OU

Now, find each pair of letters in the matrix you made earlier. Most pairs of letters will form two corners of a smaller square or rectangle within the matrix. For example, the first pair of letters (WH) are at two corners of a two-by-three rectangle also containing Y, A, B, and IJ. The enciphering of the pair WH is the pair at the two other corners of this rectangle, namely YI. (I could also have chosen, in this case). It’s important to be consistent about the order of the new pair: the one that comes first is the one on the same row as the first of the original pair. In this case, Y is on the same row as W. We can continue to translate the remaining pairs of letters in the same way, ending up with

YI EA ES VK EZ

Keyword	:	KEYWORD
Plaintext	:	WHY DON’T YOU?
Cipher text	:	YI EA ES VK EZ

Decryption is done similar. The reverse process of encryption is decryption.

Keyword	:	KEYWORD
Cipher text	:	YI EA ES VK EZ
Plain text	:	WHY DON’T YOU?

Hill cipher

Hill cipher is a polygraphic substitution cipher based on linear algebra. Invented by Lester S. Hill in 1929.

In general terms, the Hill cipher can be expressed as:

$$C = E_K(P) \quad KP \bmod 26$$

$$P = D_K(C) \quad K^{-1} C \bmod 26 \quad K^{-1} KP = P$$

Where

C = Cipher text, P = Plaintext, K = Keyword.

For m=3, the system can be described as:

$$C_1 = (K_{11}P_1 + K_{12}P_2 + K_{13}P_3) \mod 26$$
$$C_2 = (K_{21}P_1 + K_{22}P_2 + K_{23}P_3) \mod 26$$
$$C_3 = (K_{31}P_1 + K_{32}P_2 + K_{33}P_3) \mod 26$$

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} \mod 26$$

Example

Keyword : FXAMPJTQO

Message : ABS

$$K = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} P = \begin{pmatrix} 1 \\ 2 \\ 19 \end{pmatrix}$$

Polyalphabetic ciphers

The polyalphabetic substitution cipher have the following features:

1) A set of related mono-alphabetic substitution rules is used.
2) A key determines which particular rule is chosen for a given transformation.

A matrix known as Vigenere tableau is constructed. The process of encryption is simple. Given a key letter x and a plaintext letter y, the cipher text letter is at the intersection of the row labeled x and the column labeled y.

To encrypt a message, a key is needed that is as long as the message.

Example

Keyword	:	D	E	C	E	P	T	I	V	E
Plaintext	:	W	E	A	R	E	H	E	R	E
Cipher text	:	Z	I	C	V	T	W	M	M	I

	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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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

Decryption is equally simple. The key letter identifies the row. The position of the cipher text letter in that row determines the column and the plain text letter is at the top of the column.

Keyword	:	D	E	C	E	P	T	I	V	E
Cipher text	:	Z	I	C	V	T	W	M	M	I
Plaintext	:	W	E	A	R	E	H	E	R	E

TRANSPOSITION CIPHER

In a transposition cipher the letters of the plaintext are shifted about to form the cryptogram.

One of the easiest ways to achieve transposition is the Single Columnar Transposition Cipher. To use it, one needs a keyword or phrase, whose letters are numbered according to their presence in the alphabet.

Example:

The keyword *Ozymandias* is numbered in the following way:

O	Z	Y	M	A	N	D	I	A	S
7	10	9	5	1	6	3	4	2	8

That is, the first occurance of the letter *A* is numbered **1**, the second **2**. There are no *B:s* or *C:s* so the next letter to be numbered are the *D* followed by *I*, and so on.

Next the plaintext is written in rows under the numbered keyword, one letter under each letter of the keyword. Let's say that the plaintext to be encrypted is *Company has reached primary goal*. It will look like this:

O	Z	Y	M	A	N	D	I	A	S
7	10	9	5	1	6	3	4	2	8
C	o	m	p	a	n	y	h	a	s
R	e	a	c	h	e	d	p	r	i
M	a	r	y	g	o	a	l		

Now the letters of the plaintext are copied down by reading them off columnwise in the order stated by the enumeration of the keyword. The result is the finished cryptogram, which - of course - are put into groups of five letters, like this:

AHGAR YDAHP LPCYN EOCRM SIMAR OEA

Plain text: *Company has reached primary goal*

Keyword : O Z Y M A N D I A S

Cipher text : AHGAR YDAHP LPCYN EOCRM SIMAR OEA

To decrypt a received message enciphered by this method, one first must calculate the number of letters present in the cryptogram. This is done to see how many letters there originally were in the last row. As can be seen above, the two last columns - the ones numbered **2** and **8** - only contains two letters and this is important. Now the cryptogram above contains 28 letters and as a legitimate user of the crypto system, one knows that the keyword is ten letters wide. Therefore the last row must consist of eight letters only, the two final positions being empty. Keeping that in mind - or better still, marking the two final position of row three in some way to indicate that they shouldn't be used - one numbers the keyword letters (just as when encrypting) and then start by writing the first three letters of the cryptogram under keyword letter number one, thus:

O	Z	Y	M	A	N	D	I	A	S
7	10	9	5	1	6	3	4	2	8
.	.	.	.	a	.	.	.	.	.
.	.	.	.	h	.	.	.	.	.
.	.	.	.	g	.	.	.	*	*

Next comes column number two. Since the last position in column two is marked by a star and shouldn't be used, one only writes the next **two** letters, instead of three. Continue in the same way by writing the next three letters under keyword letter number three, and so on up to keyword letter eight, it will look like this:

O	Z	Y	M	A	N	D	I	A	S
7	10	9	5	1	6	3	4	2	8
C	.	.	p	a	n	y	h	a	.
R	.	.	c	h	e	d	p	r	.
M	.	.	y	g	o	a	l	*	*

Now column eight follows, and there only two letters should be written as stated above (the position marked by a star being left empty). This leaves six letters of the cryptogram, and these - of course - are written in column nine and ten, and then the cleartext can be read in the normal way, row by row.

Keyword : O Z Y M A N D I A S

Cipher text : AHGAR YDAHP LPCYN EOCRM SIMAR OEA

Plain text: *Company has reached primary goal*

❑

UNIT – II

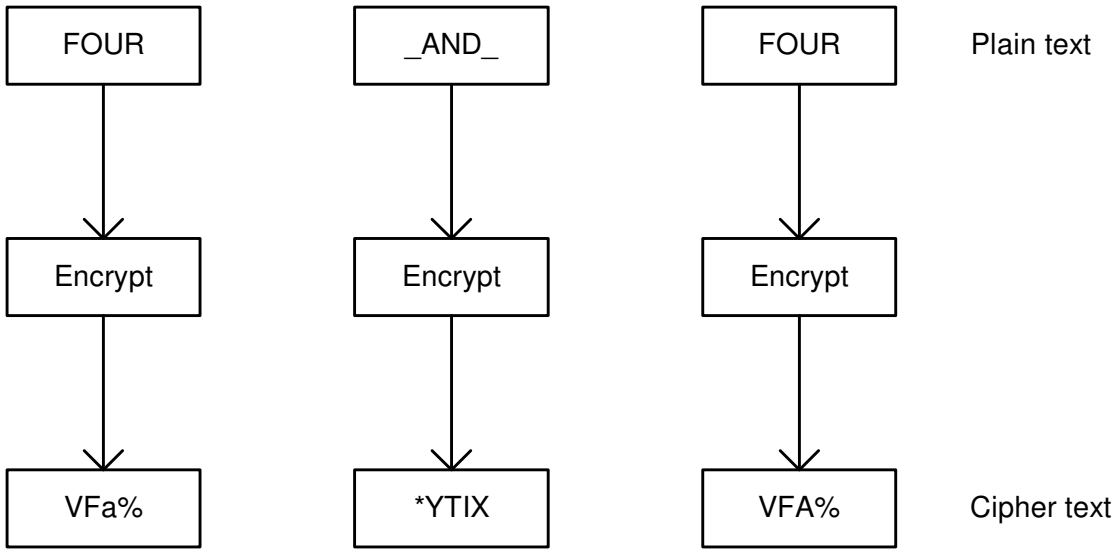
2.1 BLOCK CIPHER

Definition: Block cipher is a technique which involves encryption of one block of test at a time. Decryption also takes one block of encrypted text at a time.

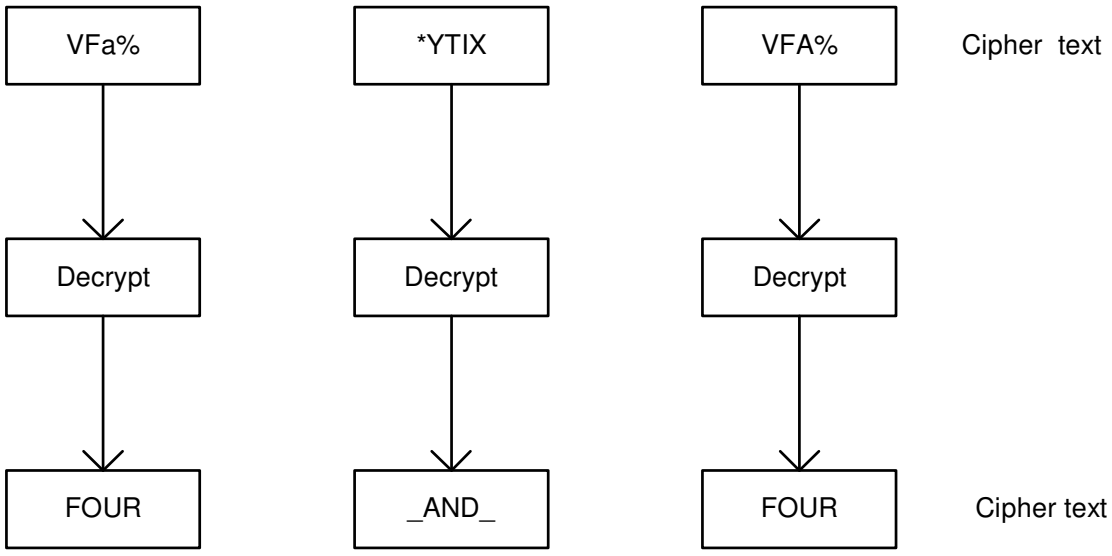
In block ciphers, rather than encrypting one byte at a time, a block of bytes is encrypted at one go.

For example, we have a plain text FOUR_AND_FOUR to be encrypted. Using block cipher, FOUR could be encrypted first followed by _AND_ and finally FOUR. During decryption also, each block would be translated back to the original form.

The actual process of block cipher technique is depicted in the figure 1.



a) Encryption Process at Sender's end



b) Decryption Process at Receiver's end

Fig. 1 Block Ciphers

The blocks used in block cipher generally contain 64 bits or more.

The communication takes placed only in bits. Therefore, the binary equivalent of the ASCII characters FOUR are encrypted by using any algorithm. The resultant encrypted bits are converted back into their ASCII equivalents. Therefore we get any funny symbols such as Vfa%.

For decryption, reverse process can e performed.

Problems in Block cipher

The obvious problem with block cipher is repeating text. For repeating text, the same cipher is generated. It could give due to the cryptanalyst to break the strings, therefore the entire original message can then be revealed.

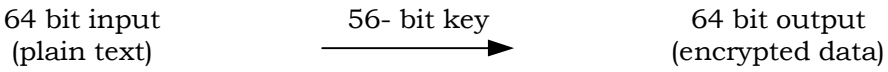
To deal with this problem, bloc ciphers are used in chaining mode.

Block ciphers are used most often in computer-based cryptographic algorithms as compared to stream cipher. It uses the concept of both confusion and diffusion.

2.2 DATA ENCRYPTION STANDARED (DES)

- Most widely used encryption scheme.
- Two inputs are used for this algorithm,
    1) Plaintext (to be encrypted)
    2) key

➢64 bit block of input and 56 bit key.



Steps

There are three main steps used in this algorithm,

- ❖ Plain text to Initial permutation (P – IP)
- ❖ Initial Permutation to Inverse Permutation $(\mathrm{Ip} - \mathrm{Ip}^{-1})$
- ❖ Expansion permutation (E)

1. Plaintext to initial permutation

- ❖ 64 bit plain text passes through an initial permutation
- ❖ Bits are rearranged rearranging bits are called permutted input.

Consider the following 64-bit input (plain text)

1	2	3	4	5	6	7	8
M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

Plaintext (M)

After initial permutation the rearranging bits are as follows,

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{41}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

Initial permutation (Ip)

By equation,

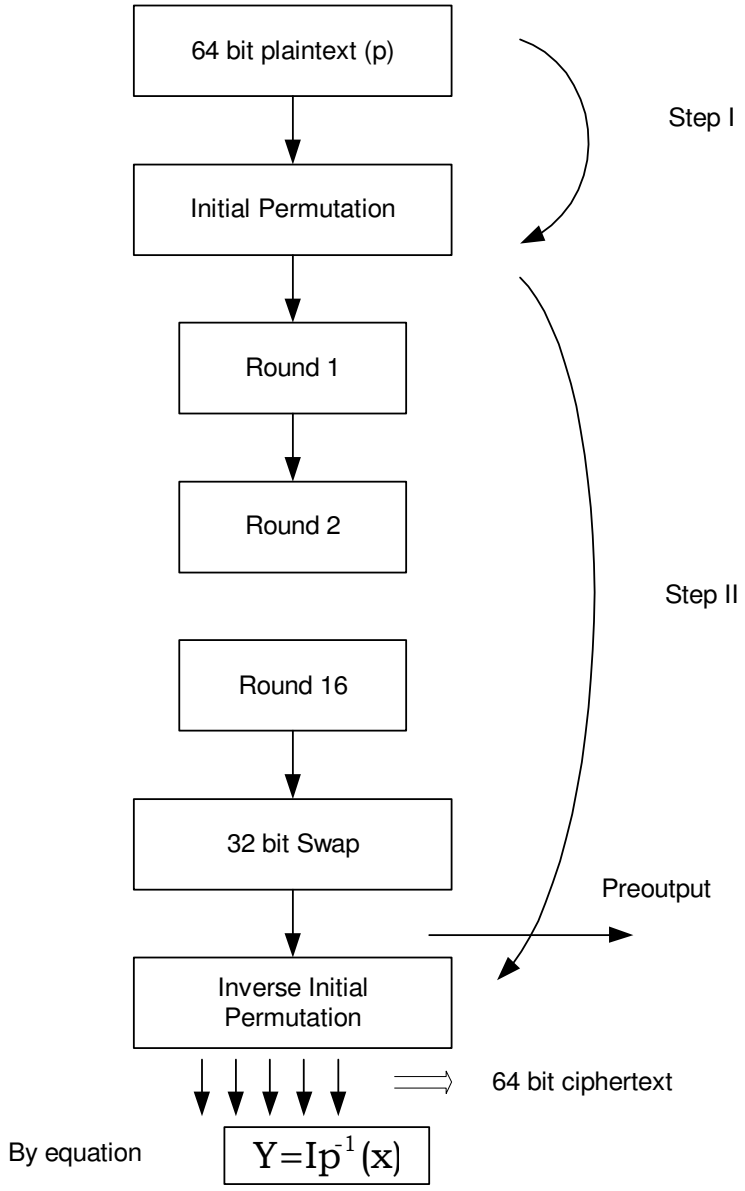
$$X = Ip\ (M)$$

M – Plaintext

X – Initial permutation

2. Initial Permutation to Inverse Permutation

- This steps involves 16 rounds
- Each round involves both permutation and substitution functions.
- The output of 16th round consists of 64 bits which are the function of input plaintext and key.
- i.e., f (p + K)
- Left and right halves of the output are swapped i.e., the o/p of the 16th round is splitted into separate 32 bits and then swapped.
- After swapping the 64 bit block is called pre-output.
- Finally pre-output is passed through Inverse permutation function. (Ip^{-1})
- Ip^{-1} produces 64 bit ciphertext.



Y – inverse permutation

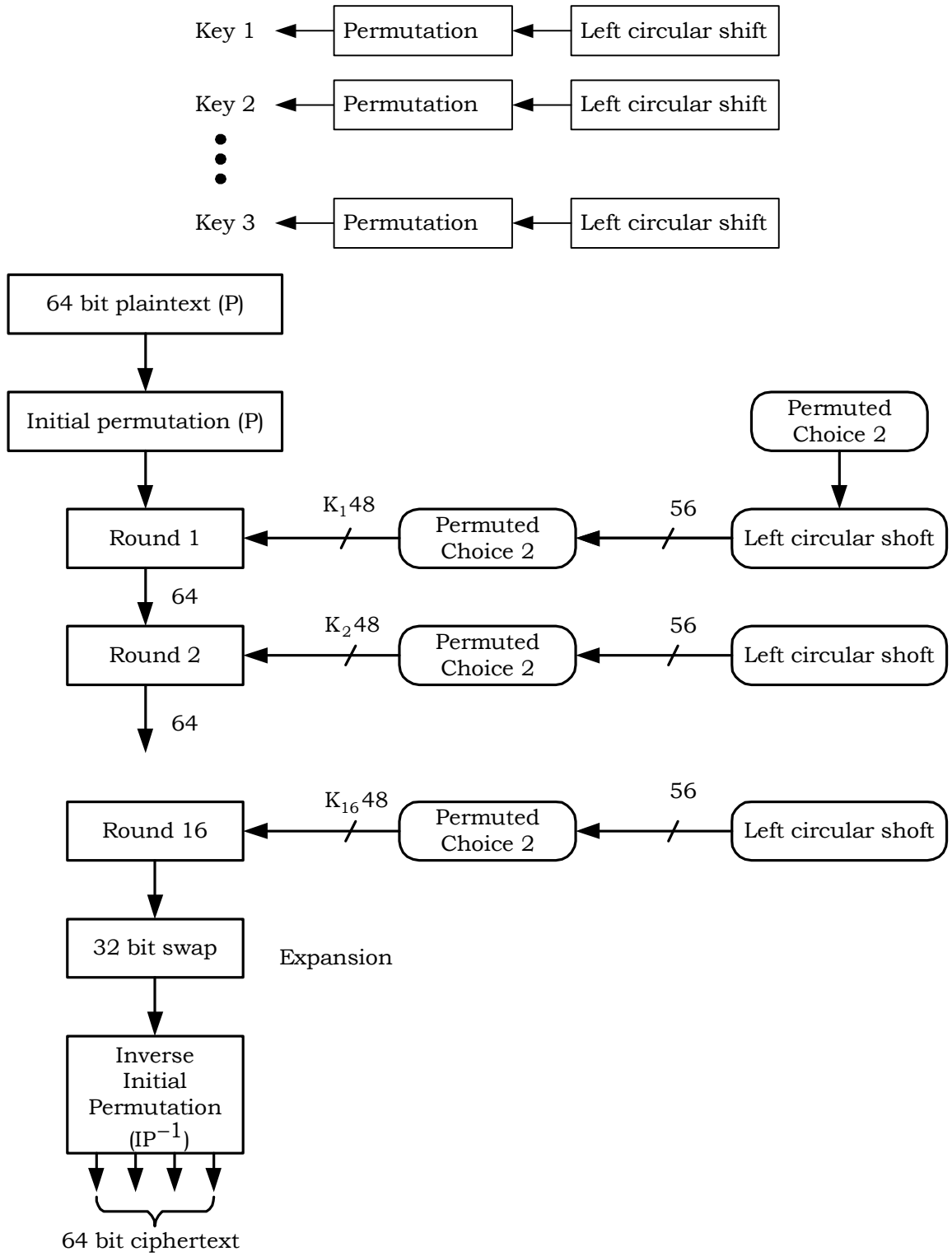
X – initial permutation

Key

- 56 bit key is used here
- At first, key is passed through initial permutation function (IP)
- Then for each round (out of 16), a subkey (k_i) is produced.
- Subkey (k_i) is the combination of,
  - i) Left circular shift
  - ii) Permutation
- Permutation is same for all 16 rounds.

❖ Left circular shift changes the key bits at each round.

(Hence the keybits are changed different subkey is produced at each round).



Details of single Round

- 64 bit block is splitted into 2 32 bit blocks, labeled as L (left) and R (Right)
- (R) input is first expanded to 48 bits by using duplication of 16 of R bits (32+16 = 48)
- The resulting 48 bits are XORed with k_i. This 48 bit result passes through a substitution function that produces a 32 bit output called permutation function (P).

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

Table(a):Expansion permutation

16	17	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

Table (b):Permutation function (P)

Decryption

Uses the same algorithm as encryption that application is reversed.

Role of S-boxes

- It consists of eight S-box

  i/p – 6 bit

  o/p – 4 bit
- The first and last bit of i/p is selected from any one row from one to four rows.
- The middle 4 bits is selected from one of the 16 coloums.

- The decimal is converted to its 4-bit representation to produce the o/p.
- For ex, in S1, i/p → 011001

  i.e.,

  the row is 01 (row 1)

  the column is 1100 (column 12)

  So, the o/p is 1001 (9)

2.3 TRIPLE DES

The meet-in-the-middle attack on Double DES is not practical in cryptography, it is always better to take minimum possible chances.

Therefore Double DES paving the way for Triple DES. Triple DES is DES-three times. It comes in two flavours:

i) One that uses three keys

ii) Other that uses two keys

1. Triple DES with three Keys

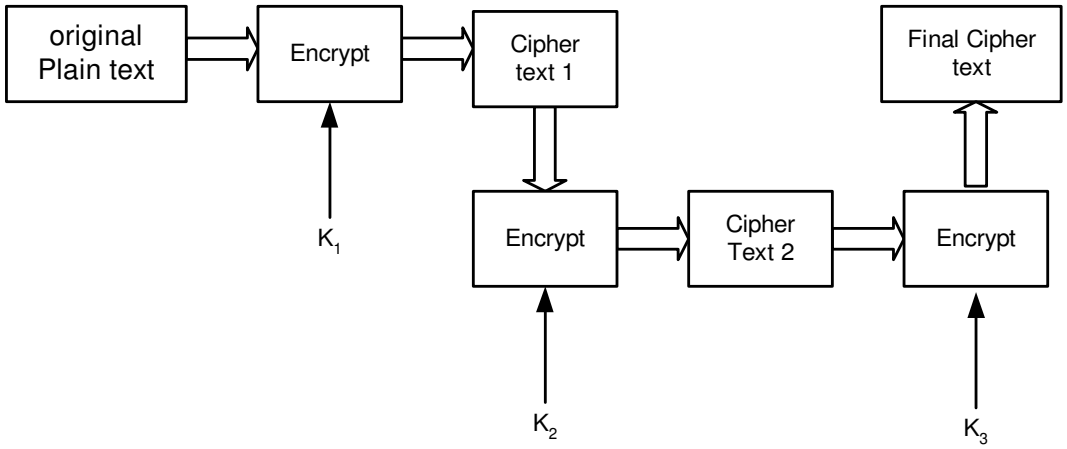


Fig. 1 Triple DES with three Keys

The plaintext 'ρ' is first encrypted with a key K_1, and then encrypted with a second key K_2 and finally with a third key K_3 where K1, K2, and K3 are different from each other. It can be denoted as

$$C = EK_3\,(EK_2\,(EK_2\,(EK_1\,(\rho))))$$

To Decrypt the cipher text C to obtain the plaintext ρ, we need to perform the operation.

$$\rho = DK_3\,(DK_2\,(DK_1\,(C)))$$

It is used in many products like PGP. S/MIME.

Triple DES with two Keys

Since Triple DES with three key has drawback of requiring 168 bits for the key, it is difficult to have in practical situations.

Therefore, Tuchman uses just two keys for Triple DES. The algorithm can works as follows:

Steps

1) Encrypt the plaintext with key K_1

   $EK_1(\rho)$

2) Decrypt the output of Step 1 with Key K_2

   $DK_2\,(EK_1\,(\rho)$

3) Finally, Encrypt the output of step 2 with key K_1

   $EK_1\,(DK_2\,(EK_1\,(\rho)))$

This process can be shown figure 2.

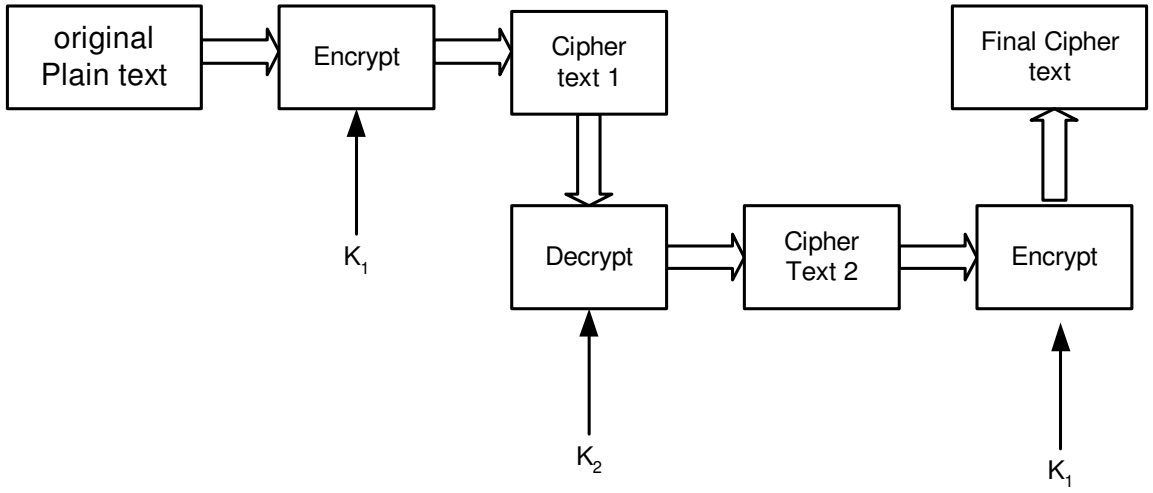


Fig. 2 Triple DES with two keys

To decrypt the cipher text and obtain the original plain text ρ we need to perform the operation,

$$\rho = DK_1\,(EK_2\,(DK_1\,(C)))$$

It is only significance that it allows triple DES to work with two, rather than three keys. This is also called as Encrypt – Decrypt – Encrypt (EDE) mode.

2.4 IDEA

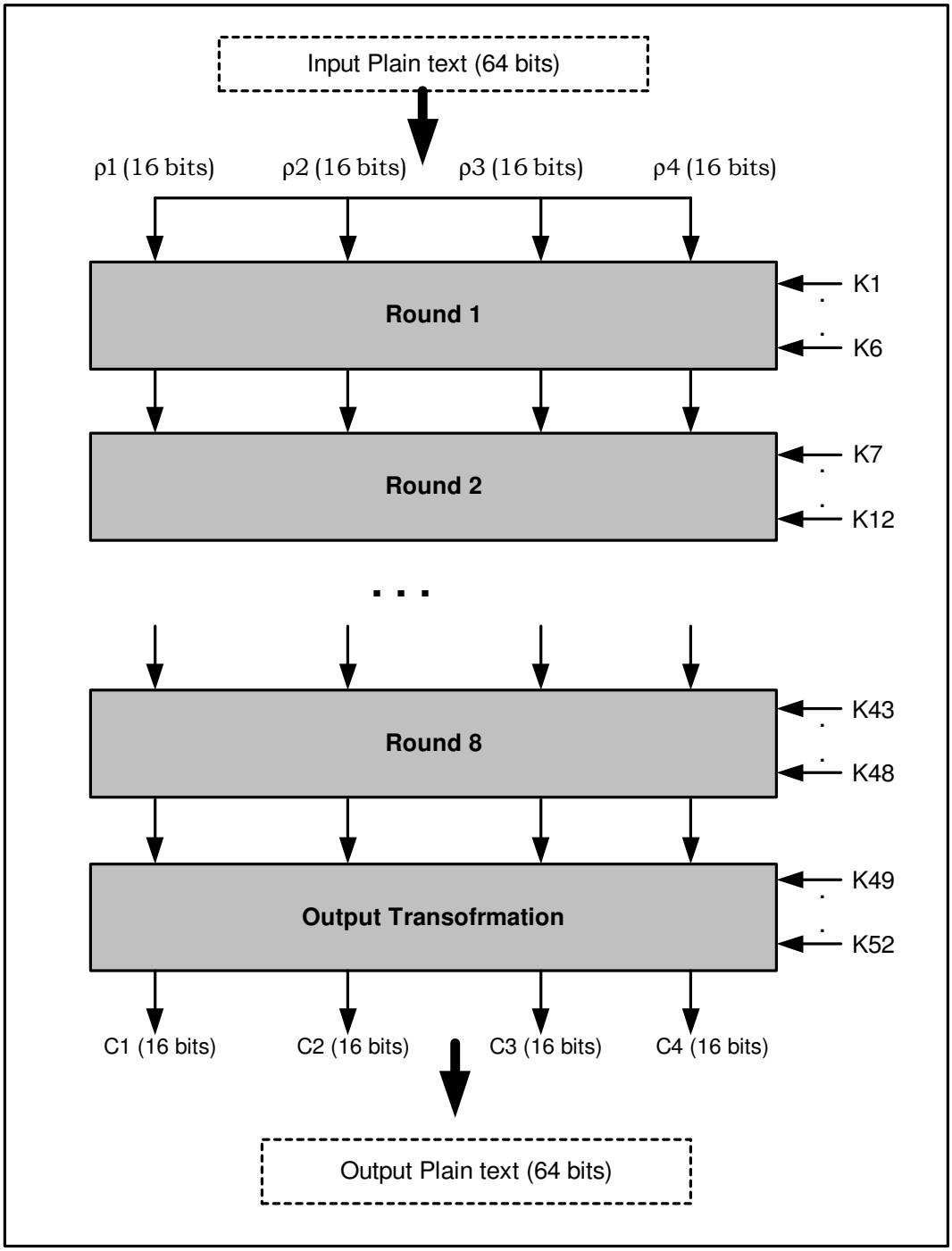
IDEA – International Data Encryption Algorithm.

Principles

- IDEA is one of the strongest cryptographic algorithms, launched in 1990.
- IDEA is a block cipher
- It works on 64 bit plaintext block
- Key consists of 128 bits.
- Like DES, Same algorithm is used for Encryption and decryption.
- PGP is based on IDEA.

Operations

- The 64 bit plain text block is divided into four portions, Say ρ_1 to ρ_4 (each of 16 bits)
- There are eight rounds, for each round six sub-keys are generated from the original key.
- These six sub keys namely K_1 to K_6 and four input blocks are given as input to the Round 1.
- Similarly for second round, keys K_7 to K_{12} and output of round 1 is given as input to the Round 2.
- This will be continued for further rounds.
- The final step consists of an output transformation, which is the four block for cipher text named C_1 to C_4.
- These are combined to form the final 64 bit cipher text block



Details of one round

There are 8 rounds in IDEA. Each round involves a series of operations on the four data blocks using six keys. The operations are as follows

Steps

1) Multiply P1 block with Subkey K1.

2) Add P2 block to Subkey K2.
3) Add P3 block to Subkey K3.
4) Multiply P4 block with Subkey K4.
5) Do the bitwise XOR operation the results of step 1 and step 3.
6) Do the bitwise XOR operation of results of step 2 and step 4.
7) Multiply the results of step 5 with Subkey K5.
8) Add the results of Step 6 and Step 7.
9) Multiply the results of step 8 with Subkey K6.
10) Add the results of step 7 and step 9.
11) Do the XOR operation the results of step 1 and step 9.
12) Do the XOR operation the results of step 3 and step 9.
13) Do the XOR operation the results of step 2 and step 10.
14) Do the XOR operation the results of step 4 and step 10.

Output transformation

The output transformation is a one-time operation. It takes place at the end of 8^{th} round.

Here, we give four sub-blocks say R1 to R4 and four sub-keys say K1 to K4 are applied as input. The process are as follows

Step 1: Multiply R1 with K1.

Step 2: Add R2 and K2.

Step 3: Add R3 and K3.

Step 4: Multiply R4 and K4.

The output of this process is the final 64-bit cipher text, which is the combination of 4 cipher text sub-blocks C1 to C4.

Decryption

The decryption process is exactly the same as the encryption process. The decryption sub-keys are actually inverses of encryption sub-keys.

Strength of IDEA

IDEA uses 128-bit key, which is double than the key size of DES.

To break into IDEA, 2^{128} encryption operations would be required. It takes several million years to break.

2.5 RC5

RC5 is a symmetric Encryption algorithm developed by Ron Rivest

Characteristics

- Suitable for hardware or software
- Fast
- Variable number of rounds
- Variable length key
- High security

Principles

In RC5, the word size (input plain text block) number of rounds and number 8-bit bytes in the key, all can be of variable length. The values are shown in below table.

Parameter	Allowed values
Word size in bits (w)	16,32,64
Number and rounds (r)	0-255
Number of 8-bit byte in key (b)	0-255

Operation

The operation consisting of two steps

i) one initial operation

ii) number of rounds (vary from 0 to 255)

Initial operation

- The 64 bits input plain text is divided into 32 bit blocks A and B.
- Then the first two subkeys S[0] and S[1] are added to A and B respectively. This produces C and D respectively.

Details of one Round operation

The process for the first round are as follows:

Step 1: Do the XOR of C with D to form E.

Step 2: Now E is Circular-left shifted by D positions.

Step 3: Add E to the next sub-key S[2]. The output is assigned to F.

Step 4: Do XOR of D with F to produce G.

Step 5: Now G is circular-left shifted by F positions.

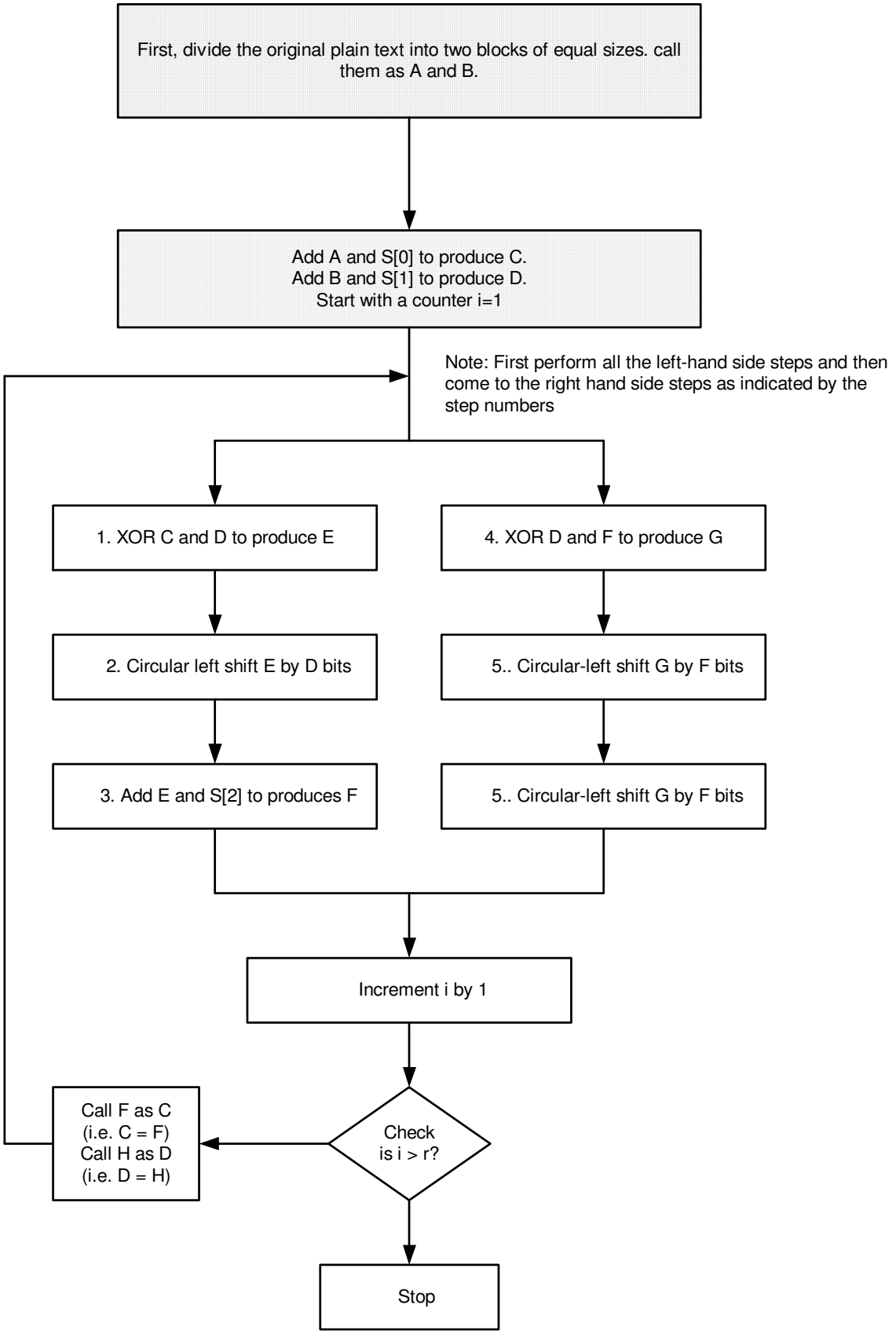
Step 6: Add G to the next sub-key S[3]. The output is assigned to H.

Step 7: Increment counter value (i) to one and check it with 'r' value, if it is less than r, we rename F as C and H as D and return back to step 1.

Repeat the above steps until counter value(i) is greater than r.

Note:

The output of one block is fed back as the input to another block. This made quite complicate to decipher.


First, divide the original plain text into two blocks of equal sizes. call them as A and B.
Add A and S[0] to produce C.
Add B and S[1] to produce D.
Start with a counter i=1
Note: First perform all the left-hand side steps and then come to the right hand side steps as indicated by the step numbers
1. XOR C and D to produce E
2. Circular left shift E by D bits
3. Add E and S[2] to produces F
4. XOR D and F to produce G
5.. Circular-left shift G by F bits
5.. Circular-left shift G by F bits
Increment i by 1
Check is i > r?
Call F as C (i.e. C = F) Call H as D (i.e. D = H)
Stop

RC5 Modes

To enhance the effectiveness of Rc5 in interoperable implementations, RFC 2040 defines four modes of operations, namely

- RC5 block cipher
- RC5 – CBC
- RC5 – CBC – Pad
- RC5 – CTS.

2.6 BLOWFISH

Blowfish is a symmetric block cipher developed by Bruce Schneir

Characteristics

- Fast
- Compact
- Simple
- Secure

Operation

It encrypts 64 bit block with a variable length key. It contain 2-parts.

i) Sub-key Generation

ii) Data Encryption and Decryption

i) SubKey Generation

Blowfish makes use of keys and subkeys. Key size ranges from 1 to 14 words (32 bits to 448 bits).

The keys are stored in a K-array:

$$K_1, K_2 \ldots K_j \quad 1 \le j \le 4.$$

The Subkeys are stored in the P-array. Each contains 32-bit subkeys.

$$P_1, P_2 \ldots P_{18}$$

There are 4 S-boxes, each have 256 32-bit entries.

$$S_{1,0}, \quad S_{1,1}, \quad \ldots \quad S_{1,255}$$

$$S_{2,0}, \quad S_{2,1}, \quad \ldots \quad S_{2,255}$$

$$S_{3,0}, \quad S_{3,1}, \quad \ldots \quad S_{3,255}$$

$S_{4,0}$, $S_{4,1}$, … $S_{4,255}$

Steps in generating the P-array and S-boxes

1) Initialize the P-array and four S-boxes with a fixed string.
2) Perform a bitwise XOR of the P-array and the K-array, reusing words from K-array as needed.

$$P_1 = P_1 \text{ XOR } K_1$$
$$P_2 = P_2 \text{ XOR } K_2$$
$$\dots$$
$$P_{14} = P_{14} \text{ XOR } K_{14}$$
$$P_{15} = P_{15} \text{ XOR } K_1$$
$$\dots$$
$$P_{18} = P_{18} \text{ XOR } K_4$$

(since we have only the key size ranges from 1 to 14)

3) Take a 64-bit block, initialize all 64-bits to zeros. Encrypt the 64-bit block by using the current P – and S – arrays;
   Now, replace P_1 and P_2 with the output and above Encrypted text.
4) Encrypt the output of step 3 using the current P and S arrays.
   Now, replace P_3 and P_4 with the output of Encrypted text.
5) Continue the above steps for all P values to update the all elements in P – and S – arrays.

ii) Encryption and Decryption

X → 64-bit block plaintext.

P arrays and S-boxes are used during Encryption and decryption process.

Algorithm

1) Divide X into two blocks namely XL and XR of equal sizes.
2) Perform the following steps,
   i) Do a bitwise XOR of XL with P value and assigned to XL.
   ii) Do a bitwise XOR of function of XL(F(XL)) with XR and assigned to XR.

The Function F is as follows:

i) Divide 32-bit XL into four 8-bit Sub-blocks named as a, b, c and d.

ii) Compute F[a,b,c,d]= $(([(S1,a)+(S2,b)]XOR\,S3,C)+S4,d$.

iii) Swap XL and XR.

Continue Step 2 until the P value ranges from 1 to 16.

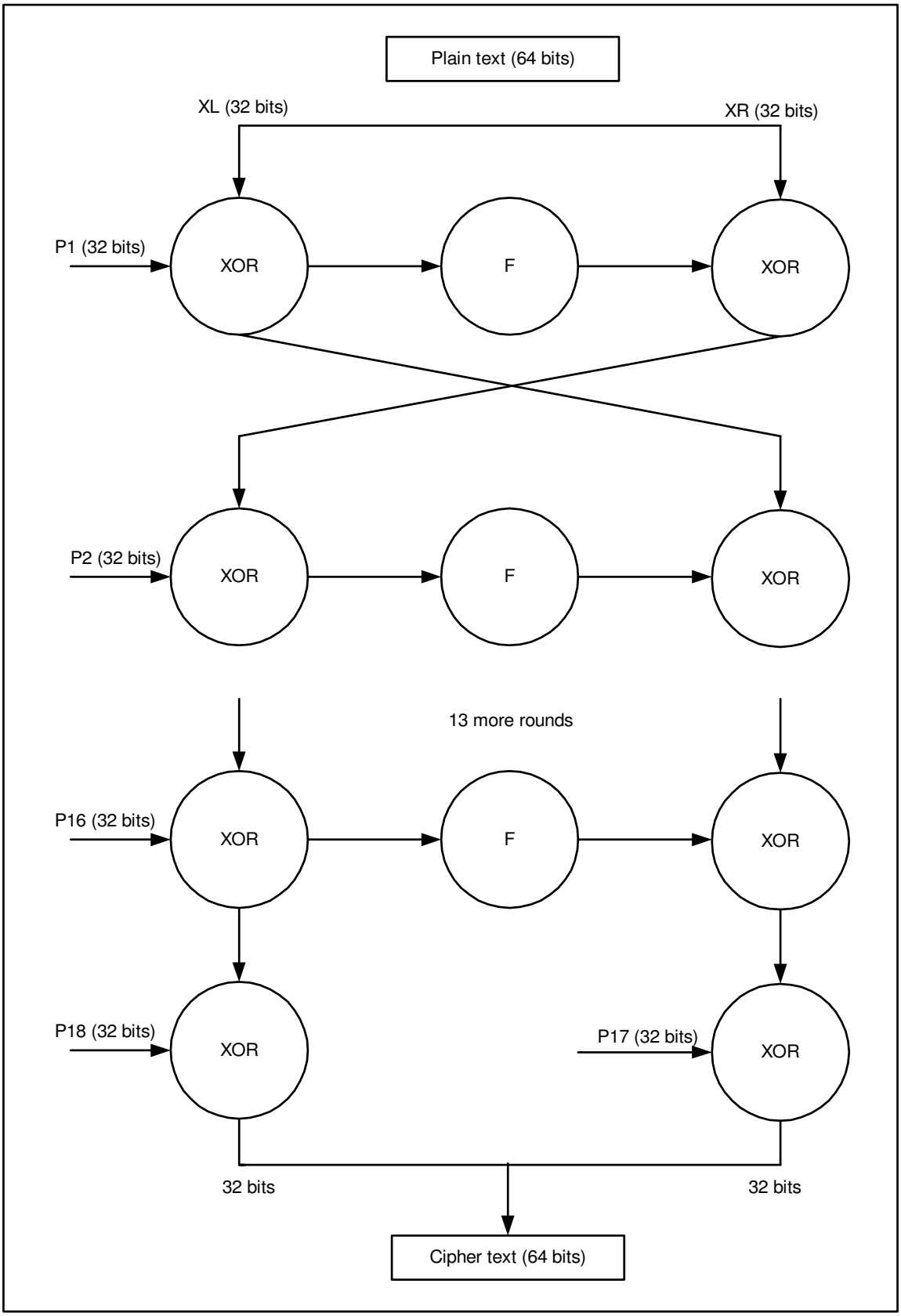
3) Swap XL and XR.

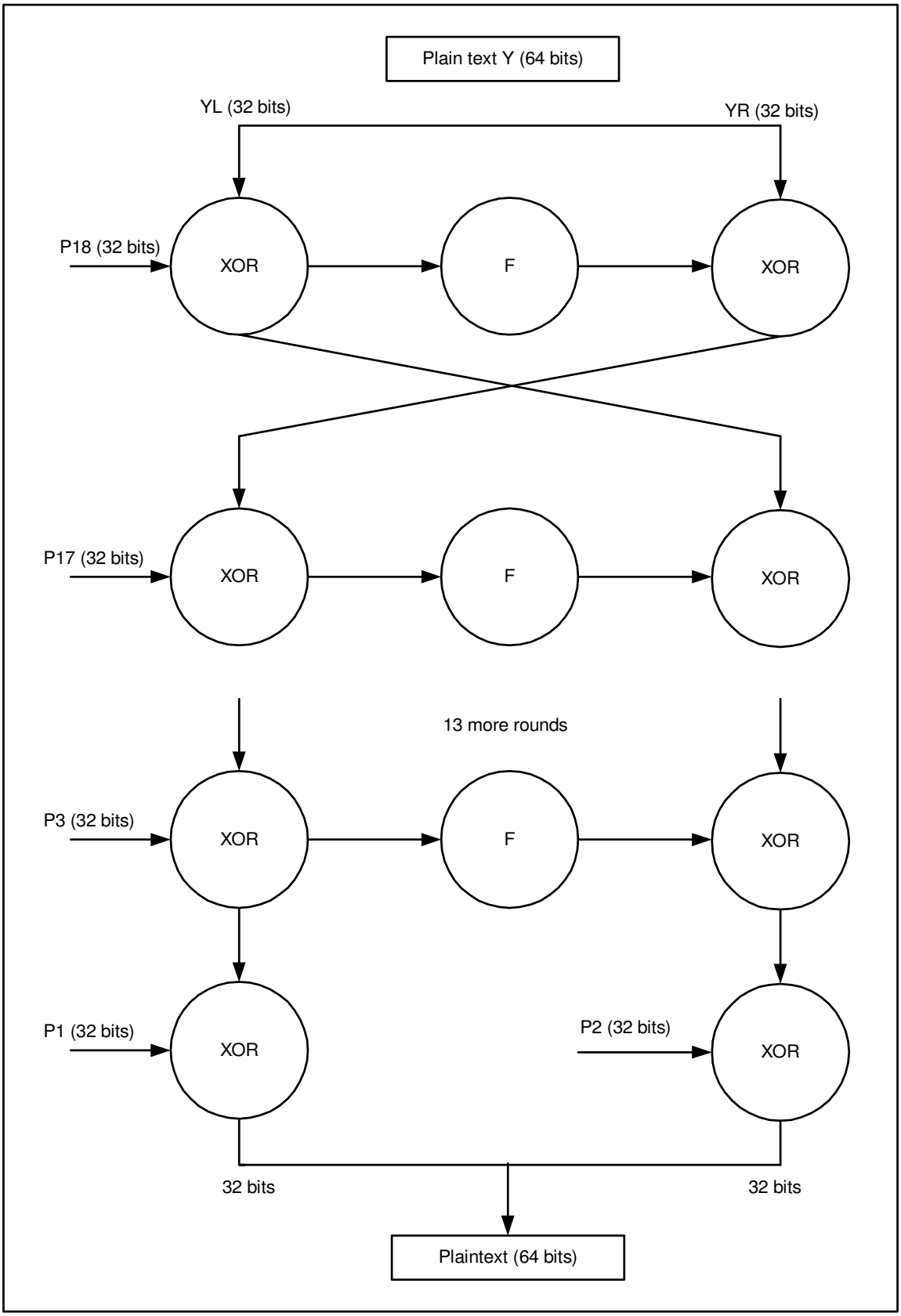
4) Do a bitwise XOR of XL with P_{18} and assigned to XL.

5) Combine XL and XR and stored into X. Now X contains the resultant cipher text.

The decryption is similar to Encryption process, with the reversal of P-array values.

The process of Encryption and decryption is shown in the figure 1(a), 1(b) respectively.


Plain text (64 bits)
XL (32 bits)
XR (32 bits)
P1 (32 bits)
XOR
F
XOR
P2 (32 bits)
XOR
F
XOR
13 more rounds
P16 (32 bits)
XOR
F
XOR
P18 (32 bits)
XOR
P17 (32 bits)
XOR
32 bits
32 bits
Cipher text (64 bits)


Plain text Y (64 bits)
YL (32 bits)
YR (32 bits)
P18 (32 bits)
XOR
F
XOR
P17 (32 bits)
XOR
F
XOR
13 more rounds
P3 (32 bits)
XOR
F
XOR
P1 (32 bits)
XOR
P2 (32 bits)
XOR
32 bits
32 bits
Plaintext (64 bits)

2.7 CAST-128

CAST-128 belongs to the class of encryption algorithms known as Feistel ciphers. The overall operation is similar to the Data Encryption Standard (DES).

Description of Algorithm

The full encryption algorithm is given in the following four steps.

INPUT: 64 bit plaintext; m1...m64

KEY K : 128 bit; k1...k128.

OUTPUT: 64 bit ciphertext c1...c64.

Steps

1. Compute 16 pairs of subkeys {Kmi, Kri} from K (key schedule)

2. Split the plaintext into left and right 32-bit halves

   L0 = m1...m32 and

   R0 = m33...m64.

3. There are 16 rounds.

   for i from 1 to 16, compute Li and Ri as follows:

   Li = Ri-1;

   Ri = Li-1 ^ f(Ri-1,Kmi,Kri),

   (f is of Type 1, Type 2, or Type 3, depending on i).

4. Exchange final blocks L16, R16 and concatenate to form the ciphertext.

   c1...c64 <-- (R16,L16)

Decryption is identical to the encryption algorithm given above, except that the rounds and therefore the subkey pairs are used in reverse order to compute (L0,R0) from (R16,L16).

Pairs of Round Keys

CAST-128 uses a pair of subkeys per round: a 32-bit quantity Km is used as a “masking" key and a 5-bit quantity Kr is used as a "rotation" key.

Non-Identical Rounds

Three different round functions are used in CAST-128. The rounds are as follows (where "D" is the data input to the f function and "Ia" - "Id" are the most

significant byte through least significant byte of I, respectively). Note that "+" and "-" are addition and subtraction (modulo 2**32), "^" is bitwise XOR, and "<<<" is the circular left-shift operation.

Type 1: I = ((Kmi + D) <<< Kri)

f = ((S1[Ia] ^ S2[Ib]) - S3[Ic]) + S4[Id]

Type 2: I = ((Kmi ^ D) <<< Kri)

f = ((S1[Ia] - S2[Ib]) + S3[Ic]) ^ S4[Id]

Type 3: I = ((Kmi - D) <<< Kri)

f = ((S1[Ia] + S2[Ib]) ^ S3[Ic]) - S4[Id]

Rounds 1, 4, 7, 10, 13, and 16 use f function Type 1.

Rounds 2, 5, 8, 11, and 14 use f function Type 2.

Rounds 3, 6, 9, 12, and 15 use f function Type 3.

Substitution Boxes

CAST-128 uses eight substitution boxes:

S1, S2, S3, and S4 are round function s-boxes

S5, S6, S7, and S8 are key schedule s- boxes.

Although 8 s-boxes require a total of 8 KBytes of storage, note that only 4 KBytes are required during actual encryption /decryption since subkey generation is typically done prior to any data input.

Variable Keysize

The CAST-128 encryption algorithm has been designed to allow a key size that can vary from 40 bits to 128 bits, in 8-bit increments (that is, the allowable key sizes are 40, 48, 56, 64, ..., 112, 120, and 128 bits).

For variable keysize operation, the specification is as follows:

1) For key sizes up to and including 80 bits (i.e., 40, 48, 56, 64,72, and 80 bits), the algorithm is exactly as specified but uses12 rounds instead of 16.

2) For key sizes greater than 80 bits, the algorithm uses the full 16 rounds.

3) For key sizes less than 128 bits, the key is padded with zerobytes (in the rightmost, or least significant, positions) out to128 bits (since the CAST-128 key schedule assumes an input key of128 bits).

In order to avoid confusion when variable keysize operation is used,the name CAST-128 is to be considered synonymous with the name CAST5;this allows a keysize to be appended without ambiguity. Thus, forexample, CAST-128 with a 40-bit key is to be referred to as CAST5-40;where a 128-bit key is explicitly intended, the name CAST5-128 should be used.

Strength of CAST-128

CAST-128 is a 12- or 16-round Feistel cipher that has a blocksize of 64 bits and a keysize of up to 128 bits; it uses rotation to provide intrinsic immunity to linear and differential attacks; it uses a mixture of XOR, addition and subtraction (modulo 2**32) in the round function; and it uses three variations of the round function itself throughout the cipher. Finally, the 8x32 s-boxes used in the round function each have a minimum nonlinearity of 74 and a maximum entryof 2 in the difference distribution table.

This cipher appears to have cryptographic strength in accordance with its keysize (128 bits) and has very good encryption / decryption performance: 3.3 MBytes/sec on a 150 MHz Pentium processor.

2.8 CONFIDENTIALITY USING SYMMETRIC ENCRYPTION

The focus of cryptology has been on the use of symmetric encryption to provide confidentiality. We will discuss about,

- Location of encryption Logic
- Link and end-to-end encryption
- Traffic anlaysis attack
- Key distribution
- Random number Generation

Placement of Encryption Function

If encryption is to be used to counter attacks on confidentiality, we need to decide what to encrypt and where the encryption function should be located.

The figure 1 shows the types of communication facilities to indicate the points of culnerability.

Potential Locations for Confidentiality Attacks

The Locations for confidentiality attacks are mentioned as follows:

i) The user can reach other workstations hosts and servers directly on the LAN or on other LAN through bridges or routers. Here, is the first point of vulnerability.

ii) The main concern in this case is Eavesdropper. An Eavesdropper can monitor the traffic on the LAN and Capture the traffic on the basis of Source and Destination address.

iii) The wiring closet itself is vulnearable.

iv) An attack can take place on any of the communication links.

v) The various processors along the path are themselves subject to attack.

Thus, there are large number of locations at which an attack can occur.

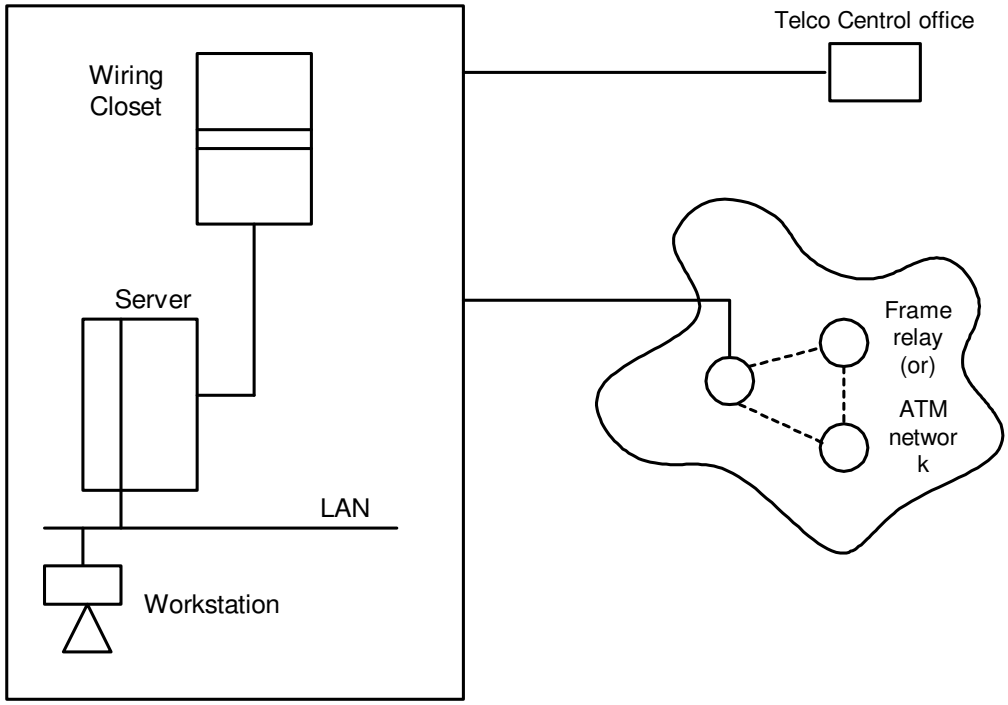


Fig.1 Points of Vulnerability

Link Verses End-to-End Encryption

The most powerful approach to secure the points of vulnerability is Encryption. There are two fundamental alternatives and encryption.

1) Link Encryption

- In Link Encryption, each Vulnearable Communication Link is equipped on both ends with an encryption device.
- Thus, all traffic over all links are secured.

- Since it requires more encryption device, its value is clear.

Disadvantage

The message must be decrypted each time it enters a switch because the switch must read the address in the packet header to route the frame. Thus message is vulnerable at each switch.

2) End-to-End Encryption

- In End-to-End encryption, encryption process is carried out at the two end systems.
- The data could be encrypted by the source host or terminal. Then encrypted data are then transmitted unaltered to destination.
- The destination shares a key with source and decrypt the data.
- This will secure the transmission against attacks on network or switches.

Traffic Confidentiality

The users must concerned about security from traffic analysis. In commercial applications, traffic analysis may yield information that the traffic generators would like to conceal.

The following are the list of information that can be derived from traffic analysis attack:

- Identities of partners
- How frequently the partners are communicating
- Message pattern, message length or quantity of message suggest important information is being exchanged.
- Events between particular partners.

Another concern is the use of traffic patterns to create a Covert channel.

Covert Channel – Means of communication in a fashion used to transfer information in a way that violets a security policy.

The effective countermeasure for this type of attack is traffic padding.

Traffic Padding – It produces ciphertext even in the absence of plaintext. This make impossible for an attacker to distinguish between true data and padding.

This process of traffic-padding is shown in figure 2.

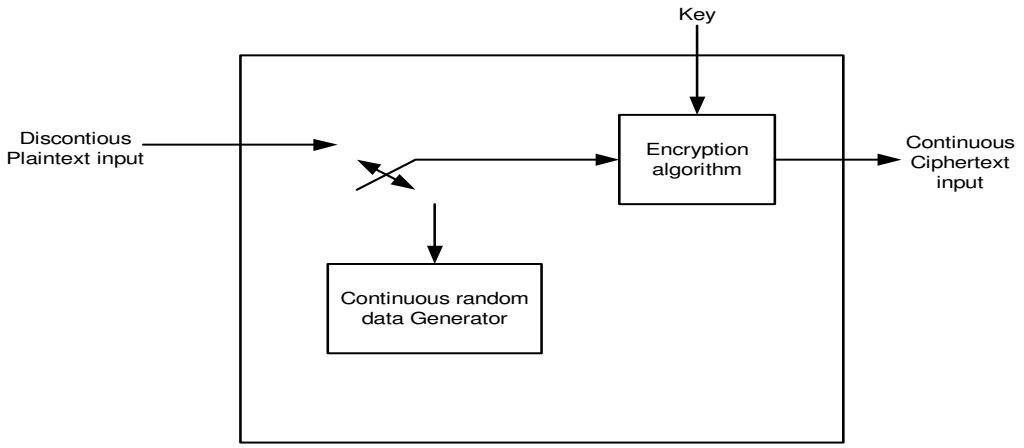


Fig. 2 Traffic-Padding Encryption Device

Key Distribution

For Symmetric Encryption two parties must be exchange the same key and that key must be protected from access by others.

Key Distribution Technique – It refers to means of delivering a key to two parties to exchange data without allowing others to see the key.

The key distribution can be achieved in number of ways. They are as follows:

1) A can select a key and deliver it to B.
2) A third party can select a key and deliver it to A and B
3) If A and B have a key, are party can transmit the new key to other, encrypted using the old key.
4) If A and B each has an encrypted connection to third party C, 'C' can deliver a key to A and B.

Random Number Generation

Random Numbers play an important role in the use of encryption for various network security applications.

Use: A number of network security algorithm based on cryptography make use of random numbers.

For examples

1) Reciprocal authentication Schemes
2) Session key generation
3) RSA public-key encryption algorithm.

❑

UNIT – III

3.1 PRINCIPLES OF PUBLIC-KEY CRYPTOSYSTEMS

Public-key Cryptosystems

Public-key algorithms rely on one key for encryption and a different but related key for decryption.

These algorithms have the following important characteristics:

1. It is computationally infeasible to determine the decryption key given only knowledge of the Cryptographic algorithm and the encryption key.

2. Either of the two related keys can be used for encryption, with the other used for decryption.

Public-key Encryption

A public-key encryption scheme has six ingredients

1. Plaintext

2. Encryption algorithm

3. Public key

4. Private key

5. Ciphertext

6. Decryption Algorithm

Plaintext: This is the readable message or data. It is fed into the algorithm as input.

Encryption Algorithm: It performs various transformations on the plaintext.

Public and Private key: This is a pair of keys in which one is used for encryption and the other is used for decryption.

Ciphertext: This is the scrambled message produced as output.

Decryption algorithm: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

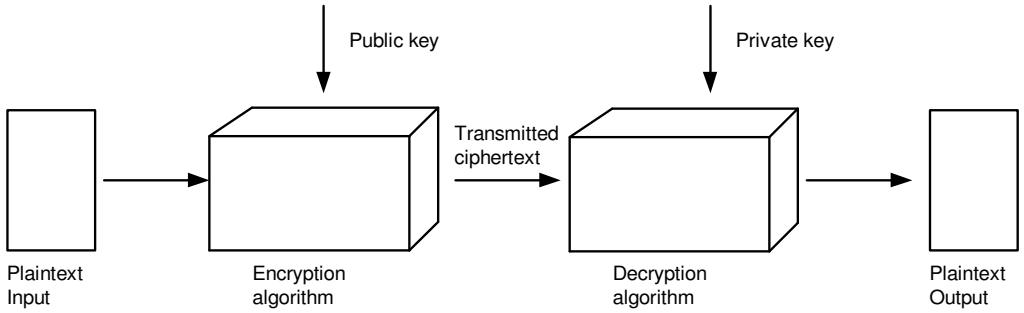


Fig. 3.1 Public key Encryption

Elements of Public-key Encryption

The essential elements of a public-key encryption is shown in Fig. 3.2.

Source A produces the message in plaintext, $X = [x_1, x_2, \ldots x_M]$. The M elements of X are letters in some finite alphabet. The message is intended for destination B.

B generate a related pair of keys:

1) a public key kU_b

2) a private key kR_b

kR_b is known only to B, whereas kU_b is publicly available and therefore accessible by A.

With the message X and the encryption key kU_b as input, A forms the ciphertext $Y = [y_1, y_2, \ldots y_N]$

$$Y = E_{KU_b}(x)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = D_{KR_b}(Y)$$

Either of the two related keys can be used for encryption, with the other being used for decryption. This enables a rather different cryptographic scheme to be implemented.

The scheme illustrated in fig. 2 provides confidentiality whereas, the fig. 3 shows the use of public-key encryption to provide authentication.

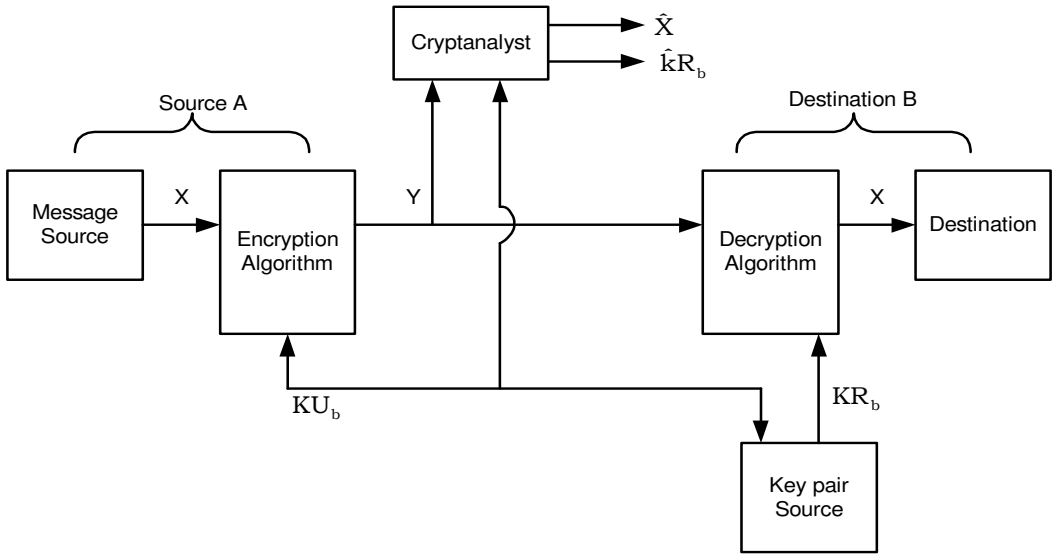


Fig. 3.2 Public key Cryptosystem: Secrecy

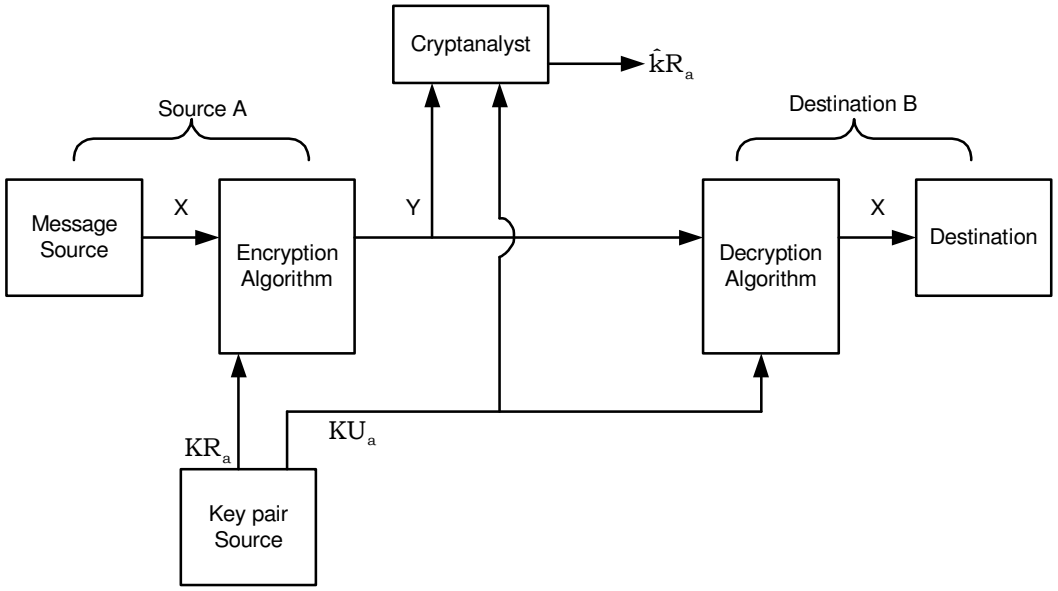


Fig. 3.3 Public key Cryptosystem: Authentication

$$Y = E_{KR_a}(x)$$

$$X = D_{KU_a}(Y)$$

A prepares a message to B and encrypts it using A's private key before transmitting it.

B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a digital signature.

In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

Applications for public-key Cryptosystems

The use of public-key cryptosystems can be classified into three broad categories:

1) Encryption/Decryption
2) Digital Signature
3) Key Exchange

Encryption/Decryption: The sender encrypts a message with the recipient's public key.

Digital Signature: The sender "Signs" a message with its private key.

Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

Key Exchange: Two sides cooperate to exchange a session key.

Some algorithms are suitable for all the three applications, whereas others can be used only for one or two of these applications.

The following table indicates the applications supported by some of the algorithms.

Table 3.1: Applications for Public-key Cryptosystems

S.No.	ALGORITHM	ENCRYPTION/ DECRYPTION	DIGITAL SIGNATURE	KEY EXCHANGE
1.	RSA	Yes	Yes	Yes
2.	Elliptic Curve	Yes	Yes	Yes
3.	Diffie-Hellman	No	No	Yes
4.	DSS	No	No	No

Requirements for Public-Key Cryptography

The conditions that the algorithms must fulfill:

1) It is computationally easy for a party B to generate a pair of keys.
2) It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M to generate the corresponding ciphertext

$$C = E_{KU_b}(M)$$

3) It is Computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message

$$M = D_{KR_b}(C) = D_{KR_b}[E_{KU_b}(M)]$$

4) It is computationally infeasible for an opponent, knowing the public key, KU_b to determine the private key.
5) It is computationally infeasible for an opponent, knowing the public key, KU_b and a ciphertext C, to recover the original message M.
6) The encryption and decryption functions can be applied in either order.

$$M = E_{KU_b}[D_{KR_b}(M)] = D_{KU_b}[E_{KR_b}(M)]$$

Public-key Cryptanalysis

A public key encryption scheme is vulnerable to brute-force attack. The countermeasure is to use large keys.

Using large keys make brute-force attack impractical, but result in encryption/decryption speeds that are too slow for general- purpose use.

Another form of attack is to find some way to compute the private key given the public key.

Finally, another form of attack is a probable-message attack.

3.2 RSA ALGORITHM

Introduction

The RSA algorithm is the most popular and proven asymmetric key cryptographic algorithm.

This algorithm is based on prime numbers.

A prime number is the one that is divisible only by 1 and itself.

Algorithm

1) Choose any two prime numbers P and Q
2) Calculate N

$$N = P \times Q$$

3) Calculate $\Phi(n)$

$\Phi(n) = (P-1) \times (Q-1)$

4) Select the public key (i.e., Encryption key) E, such that it is not a factor of $\Phi(n)$.
5) Select the private key (i.e., Decryption key) D, such that

   $(D \times E) \bmod (P-1) \times (Q-1) = 1$.
6) Calculate cipher text CT. (Encryption)

   $CT = PT^E \bmod N$
7) Send this cipher text to receiver.
8) Calculate plain text PT. (Decryption)

   $PT = CT^D \bmod N$

Example:

Let P=3 Q=5 and plain text=3. Calculate the output using RSA algorithm.

1. P = 3 Q = 5

2. N = P×Q

   =3×5

   N = 15

3. $\Phi(n) = (P-1) \times (Q-1)$

   $= (3-1) \times (5-1)$

   $\Phi(n) = 8$

4. Select encryption key E.

   E = 3 [since 3 is not a factor of 8]

5. Select Decryption key D.

   $(D \times E) \bmod \Phi(n) = 1$

   $(D \times 3) \bmod 8 = 1$

   Assume D = 3

   $(3 \times 3) \bmod 8 = 1$

   1=1

   Therefore, D = 3

6. Calculate Cipher text

   $CT = PT^E \bmod N$

$= 3^3 \bmod 15$

= 27 mod 15

CT = 12

7. Send cipher text to receiver.

8. Calculate plain text

$PT = CT^D \bmod N$

$= 12^3 \bmod 15$

= 1728 mod 15

PT = 3

3.3 ELLIPTIC CURVE CRYPTOGRAPHY

There are two approaches to elliptic curve cryptography

1) Analog of Diffie-Hellman key Exchange
2) Elliptic curve Encryption/Decryption

Analog of Diffie-Hellman Key Exchange

Key exchange using elliptic curves can be done in the following manner.

1) Pick a large integer q, which is either a prime number p or an integer of the form 2^m and elliptic curve parameters a and b.
2) Pick a base point $G = (x_1, y_1)$ in $E_p(a, b)$ whose order is a very large value n.
3) The order n of a point G on an elliptic curve is the smallest positive integer n such that nG = 0.Eq (a,b) and G are parameters of the cryptosystem known to all participants.

A key exchange between users A and B can be accomplished as follows:

1) A selects an integer n_A less than n. This is A's private key. A then generates a public key $\rho_A = n_A \times G$; the public key is a point in Eq(a,b).
2) B similarly selects a private key n_B and computes a public key ρ_B.
3) A generates the secret key $K = n_A \times \rho_B$. B generates the secret key $K = n_B \times \rho_A$.

Elliptic Curve Encryption / Decryption

1. Encode the plaintext message m to be sent as an x-y point ρ_m.

2. As with key exchange system, an encryption/decryption system requires a point G and an elliptic group Eq (a,b) as parameters.

3. Each user A selects a private key n_A and generate a public key $\rho_A = n_A \times G$.

4. To encrypt and send a message ρ_m to B, A chooses a random positive integer k and produces the cipher text, B multiplies the first point in the pair by Bs' secret key and subtracts the result from the second point:

$$C_m = \{K_G, \rho_m + k\rho_B\}$$

5. A has used B's public key ρ_B. To decrypt this ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

$$\rho_m + k\rho_B - n_B(kG) = \rho_m + (n_B G) - n_B(kG) = \rho_m$$

Security of Elliptic Curve Cryptography

The security of ECC depends on how difficult it is to determine k given kp and ρ. This is referred to as the elliptic curve logarithm problem. The fastest known technique for taking the elliptic curve logarithm is known as the pollard rho method.

$b_{ij} \rightarrow$ ith bit in jth block

$\oplus \rightarrow$ XOR operation

3.4 MESSAGE AUTHENTICATION AND HASH FUNCTIONS

Authentication Requirements

In the communications across a network, the following attacks can be identified.

i) Disclosure: Release of message contents to any person.
ii) Traffic analysis: Discover of the pattern of traffic between parties.
iii) Masquerade: Insertion of message into the network from a fraudulent source.
iv) Content modification: Changes to a contents of a message.
v) Sequence modification: any modification to a sequence of message b/w parties.
vi) Timing modification: Delay or replay of messages.

vii) Source repudiation: Denial of transmission of message by source.

viii) Destination repudiation: Denial of receipt of message by destination.

Message authentication is a procedure to verify that received messages come from the source have not been altered.

Authentication Functions

Any message authentication mechanism can be viewed as two levels.

Lower level – Functions that produces an authenticator.

Higher level – to verify the authenticity of a message.

There are three types of functions used to produce an authenticator. They are

i) Message Encryption
ii) Message Authentication Code (MAC)
iii) Hash Functions

i) Message Encryption

The ciphertext of the entire message serves as its authenticator. The analysis differs for symmetric and public key encryption schemes.

Symmetric Encryption

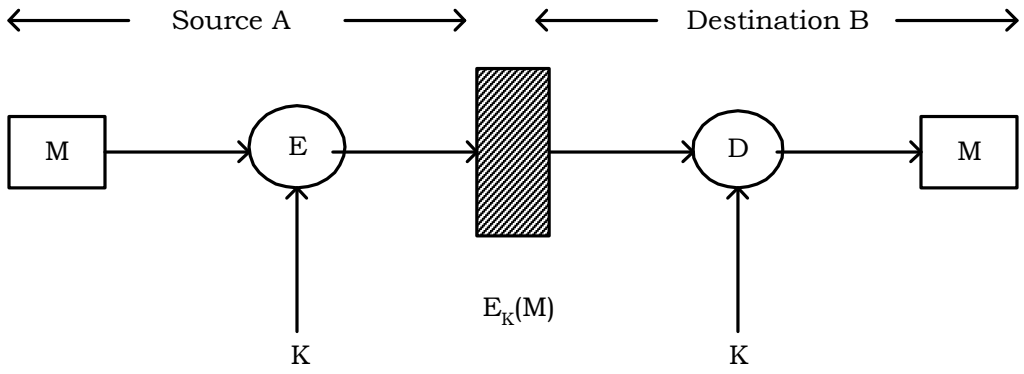


Fig. 3.4 Symmetric Encryption: Confidentiality and Authentication

The above figure 3.4 illustrate the use of symmetric encryption. A message M transmitted from source A to Destination B is encrypted by using a secret key K, it is shared by both A and B. If no other party knows the key, then confidentiality is provided no one can recover the plaintext of the message.

Symmetric encryption provides authentication as well as confidentiality.

Public-key Encryption

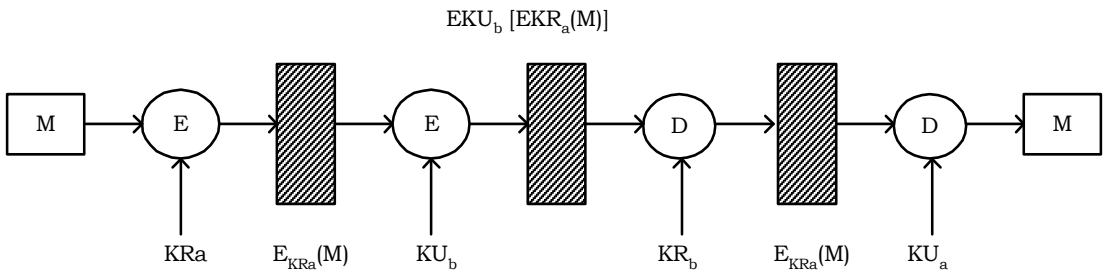


Figure 3.5 Public-key Encryption: Confidentiality, Authentication and Signature

To provide both confidentiality and authentication, A can encrypt message M using its private key, which provides the digital signature and then using B's public key which provides confidentiality.

The disadvantage of this approach is that public key algorithm is complex in each communication.

ii) Message Authentication Code

It is an alternative technique involves the use of a secret key to generate a small fixed size block of a data known as cryptographic checksum or MAC that is appended to the message.

In this technique, two parties A and B shares a common Secret key K. When A has send message to B, it calculates the MAC as a function of message and the key.

$$MAC = C_k\ (M)$$

Where

M $\rightarrow$ input message

C $\rightarrow$ MAC function

K $\rightarrow$ shared secret key

MAC $\rightarrow$ message authentication code

The message plus MAC are transmitted to recipient. The recipient also performs same calculation using same secret key to generate a new MAC. Then the received MAC is compared to calculated MAC, if it is matches, then the receiver get assured that the message has not been altered.

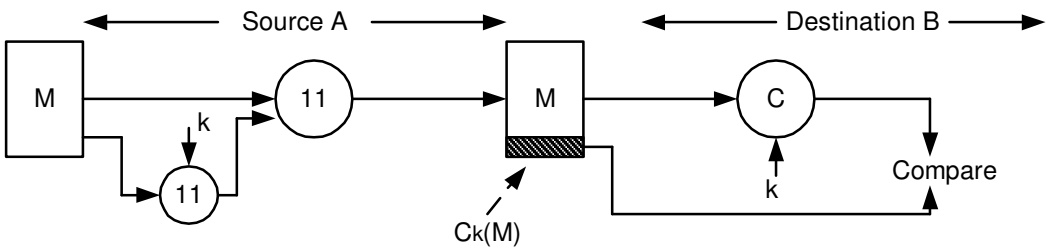


Fig. 3.6 Basic use of MAC

iii) Hash function

It is the variation of message authentication code. It accepts a variable – size message 'M' as input and produces a fixed-size output, referred to as hash code H(M).

The hash code is a function of all the bits of the message and not use a key and it provides an error-detection capability. A change to any bits in the message results in a change to the hash code.

Hash Functions

A hash value h is generated by a function H of the form,

$$h = H(M)$$

Where m $\rightarrow$ variable – length message

H(M) $\rightarrow$ fixed – length hash value.

Requirements for a Hash function

The purpose of hash function is to produce a finger print of a file message. To be useful for message authentication, a hash function H must have following properties.

- H can be applied to a block of data of any size.
- H produces a fixed-length output.
- H(x) is easy to compute for any given x.
- For any given value h, it is infeasible to find x, such that H(x) =h. This is referred to an one-way property.
- For any given block x, it is infeasible to find $y \neq x$, with H(y) = H(x). This is referred to as Weak collision resistance.

- For any pair (x, y) it is infeasible to find any pair (x,y) such that H(x) = H(y). This is sometimes referred to as strong collision resistance.

Simple Hash Functions

One of the simplest hash function is bit-by-bit exclusive – OR (XOR) of every block.

This can be expressed as follows:

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

Where.

$C_i \rightarrow$ ith bit of the hash code $1 \leq i \leq n$

$M \rightarrow$ number of n-bit blocks in the input

3.5 MD5 MESSAGE DIGEST ALGORITHM

Introduction

- ❖ The MD5 message digest algorithm was developed by Ron Rivest at MIT.
- ❖ It was widely used secure hash algorithm.
- ❖ Message digest → It is a fingerprint or the summary of a message.

MD5 Logic

- ❖ The algorithm takes a input message of any length and produces an output of 128-bit message digest.
- ❖ The input is processed in 512-bit blocks, which are further divided into 16 32-bit sub-blocks.
- ❖ The figure 1 depicts the overall processing of a message to produce a digest.

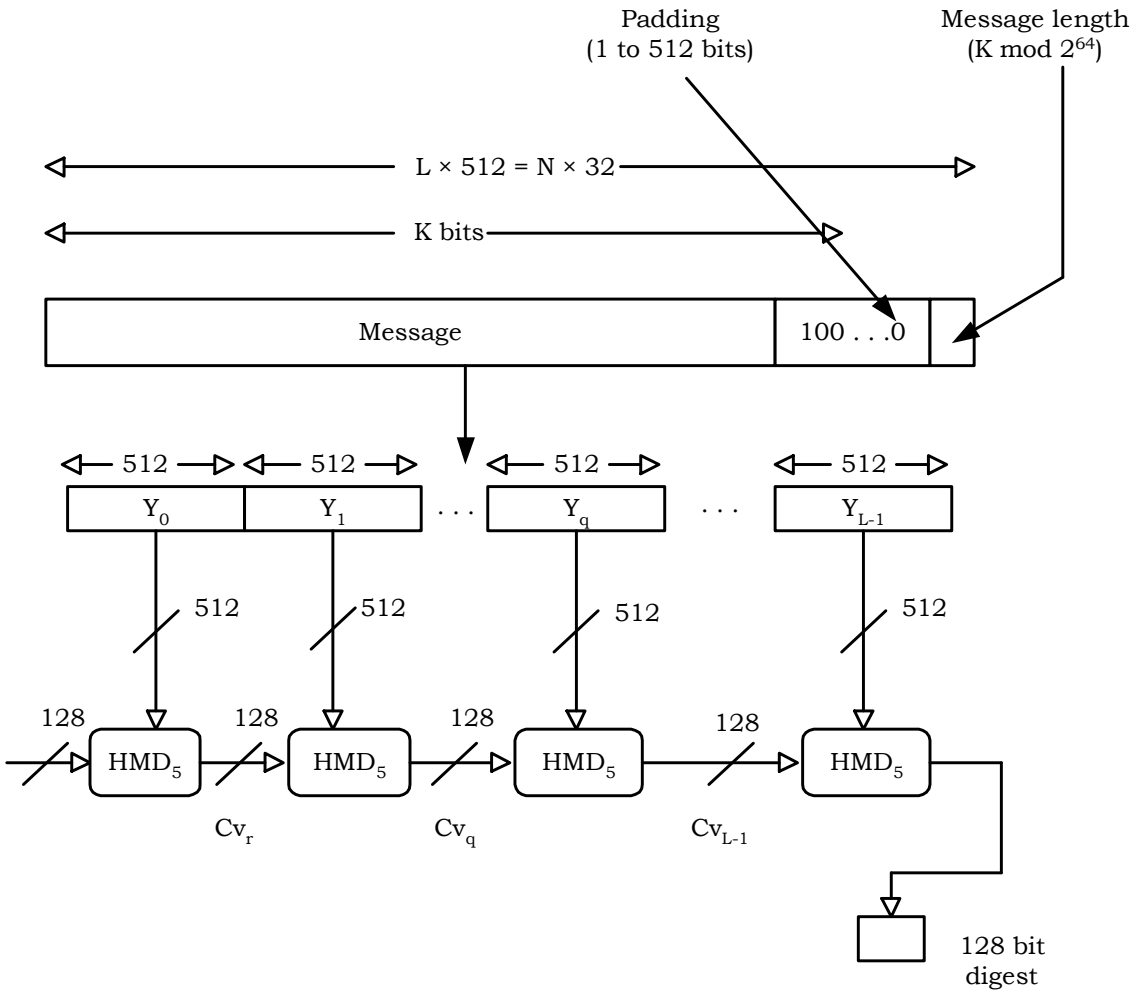


Fig.3.7 Message Digest Generation Using MD_5

STEPS

Step 1: Append Padding Bits

In this step, we add padding bits to original message, so that to make the length of the original message is equal to a value, which is 64 bit less than an exact multiple of 512.

Note: Padding is always added even if the message is already a multiple of 512.

Step 2: Append length

A 64-bit representation of the length of the original message (before padding) is appended to the result of step 1 (LSB first). Suppose if the length of the original message is greater than 2^{64}, then only we use low-order 64 bits of

the length. Finally the field contains the exact multiple of 512. (i.e., original message length mod 2^{64}).

Step 3: Divide the input into 512 bit blocks

Now, we divide the input message into blocks each of length 512 bits. In above figure, it is represented as sequence of 512-bit blocks $Y_o, Y_1 \ldots Y_{L-1}$.

Step 4: Initialize MD buffer

A 128-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as four 32-bit registers A, B, C and D. These registers are initialized to 32-bit integers (Hexadecimal values).

The initial hexadecimal values are as follows.

A:	01	23	45	67
B:	89	AB	CD	EF
C:	FE	DC	BA	98
D:	76	54	32	10

Step 5: Process message in 512 bit blocks

The heart of the algorithm is a compression function that consists of four rounds of processing. This is represented as HMD_5 in above figure.1

Its logic is illuminated in below figure 3.8.

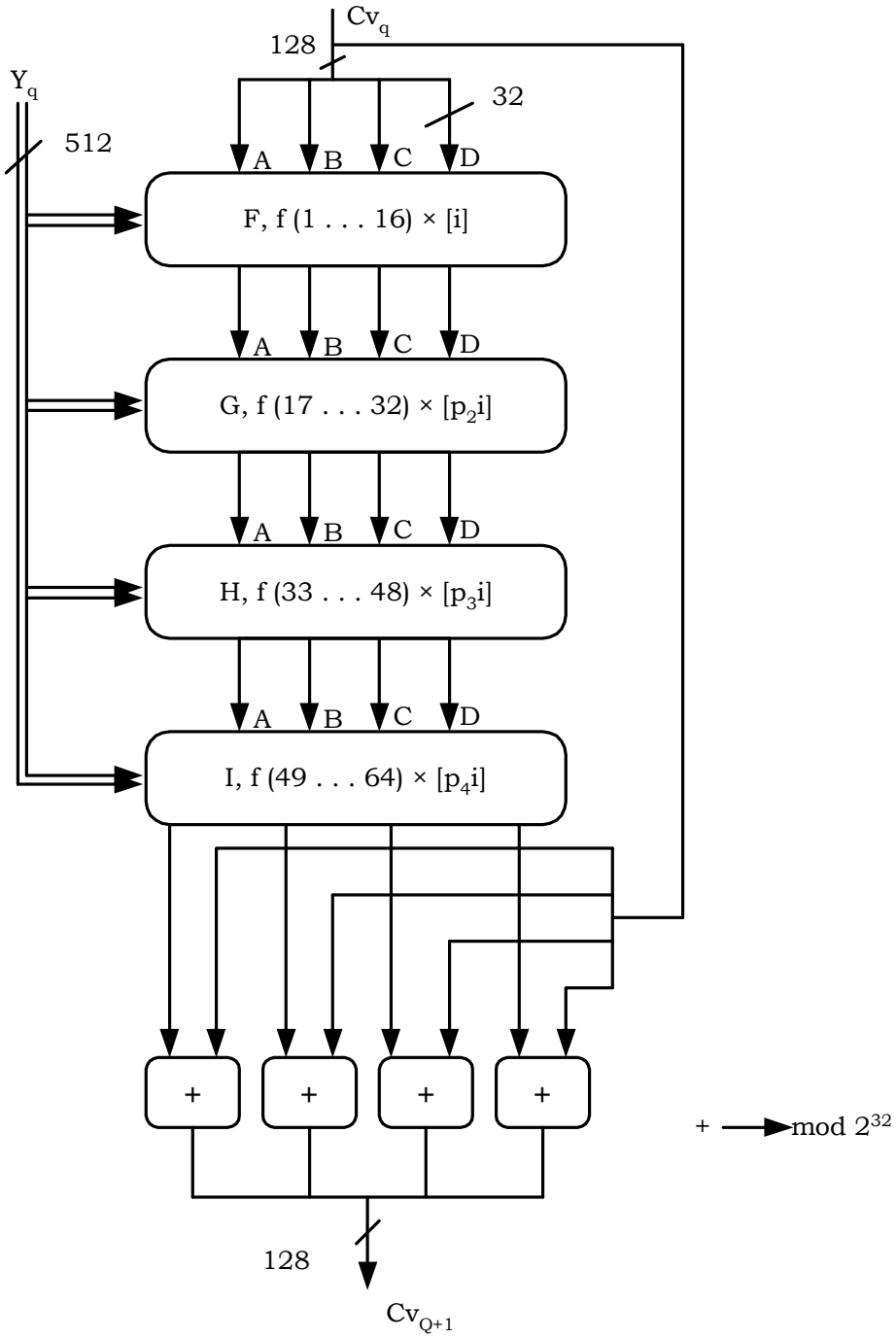


Fig. 3.8 MD5 Processing of a single 512-bit block (MD5 Compression Function)

- The four rounds have similar structure but each uses a different primitive function as F,G,H and I.
- Each round takes as input of 512 bit block (Y_q) and 128-bit buffer value ABCD and updates the content of the buffer.

- The output of fourth round is added to the input of first round (CY_q) to produce CY_{q+1}.

Step 6: Output

After all 512 bit block have been processed, the output from the Lth stage is the 128 bit message digest.

MD5 Compression function

Each round consists of sequence of 16 steps operating on the buffer ABCD.

Each step is of the form

a←b+((a+g(b,c,d)+X[K]+T[i])<<<s)

Where,

a,b,c,d	→	four words of the buffer.
G	→	one of the primitive function F,G,H,I
<<<S	→	circular left shift (rotation) of 32 bit argument by S bits.
X[K]	→	M[q × 16 + K]=the Kth 32 bit word in the qth 512-bit block of the message.
T[i]	→	The ith 32 bit worD in matrix T.
+	→	addition modulo 2^{32}.

3.6 SECURE HASH ALGORITHM

- It was developed by the National Institute of Standards and Technology (NIST).
- SHA is based on MD4 algorithm.

SHA-1 Logic

- It takes as input message with maximum length of less than 2^{64} bits and produces the output of 160-bit message digest.
- The input is processed in 512-bit blocks.
- The overall processing of a message follows the structure shown for MD5 in figure 1.

Processing Steps

Step 1: Append padding bits

The message is padded to make the length of the original message is congruent to 448 mod 513. (i.e. length = 448 mod 512).

The number of padding bits is in the range of 1 to 512.

Step 2: Append Length

In this step, a block of 64 bit is appended to the original message. Then the block contains 64 bit integer (MSB first) and length of the original message (before padding).

Step 3: Initialize MD butter

A 160-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as five 32-bit registers (A, B, C, D and E).

These registers are initialized to following 32-bit integers (Hexadecimal values)

A : 67 45 23 01
B : EF CD AB 89
C : 98 BA DC FE
D : 10 32 54 76
E : C3 D2 E1 F0

Step 4: Process message in 512-bit blocks

It consists of 4 rounds of processing of 20 steps each. The logic is illustrated in figure 1.

The four rounds have similar structure but uses different primitive logical function f1, f2, f3, and f4.

Each round takes the input of 512-bit block (Y_q) and 160 bit buffer value ABCDE and updates the content of the buffer.

Each round also makes use of an addictive constant K_t, where $0 \le t \le 79$ indicates 80 steps across four rounds, only four constants are used.

The output of 4th round is added to the input to the first round (Cv_q) to produce Cv_{q+1}.

The addition is done for each five words in the buffer using addition module 2^{32}.

Step 5: Output

After all L 512-bit blocks have been processed 160-bit message digest is produced.

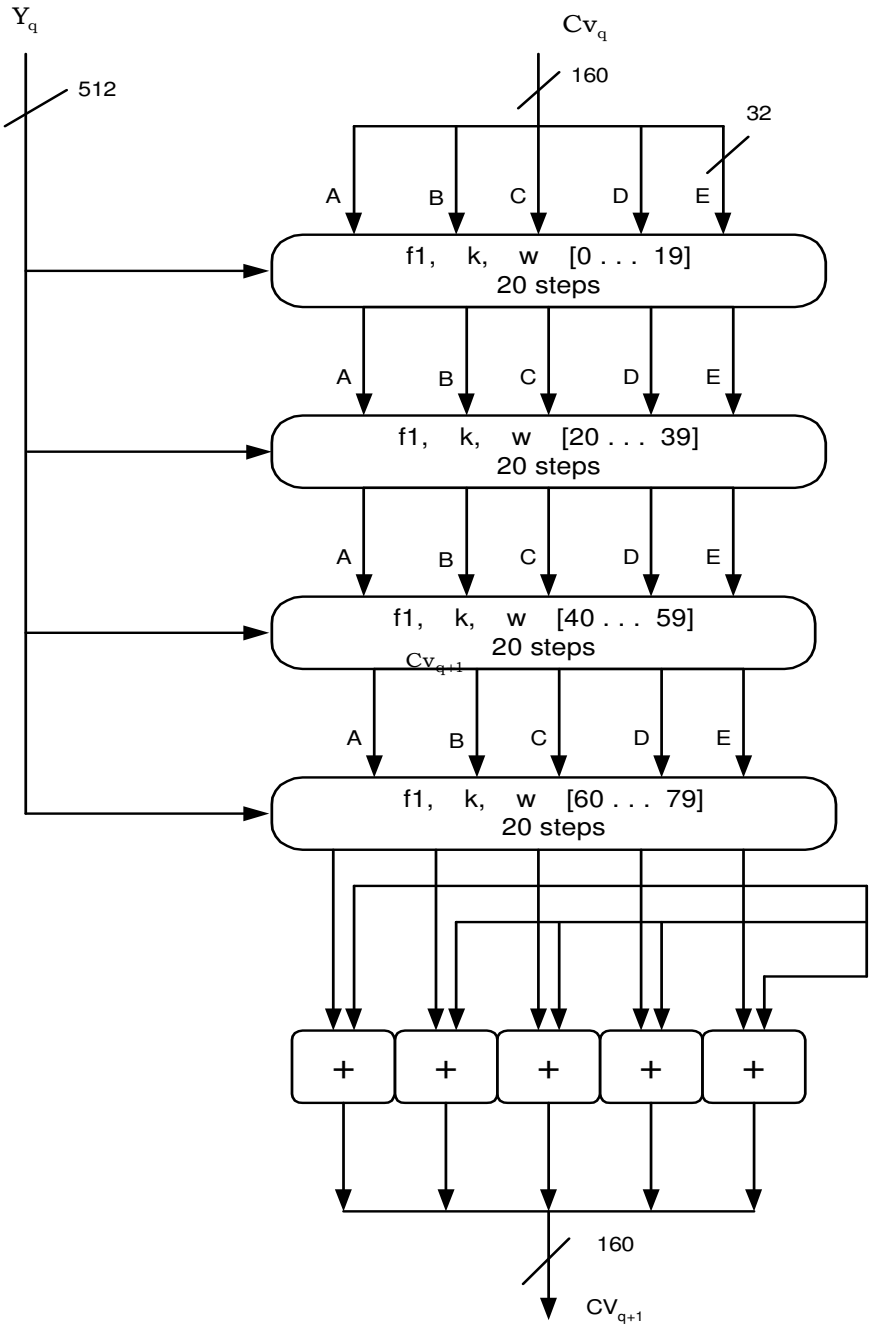


Fig. 3.9 SHA 1 Processing of a Single 512 bit Block (Sha-1 Compression function)

SHA-1 Compression Function

Each round is of the form

$$A, B, C, D, E \leftarrow (E + f(t, B, C, D) + S^5(A) + W_t + K_t), A, S^{30}(B), C, D.$$

Where

A,B,C,D,E	→	Five words of the buffer
T	→	Step number; $0 \le t \le 79$
F (t, B, C, D)	→	Primitive logical function for step t
S^k	→	Circular left shift (rotation) of the 32-bit argument by K bits
W_t	→	A 32-bit word derived from the current 512-bit input block.
K_t	→	An addictive constant
+	→	Addition modulo 2^{32}

❑

UNIT - IV

4.1 DIGITAL SIGNATURES

If there is not complete trust between sender and receiver, something more than authentication is needed. In such situations, the most attractive solution is digital signature.

Properties of Digital Signature

The digital signature must have the following properties:

It must verify the author and the date and time of the signature.

It must authenticate the contents at the time of the signature.

It must be verifiable by third parties, to resolve disputes.

Requirements for a Digital Signature

The signature must be a bit pattern that depends on the message being signed.

The signature must use some information unique to the sender, to prevent both forgery and denial.

It must be relatively easy to produce the digital signature.

It must be relatively easy to recognize and verify the digital signature.

It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.

It must be practical to retain a copy of the digital signature in storage.

Approaches for Digital Signature

A variety of approaches has been proposed for the digital signature function. These approaches fall into two categories: Direct and Arbitrated.

Direct Digital Signature

The direct digital signature involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source.

A digital signature may be formed by encrypting the entire message with the sender's private key or by encrypting a hash code of the message with the sender's private key.

Confidentiality can be provided by further encrypting the entire message plus signature with either the receiver's public key or a shared secret key.

The validity of the scheme depends on the security of the sender's private key. If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature.

Administrative controls relating to the security private keys can be employed to thwart or atleast to weaken this ploy.

Arbitrated Digital Signature

The problems associated with direct digital signature can be addressed by suing an arbiter.

The arbitrated signature scheme operate as follows:

Every signed message from a sender X to a receiver Y goes first to an arbiter A.

The arbiter A subjects the message and its signature to a number of tests to check its origin and content.

The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter.

The presence of A solves the problem faced by direct signature schemes that X might disown the message.

The arbiter plays a sensitive and crucial role in this sort of scheme, and all parties must have a great deal of trust that the arbitration mechanism is working properly.

4.2. AUTHENTICATION PROTOCOLS

THER ARE TWO GENERAL AUTHENTICATION PROTCOLS. THEY ARE:

One-way authentication
Mutual authentication

One-way Authentication

Two approaches have been used in one-way authentication. They are

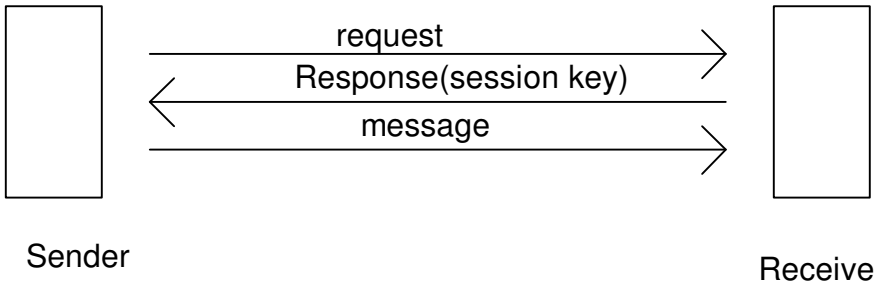
Symmetric Encryption Approach

Public-key Encryption Approach

Symmetric Encryption Approach

In, Symmetric Encryption approach, the decentralized key distribution is not possible. Because, it requires the sender to issue a request to the intended

recipient, and then wait for the response from the receiver. The response from the receiver includes a session key. Only after getting the response the sender will send the message.



So, the key distribution center (KDC) strategy is used. This is shown in the following figure:

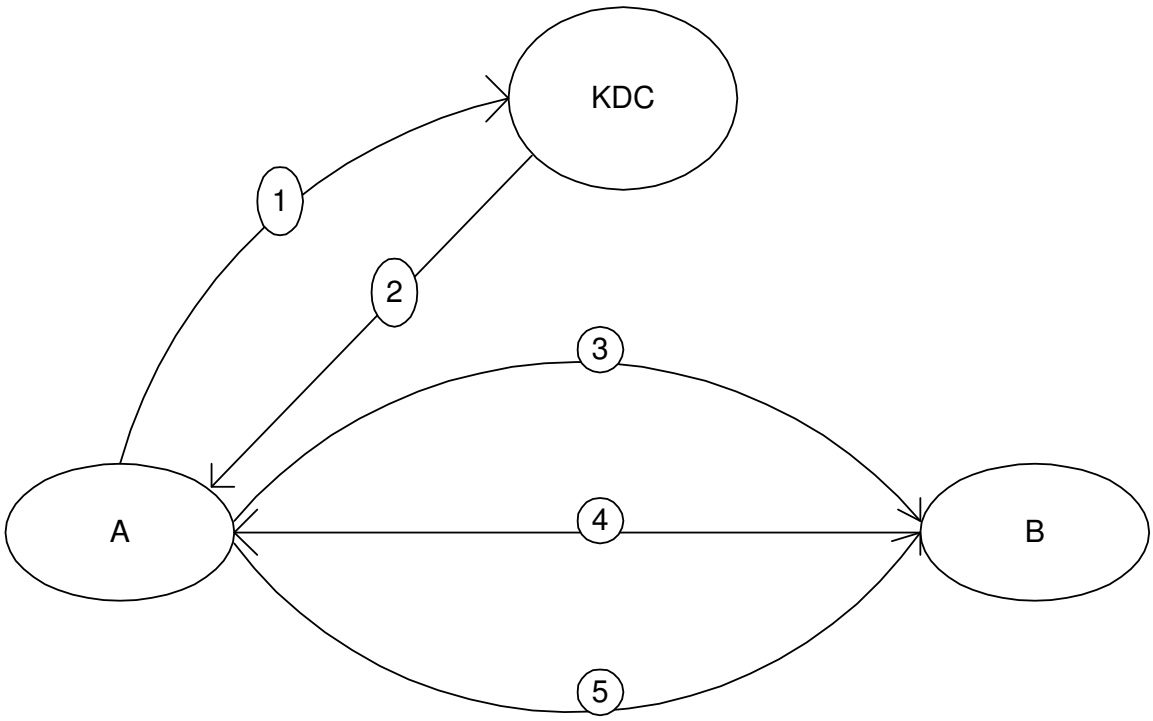


Fig. 4.1 a) key Distribution Scenario

1 and 2 → Key distribution steps

3, 4 and 5 → Authentication steps

In symmetric Encryption, the steps 4 and 5 of the KDC must be eliminated, because we do not want (B), the recipient to be online at the same time as the sender (A).

So, for a message with content M, the sequence is as follows:

A→KDC:

KDC→A:

A→B:

This approach guarantees that only the intended recipient of the message will be able to read it.

This approach also provides a level of authentication that the sender is A.

This approach does not protect against replays. So, as a measure of defense, we include a timestap with the message. But, this also has limited usefulness.

Public-key Encryption Approach

There are different Public-key Encryption approaches available. One for confidentiality, one for authentication and one for both. These approaches are illustrated in the following figure:

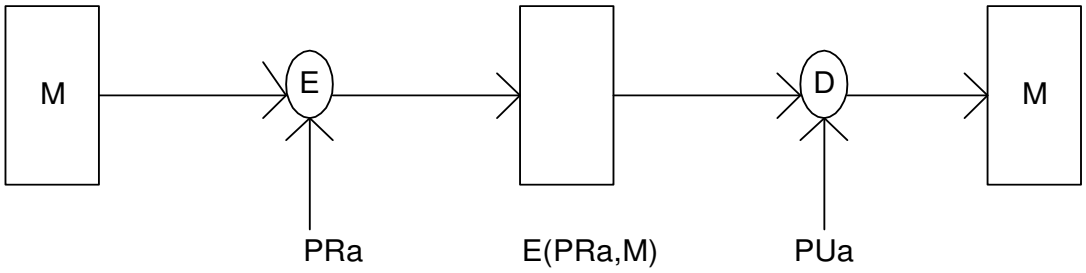
These approaches require that either the sender know the recipient's public key (confidentiality).

(or)

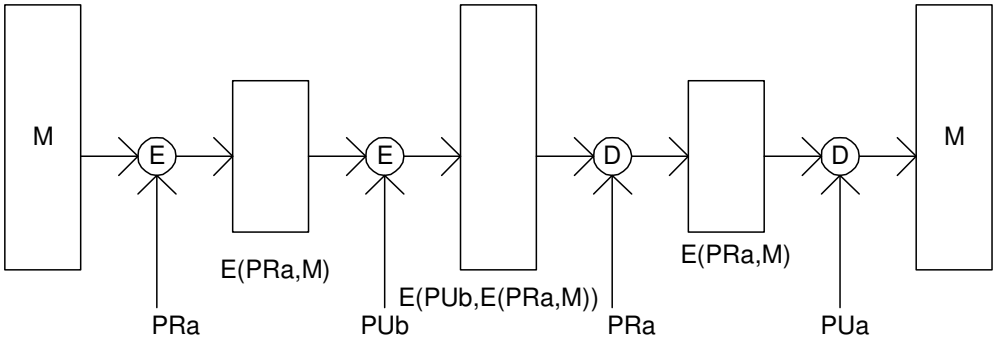
The recipients know the sender's public key (Authentication).

(or)

Both. (confidentiality plus authentication).



b) Public key Encryption: Authentication and signature



C) Public key Encryption: Confidentiality, Authentication and Signature

If confidentiality is the primary concern, then the following may be more efficient:

$A \rightarrow B: E(PU_b, K_s) || E(K_s, M)$

If authentication is the primary concern, then the following may be used:

$A \rightarrow B: M || E(PR_a, H(M))$

Mutual Authentication

Mutual Authentication protocols enable communicating parties to satisfy themselves mutually about each other's identity and exchange session keys.

There are two issues in this authenticated key exchange: they are

Confidentiality

Timeliness

Correct identification and session key information must be communicated in encrypted form, so that confidentiality is maintained.

Timeliness is important because of the threat of message replays.

A successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

Some of the examples of replay attacks are as follows:

Simple replay

Repetition that can be logged

Repetition that cannot be detected

Backward replay without modifications

One approach to overcome replay attacks is to attach a sequence number to each message used in an authentication exchange.

A new message is accepted only if its sequence number is in the proper order.

The major difficulty with this approach is the maintenance of sequence number. i.e., Each party has to keep track of the last sequence number for each claimant it has dealt with.

Because of this overhead, we go for another methods.

There are two other approaches available:

They are

Timestamps

Challenge/Response

Timestamps

In this methods, timestamp A accepts a message only if the message contains a timestamp.

This approach requires that the clocks among the various participants must be synchronized.

Challenge/Response

If party A expects a fresh message from B, then first A sends B a nonce (challenge) and waits for the message (response) from B.

The response from B must contain the correct nonce value.

Timestamp approach cannot be used for connection-oriented applications, because there is some inherent difficulties in this approach.

The challenge-Response approach is unsuitable for connection less type application, because this approach requires a handshake before any connectionless transmission.

This creates an overhead on transmission.

Symmetric Encryption Approach

This approach involves the use of a trusted key distribution center (KDC).

Each party in the network shares a secret key known as market key with the KDC.

The KDC is responsible for generating keys, and for distributing those keys using the master key to protect the distribution.

An example of this approach is Kerberos system.

The protocol for secret key distribution using a KDC is as follows:

1. A→KDC

KDC→A

A→B

B→A

A→B

Secret key K_a is shared between A and KDC. Secret key K_b is shared between B and KDC.

The purpose of this protocol is to distribute the session key K_s, to A and B in a secure manner.

The above protocol performs handshake at steps 4 and 5. The protocol is vulnerable to a form of replay attack.

To overcome this weakness, Denning proposes a protocol.

This protocol includes the addition of timestamp.

The protocol includes the additional of a timestamp to steps 2 and 3.

The Denning protocol provides an increased degree of security.

However, a new concern is raised: namely, that this scheme requires reliance on clocks that are synchronized throughout the network. The risk involved here is suppress-replay attacks.

Suppress-Replay Attacks

When a sender's clock is ahead of the intended recipient's clock, an opponent can intercept a message from the sender and replay it late when the timestamp in the message becomes current at the recipient's state. This is referred as suppress-replay Attacks.

One way to counter suppress-replay attacks is to enforce the requirement that parties check their clocks against the KDC's clock.

Public-key Encryption Approach

The protocol for public-key Encryption approach using timestamp is as follows:

A→AS

AS→A

A→B

In this case, the central system is referred to as an authentication server (AS), because it is not actually responsible for secret key distribution.

Rather, the AS provides public-key certificates. The session key is chosen and encrypted by A.

Hence there is no risk of exposure by the AS. The timestamps protect against replays of compromised keys.

This protocol is compact. But it requires synchronization of clocks.

So we go for other approaches, which makes use of nonces.

The protocol which makes use of nonces seems to be a secure protocol.

4.3 KERBEROS

What is Kerberos?

- A network authentication protocol developed at MIT as part of Project Athena
- Uses private-key cryptography for providing authentication across open networks
- Mediates authentication through a trusted 3rd party
- Developed before the popularity of public-key cryptography and systems like SSL

Kerberos: The basic protocol

The following steps explain the basic Kerberos protocol

Step 1: Kerberos authentication is based on symmetric key cryptography.

Step 2: The Kerberos KDC provides scalability.

Step 3: A Kerberos ticket provides secure transport of a session key.

Step 4: The Kerberos KDC distributes the session key by sending it to the client.

Step 5: The Kerberos Ticket Granting Ticket limits the use of the entities' master keys.

The notations that will be used in the illustrations:

- The u stands for user, s stands for resource server, and k stands for KDC.
- S stands for session key; Sus means the session key shared between the user and the resource server.
- M stands for master key; Mu is the master key of the user.

Figure 4.2

Drawing (1) in Figure 4.2 represents the session key shared between the user and resource server.

Drawing (2) represents the same session key, but this time encrypted.

Drawing (3) represents the same session key, encrypted using the master key of the user.

Kerberos design assumptions

Before diving into the nuts and bolts of the protocol, let's have a quick look at some of the design assumptions the Kerberos designers at MIT took. It is very important to keep these assumptions in mind as we run through the Kerberos internals.

- Kerberos always deals with three entities: users, servers, and a set of security servers that mediate between the users and the servers for authentication.

- Time is trusted. This is because Kerberos uses timestamps to protect against replay attacks.
- The user trusts its workstation completely. This is because Kerberos caches authentication tokens on the client side.
- The security server must be online all the time. Kerberos requires the availability of the security server in order to generate new Kerberos security tokens.
- The servers are stateless. Kerberos wants to limit the amount of security principal–related information that is kept on the server side.
- Users' password time on user machine must be minimized. Kerberos looks at a user password as a weak secret—it should be protected the best possible. One of the ways to do this is to limit its time on the user workstation. Another way is to create a key hierarchy.

Limitations of Kerberos

- Of the three A's, Kerberos only provides authentication – Other protocols (such as NIS or LDAP) are still needed for authorization
- Applications must be "Kerberized" to take advantage
  - Kerberos provides standard APIs to help with this
  - There are also PAM modules for Kerberos authentication
- Cannot migrate existing password hashes into the Kerberos database
- Authentication is only as good as the user's password
- Assumes relatively secure hosts on an insecure network

Benefits of Kerberos

- Standards-based strong authentication
- Broad operating-system support
- Provides for single sign-on (SSO) capability
- Passwords never traverse the network
- Password guessing more difficult
- Stolen authentication tickets are hard to reuse

4.4 X. 509 Directory Authentication Service

X.509 is part of the X.500 series of recommendations that define a directory service.

The directory is a server or distributed set of servers that maintains a database of information about users.

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users.

X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety contents.

X.509 is based on the use of public-key cryptography and digital signatures.

X.509 Certificates

The heart of X.509 scheme is the public key certificate associated with each user.

These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

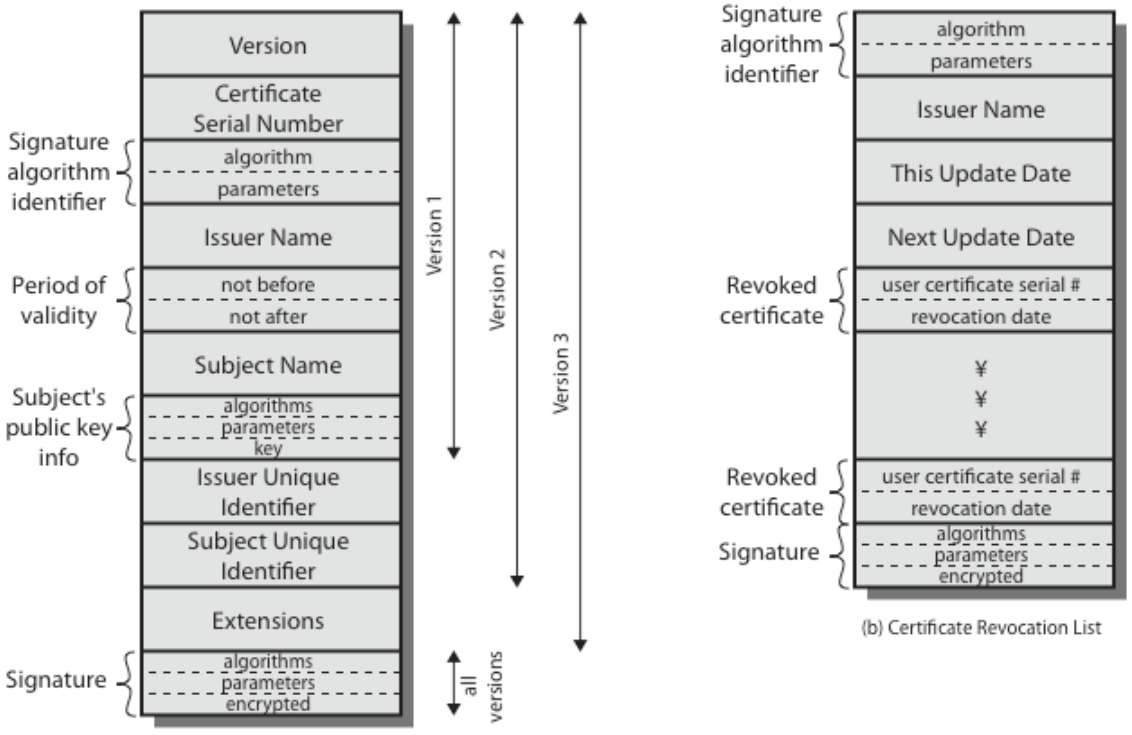


Figure 4.3 X.509 Certificate

- Issued by a Certification Authority (CA), containing:
- version (1,2, or 3)
- serial number (unique within CA) identifying certificate
- signature algorithm identifier
- issuer X.500 name (CA)
- period of validity (from – to dates)
- subject X.500 name (name of owner)
- subject public-key info (algorithm, parameters, key)
- issuer unique identifier (v2+)
- subject unique identifier (v2+)
- extension fields (v3)
- signature (of hash of all fields in certificate)
- notation CA<<A>> denotes certificate for A signed by CA

Obtaining a user's certificate

User certificate generated by a CA have the following characteristics:

Any user with access to CA can get any certificate from it

Only the CA can modify a certificate

Because certificates are unforgeable, they can be placed in a public directory.

Certification Authority Hierarchy

The fig. 4.4 shows an example of CA hierarchy. The connected circles indicate the hierarchial relationship among the CAs. The associated boxes indicate certificates maintained in the directory for each CA entry.

The directory entry for each CA includes 2 types of certificates:

Forward certificate

Reverse certificate

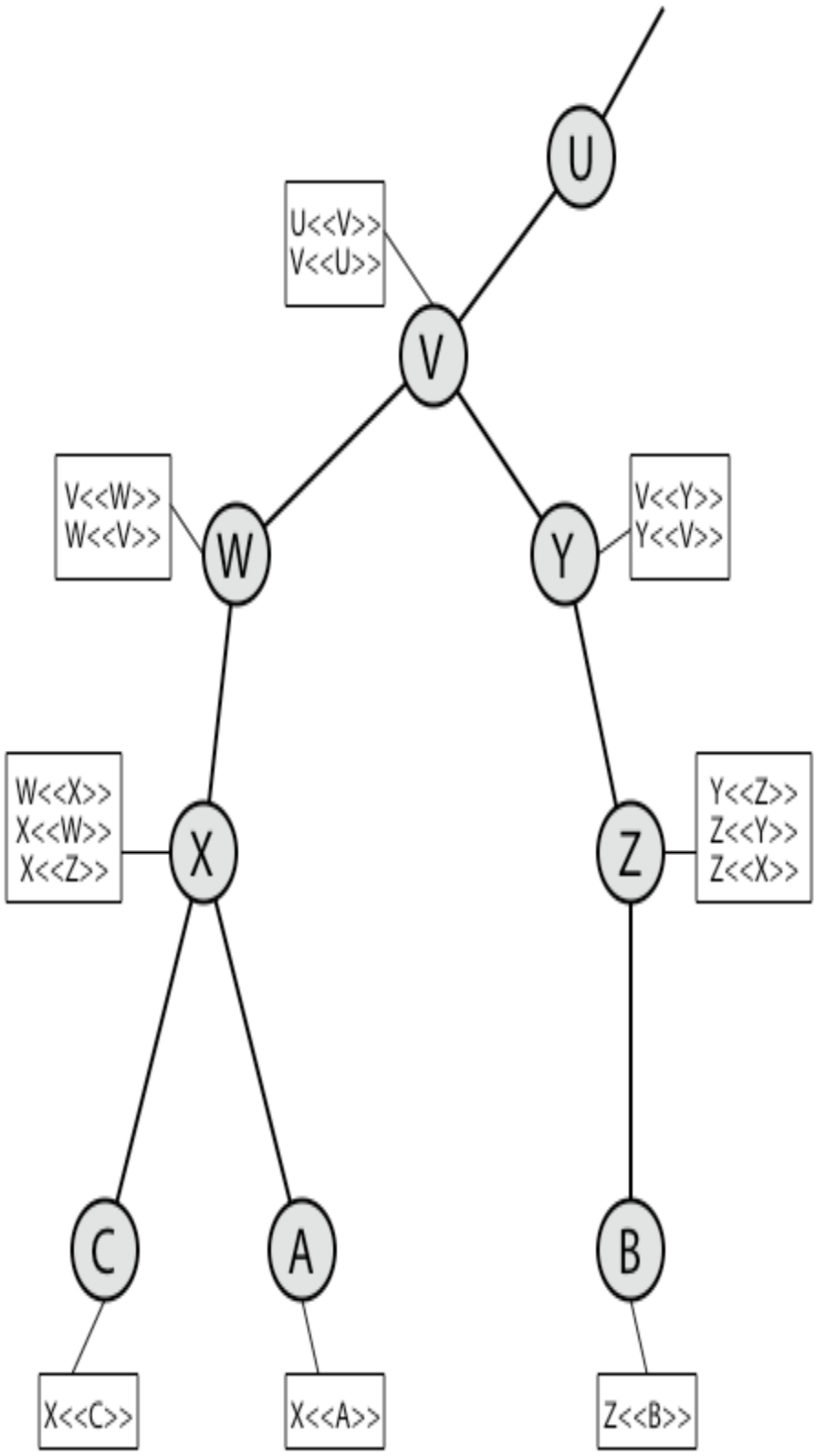


Fig. 4.4 An example of CA hierarchy

Forward Certificates

Certificates of X generated by other CA's.

Reverse Certificates

Certificates generated by X that are the certificates of other CAs.

Fig 4.3 X.509 Hierarchy: A Hypothetical example

In this example, user A can acquire the following certificates from the directory to establish a certification path to B:

X<<W>>W<<V>>Y>>Y>>Z>>Z<<B>>

When A has obtained these certificates, it can unwrap the certification path in sequence to recover a trusted copy of B's public key.

B obtains A's public key from the following certification path.

Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

B can obtain this set of certificates from the directory, or A can provide them as part of its initial message to B.

Revocation of Certificates

A new certificate is issued just before the expiration of the old one. It may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons:

The user's private key is assumed to be compromised.

The user is no longer certified by this CA.

The CA's certificate is assumed to be compromised.

Authentication Procedures

X. 509 includes three alternative authentication procedures:

1) One-Way Authentication

2) Two-Way Authentication

3) Three-Way Authentication

All use public-key signatures

One-way Authentication

1message (A $\rightarrow$) used to establish

The identity of A and that message is from A

Message was intended for B

Integrity and originality of message

Message must include timestamp, nonce, B's identity and is signed by A

May include additional info for B

Eg session key

Two-way Authentication

2 message ($A \rightarrow B$, $B \rightarrow A$) which also establishes in addition

The identity of B ad that reply is from B

That reply is intended for A

Integrity and originality of reply

Reply includes original nonce from A, also timestamp and nonce from B

May include additional info for A

Three-Way Authentication

3 messages ($A \rightarrow B$, $B \rightarrow A$, $A \rightarrow B$) which enables above authentication without synchronized clocks

Has reply from A back to B containing signed copy of nonce from B

Means that timestamps need not be checked or relied upon

X.509 Version 3

Has been recognized that additional information is needed in a certificate

Email/URL, policy details, usage constraints

Rather than explicitly naming new fields defined a general extension method

Extensions consist of

Extension identifier

Criticality indicator

Extension value

Certificate Extensions

Key and policy information

Convey info about subject and issuer keys, plus indicators of certificate policy

Certificate subject and issuer attributes

Support alternative names, in alternative formats for certificate subject and/or issuer

Certificate path constraints

Allow constraints on use of certificates by other CA's.

4.5 Public Key Infrastructure (PKI)

PKI provides assurance of public key. It provides the identification of public keys and their distribution. An anatomy of PKI comprises of the following components.

- Public Key Certificate commonly referred to as 'digital certificate'.
- Private Key tokens.
- Certification Authority.
- Registration Authority.
- Certificate Management System.

Digital Certificate

For analogy, a certificate can be considered as the ID card issued to the person. People use ID cards such as a driver's license, passport to prove their identity. A digital certificate does the same basic thing in the electronic world, but with one difference.

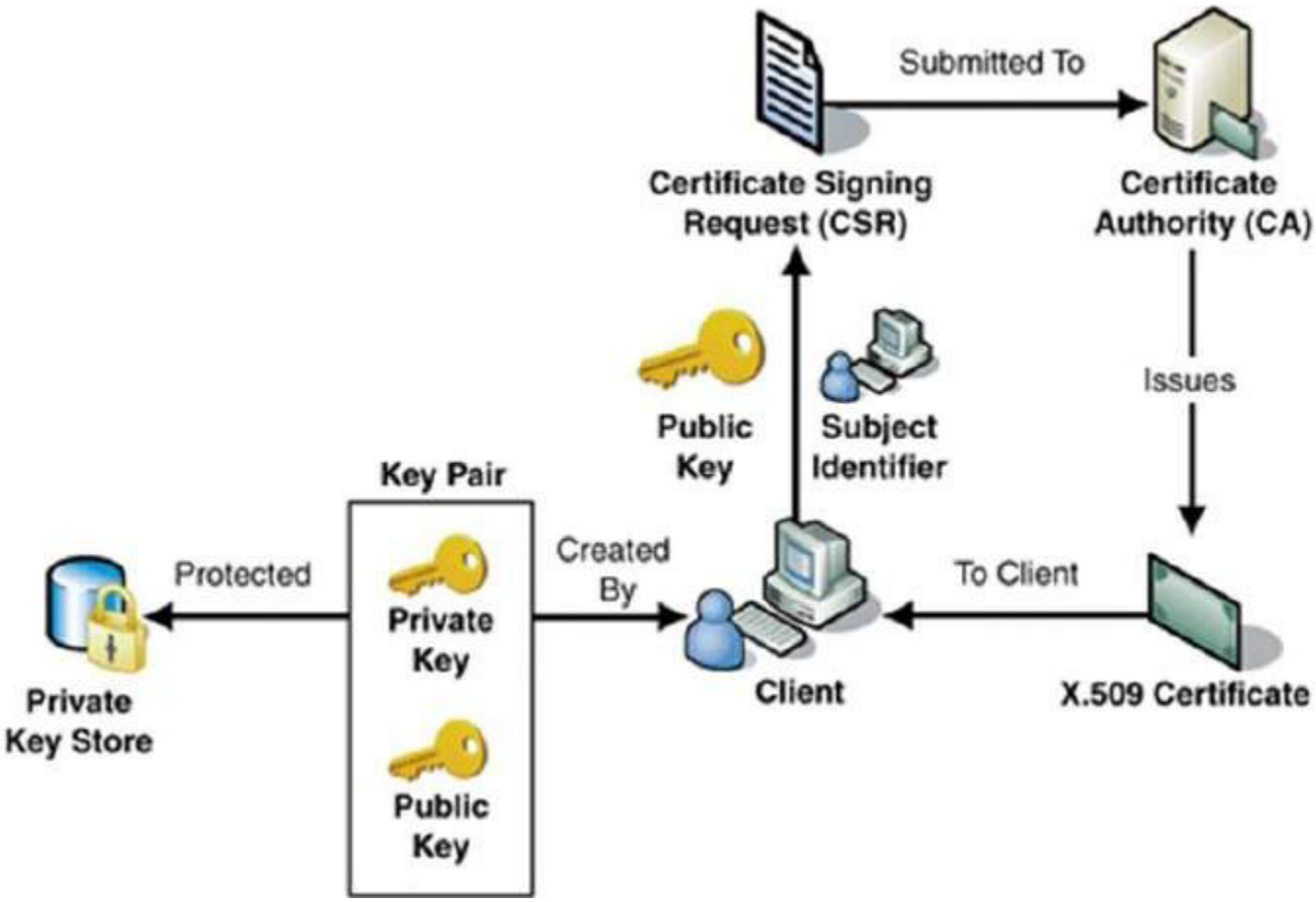
Digital Certificates are not only issued to people but they can be issued to computers, software packages or anything else that need to prove the identity in the electronic world.

- Digital certificates are based on the ITU standard X.509 which defines a standard certificate format for public key certificates and certification validation. Hence digital certificates are sometimes also referred to as X.509 certificates.

Public key pertaining to the user client is stored in digital certificates by The Certification Authority (CA) along with other relevant information such as client information, expiration date, usage, issuer etc.

- CA digitally signs this entire information and includes digital signature in the certificate.
- Anyone who needs the assurance about the public key and associated information of client, he carries out the signature validation process using CA's public key. Successful validation assures that the public key given in the certificate belongs to the person whose details are given in the certificate.

The process of obtaining Digital Certificate by a person/entity is depicted in the following illustration.



As shown in the illustration, the CA accepts the application from a client to certify his public key. The CA, after duly verifying identity of client, issues a digital certificate to that client.

Certifying Authority (CA)

As discussed above, the CA issues certificate to a client and assist other users to verify the certificate. The CA takes responsibility for identifying correctly the identity of the client asking for a certificate to be issued, and ensures that the information contained within the certificate is correct and digitally signs it.

Key Functions of CA

The key functions of a CA are as follows −

- **Generating key pairs** − The CA may generate a key pair independently or jointly with the client.
- **Issuing digital certificates** − The CA could be thought of as the PKI equivalent of a passport agency − the CA issues a certificate after client provides the credentials to confirm his identity. The CA then signs the certificate to prevent modification of the details contained in the certificate.
- **Publishing Certificates** − The CA need to publish certificates so that users can find them. There are two ways of achieving this. One is to publish certificates in the equivalent of an electronic telephone directory. The other is to send your certificate out to those people you think might need it by one means or another.
- **Verifying Certificates** − The CA makes its public key available in environment to assist verification of his signature on clients' digital certificate.
- **Revocation of Certificates** − At times, CA revokes the certificate issued due to some reason such as compromise of private key by user or loss of trust in the client. After revocation, CA maintains the list of all revoked certificate that is available to the environment.

Classes of Certificates

There are four typical classes of certificate −

- **Class 1** − These certificates can be easily acquired by supplying an email address.
- **Class 2** − These certificates require additional personal information to be supplied.
- **Class 3** − These certificates can only be purchased after checks have been made about the requestor's identity.

- **Class 4** – They may be used by governments and financial organizations needing very high levels of trust.

Registration Authority (RA)

CA may use a third-party Registration Authority (RA) to perform the necessary checks on the person or company requesting the certificate to confirm their identity. The RA may appear to the client as a CA, but they do not actually sign the certificate that is issued.

Certificate Management System (CMS)

It is the management system through which certificates are published, temporarily or permanently suspended, renewed, or revoked. Certificate management systems do not normally delete certificates because it may be necessary to prove their status at a point in time, perhaps for legal reasons. A CA along with associated RA runs certificate management systems to be able to track their responsibilities and liabilities.

Private Key Tokens

While the public key of a client is stored on the certificate, the associated secret private key can be stored on the key owner's computer. This method is generally not adopted. If an attacker gains access to the computer, he can easily gain access to private key. For this reason, a private key is stored on secure removable storage token access to which is protected through a password.

Different vendors often use different and sometimes proprietary storage formats for storing keys. For example, Entrust uses the proprietary .epf format, while Verisign, GlobalSign, and Baltimore use the standard .p12 format.

Hierarchy of CA

With vast networks and requirements of global communications, it is practically not feasible to have only one trusted CA from whom all users obtain their certificates. Secondly, availability of only one CA may lead to difficulties if CA is compromised.

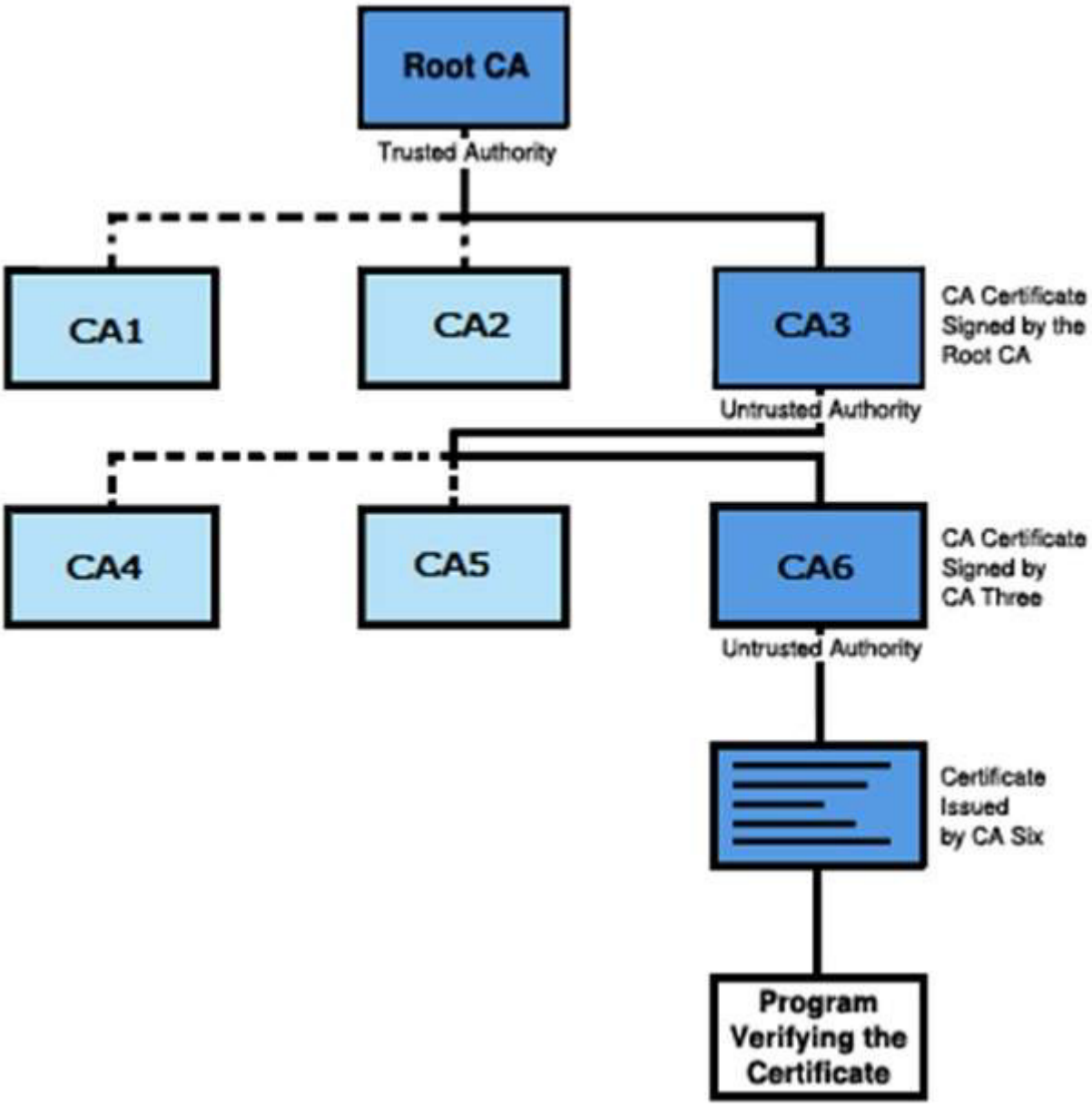
In such case, the hierarchical certification model is of interest since it allows public key certificates to be used in environments where two communicating parties do not have trust relationships with the same CA.

- The root CA is at the top of the CA hierarchy and the root CA's certificate is a self-signed certificate.

- The CAs, which are directly subordinate to the root CA (For example, CA1 and CA2) have CA certificates that are signed by the root CA.
- The CAs under the subordinate CAs in the hierarchy (For example, CA5 and CA6) have their CA certificates signed by the higher-level subordinate CAs.

Certificate authority (CA) hierarchies are reflected in certificate chains. A certificate chain traces a path of certificates from a branch in the hierarchy to the root of the hierarchy.

The following illustration shows a CA hierarchy with a certificate chain leading from an entity certificate through two subordinate CA certificates (CA6 and CA3) to the CA certificate for the root CA.



Verifying a certificate chain is the process of ensuring that a specific certificate chain is valid, correctly signed, and trustworthy. The following procedure verifies a certificate chain, beginning with the certificate that is presented for authentication −

- A client whose authenticity is being verified supplies his certificate, generally along with the chain of certificates up to Root CA.
- Verifier takes the certificate and validates by using public key of issuer. The issuer's public key is found in the issuer's certificate which is in the chain next to client's certificate.

- Now if the higher CA who has signed the issuer's certificate, is trusted by the verifier, verification is successful and stops here.
- Else, the issuer's certificate is verified in a similar manner as done for client in above steps. This process continues till either trusted CA is found in between or else it continues till Root CA.

4.6 Knapsack algorithm

The First General Public-Key Algorithm used what we call the Knapsack Algorithm. Although we now know that this algorithm is not secure we can use it to look at how these types of encryption mechanisms work.

The knapsack algorithm works like this:

Imagine you have a set of different weights which you can use to make any total weight that you need by adding combinations of any of these weights together.

Let us look at an example:

Imagine you had a set of weights 1, 6, 8, 15 and 24. To pack a knapsack weighing 30, you could use weights 1, 6, 8 and 15. It would not be possible to pack a knapsack that weighs 17 but this might not matter.

You might represent the weight 30 by the binary code 11110 (one 1, one 6, one 8, one 15 and no 24).

Example:

Plain text 10011 11010 01011 00000

Knapsack 1 6 8 15 24 1 6 8 15 24 1 6 8 15 24 1 6 8 15 24

Cipher text 1 + 15 + 24 = 40 1 + 6 + 15 = 22 6 + 15 + 24 = 45 0 = 0

What total weights is it possible to make?

So, if someone sends you the code 38 this can only have come from the plain text 01101.

When the Knapsack Algorithm is used in public key cryptography, the idea is to create two different knapsack problems. One is easy to solve, the other not. Using the easy knapsack, the hard knapsack is derived from it. The hard knapsack becomes the public key. The easy knapsack is the private key. The public key can be used to encrypt messages, but cannot be used to decrypt messages. The private key decrypts the messages.

The Superincreasing Knapsack Problem:

An easy knapsack problem is one in which the weights are in a superincreasing sequence. A superincreasing sequence is one in which the next term of the sequence is greater than the sum of all preceding terms. For example, the set {1, 2, 4, 9, 20, 38} is superincreasing, but the set {1, 2, 3, 9, 10, 24} is not because 10 < 1+2+3+9.

It is easy to solve a superincreasing knapsack. Simply take the total weight of the knapsack and compare it with the largest weight in the sequence. If the total weight is less than the number, then it is not in the knapsack. If the total weight is greater than the number, it is in the knapsack. Subtract the number from the total, and compare with the next highest number. Keep working this way until the total reaches zero. If the total doesn't reach zero, then there is no solution.

So, for example, if you have a knapsack that weighs 23 that has been made from the weights of the superincreasing series {1, 2, 4, 9, 20, 38} then it does not contain the weight 38 (as 38 > 23)

but it does contain the weight 20; leaving 3;

which does not contain the weight 9 still leaving 3;

which does not contain the weight 4 still leaving 3;

which contains the weight 2, leaving 1; which contains the weight 1.

The binary code is therefore 110010.

It is much harder to decrypt a non-superincreasing knapsack problem. Give a friend a non-superincreasing knapsack and a total and see why this is the case.

One algorithm that uses a superincreasing knapsack for the private (easy) key and a non-superincreasing knapsack for the public key was created by Merkle and Hellman They did this by taking a superincreasing knapsack problem and converting it into a non-superincreasing one that could be made public, using modulus arithmetic.

Making the Public Key

To produce a normal knapsack sequence, take a superincreasing sequence; e.g. {1, 2, 4, 10, 20, 40}. Multiply all the values by a number, n, modulo m. The modulus should be a number greater than the sum of all the numbers in the sequence, for example, 110. The multiplier should have no

factors in common with the modulus. So let's choose 31. The normal knapsack sequence would be:

1×31 mod(110) = 31

2×31 mod(110) = 62

4×31 mod(110) = 14

10×31 mod(110) = 90

20×31 mod(110) = 70

40×31 mod(110) = 30

So the public key is: {31, 62, 14, 90, 70, 30} and

the private key is {1, 2, 4, 10, 20.40}.

Let's try to send a message that is in binary code:

100100111100101110

The knapsack contains six weights so we need to split the message into groups of six:

100100

111100

101110

This corresponds to three sets of weights with totals as follows

100100 = 31 + 90 = 121

111100 = 31+62+14+90 = 197

101110 = 31+14+90+70 =205

So the coded message is 121 197 205.

Now the receiver has to decode the message...

The person decoding must know the two numbers 110 and 31 (the modulus and the multiplier). Let's call the modulus "m" and the number you multiply by "n".

We need n−1, which is a multiplicative inverse of n mod m, i.e. n(n−1) = 1 mod m

In this case I have calculated n−1 to be 71.

All you then have to do is multiply each of the codes 71 mod 110 to find the total in the knapsack which contains {1, 2, 4, 10, 20, 40} and hence to decode the message.

The coded message is 121 197 205:

121×71 mod(110) = 11 = 100100

197×71 mod(110) = 17 = 111100

205×71 mod(110) = 35 = 101110

The decoded message is:

100100111100101110.

UNIT - V

5.1 INTRUDERS

A significant security problem for networked systems is unauthorized login or use of a system, by local or remote users or by software such as a virus, worm, or Trojan horse. One of the most publicized threats to security is the intruder (or hacker or cracker). Anderson identified three classes of intruders. They are,

• **Masquerader:** An individual who is not authorized to use the computer (outsider)

• **Misfeasor:** A legitimate user who accesses unauthorized data, programs, or resources (Insider)

• **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection (either) Intruder attacks range from the benign (simply exploring net to see what is there); to the serious (who attempt to read privileged data, perform unauthorized modifications, or disrupt system)

5.2 INTRUSION TECHNIQUES

- Aim to gain access and/or increase privileges on a system
- Basic attack methodology
    - Target acquisition and information gathering
    - Initial access
    - Privilege escalation
    - Covering tracks
- Key goal often is to acquire passwords
- Exercise access rights of owner

A basic technique for gaining access is to acquire a user (preferably administrator) password, so the attacker can login and exercise all the access rights of the account owner.

Password Guessing

- One of the most common attacks
- Attacker knows a login (from email/web page etc)
- Attempts to guess password for it
    - defaults, short passwords, common word searches

- user info (variations on names, birthday, phone, common words/interests)
- exhaustively searching all possible passwords

- Check by login or against stolen password file
- Success depends on password chosen by user
- Surveys show many users choose poorly

Password Capture

- Another attack involves **password capture**
  - Watching over shoulder as password is entered
  - Using a Trojan horse program to collect
  - Monitoring an insecure network login
  - eg. telnet, FTP, web, email
  - Extracting recorded info after successful login (web history/cache, last number dialled etc)
- Using valid login/password can impersonate user
- Users need to be educated to use suitable precautions/countermeasures

There is also a range of ways of "capturing" a login/password pair, from the low-tech looking over the shoulder, to the use of Trojan Horse programs (eg. game program or nifty utility with a covert function as well as the overt behaviour), to sophisticated network monitoring tools, or extracting recorded info after a successful login - say from web history or cache, or last number dialled memory on phones etc. Need to educate users to be aware of whose around, to check they really are interacting with the computer system (trusted path), to beware of unknown source s/w, to use secure network connections (HTTPS, SSH, SSL), to flush browser/phone histories after use etc.

5.3 INTRUSION DETECTION

- unavoidably will have security failures
- So need to detect intrusions. Then
  - block if detected quickly
  - act as prevention
  - collect info to improve security
- Assume intruder will behave differently to a legal user

Approaches to Intrusion Detection

1. Statistical anomaly detection: collect data relating to the behavior of legitimate users, then use statistical tests to determine with a high level of confidence whether new behavior is legitimate user behavior or not.

a. Threshold detection: Define thresholds, independent of user, for the frequency of occurrence of events.

b. Profile based: Develop profile of activity of each user and use to detect changes in the behavior.

The main advantage of the use of statistical profiles is that a prior knowledge of security flaws is not required. Thus it should be readily portable among a variety of systems.

2. Rule-based detection: Attempt to define a set of rules used to decide if given behavior is an intruder.

a. Anomaly detection: Rules detect deviation from previous usage patterns

b. Penetration identification: Expert system approach that searches for suspicious behavior

Audit Records

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

- **Native audit records:** Virtually all main O/S's include accounting software that collects information on user activity, advantage is its already there, disadvantage is it may not contain the needed information

- **Detection-specific audit records:** implement collection facility to generate custom audit records with desired info, advantage is it can be vendor independent and portable, disadvantage is extra overhead involved.

Base-Rate Fallacy

To be of practical use, an intrusion detection system should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms, or much time will be

wasted analyzing the false alarms. Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. A study of existing intrusion detection systems indicated that current systems have not overcome the problem of the base-rate fallacy.

5.4 VIRUSES AND RELATED THREATS

- A piece of self-replicating code attached to some other code
- Both propagates itself & carries a payload
  - Carries code to make copies of itself
  - As well as code to perform some covert task

Virus Operation

During its lifetime, a typical virus goes through the following four phases:

- **Dormant phase:** Virus is idle, waiting for trigger event (eg date, program or file, disk capacity). Not all viruses have this stage.
- **Propagation phase:** Virus places a copy of itself into other programs / system areas.
- **Triggering phase:** Virus is activated by some trigger event to perform intended function.
- **Execution phase:** Desired function (which may be harmless or destructive) is performed.

Most viruses work in a manner specific to a particular operating system or even hardware platform, and are designed to take advantage of the details and weaknesses of particular systems.

Virus Structure

```
Program V: =
        {
goto main;
                1234567;
                subroutine infect-executable: =
{
loop:
                                file := get-random-executable-file;
                                if (first-line-of-file = 1234567) then goto loop
                                else prepend V to file;
}
                subroutine do-damage: =  {whatever damage is to be done}
```

```
        subroutine trigger-pulled: = {return true if condition holds}
        main: main-program: =
                {
infect-executable;
                        if trigger-pulled then do-damage;
                        goto next;
}
        next:
}
```

The first line of code jumps to the main virus program. The second line is a special marker for infected programs. The main virus program first seeks out uninfected executable files and infects them. Then it may perform some action, usually detrimental to the system, depending on some trigger. Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program. This type of virus can be detected because the length of the program changes. More sophisticated variants attempt to hide their presence better, by for example, compressing the original program.

Types of Viruses

There has been a continuous arms race between virus writers and writers of antivirus software, with the following categories being among the most significant types of viruses:

- **Parasitic virus:** Traditional and still most common form of virus, it attaches itself to executable files and replicates when the infected program is executed

- **Memory-resident virus:** Lodges in main memory as part of a resident system program, and infects every program that executes

- **Boot sector virus:** Infects a master boot record and spreads when a system is booted from the disk containing the virus

- **Stealth virus:** A virus explicitly designed to hide itself from detection by antivirus software

- **Polymorphic virus:** Mutates with every infection, making detection by the “signature” of the virus impossible.

• **Metamorphic virus:** Mutates with every infection, rewriting itself completely at each iteration changing behavior and/or appearance, increasing the difficulty of detection.

5.5 WORMS

- Replicating but not infecting program
- Typically spreads over a network
  - Morris Internet Worm in 1988
  - Led to creation of CERTs
- Using users distributed privileges or by exploiting system vulnerabilities
- Widely used by hackers to create **zombie PC's**, subsequently used for further attacks, esp DoS
- Major issue is lack of security of permanently connected systems, esp. PC's

A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again, and usually to also perform some unwanted function. A worm actively seeks out more machines to infect and each machine that is infected serves as an automated launching pad for attacks on other machines. To replicate itself, a network worm uses some sort of network vehicle such as email, remote execution, or remote login. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.

Worm Operation

A network worm exhibits the same characteristics as a computer virus: a dormant phase, a propagation phase, a triggering phase, and an execution phase. The propagation phase generally:

1) Searches for other systems to infect by examining host tables etc
2) Establishes a connection with a remote system
3) Copies itself to the remote system and cause the copy to be run.

Worm Technology

The state of the art in worm technology includes the following:

- Multiplatform: not limited to Windows, can attack a variety of O/S's, esp UNIX.
- Multiexploit: penetrate systems in a variety of ways

- Ultra fast spreading: using prior scan to get addresses of vulnerable machines
- Polymorphic: adopt virus polymorphic technique to evade detection
- Metamorphic: change both appearance & behavior patterns
- Transport vehicles: to spread other distributed attack tools, eg zombies
- Zero-day exploit: exploit unknown vulnerability

5.6 FIREWALLS

A firewall is inserted between the premises network and the Internet to establish a controlled link and to raise an outer security wall or perimeter, forming a single choke point where security and audit can be imposed. A firewall:

1) Defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.
2) Provides a location for monitoring security-related events
3) Is a convenient platform for several Internet functions that are not security related,
4) Such as NAT and Internet usage audits or logs.
5) A firewall can serve as the platform for IPSec to implement virtual private
6) Networks.

The firewall itself must be immune to penetration, since it will be a target of attack.

Firewall Limitations

1) Cannot protect against attacks that bypass the firewall, eg PCs with dial-out capability to an ISP, or dial-in modem pool use
2) Do not protect against internal threats, eg disgruntled employee or one who cooperates with an attacker
3) Cannot protect against the transfer of virus-infected programs or files, given wide variety of O/S & applications supported

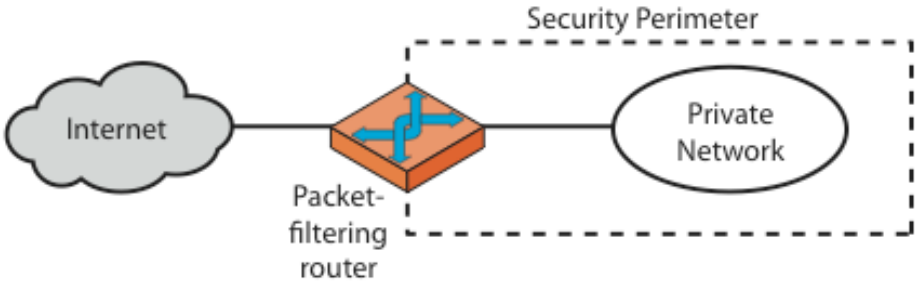
Three common types of firewalls

1) Packet filters
2) Application-level gateways
3) Circuit-level gateways.

Firewalls – Packet Filters

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet to forward or discard the packet. Filtering rules are based on information contained in a network packet such as src & dest IP addresses ports, transport protocol & interface. Some advantages are simplicity, transparency & speed.If there is no match to any rule, then one of two default policies are applied:

- That which is not expressly permitted is prohibited (default action is discard packet), conservative policy
- That which is not expressly prohibited is permitted (default action is forward packet), permissive policy



(a) Packet-filtering router

The packet filter firewall placement in the border router, on the security perimeter, between the external less-trusted Internet, and the internal more trusted private network.

Attacks on Packet Filters

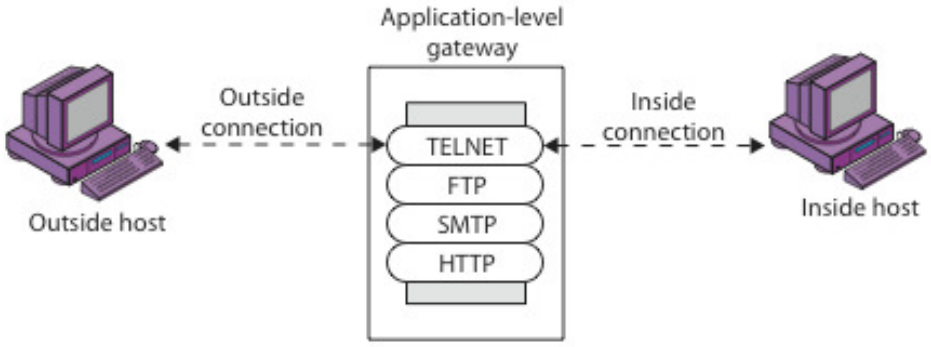
• **IP address spoofing:** where intruder transmits packets from the outside with internal host source IP addr, need to filter & discard such packets

• **Source routing attacks:** where source specifies the route that a packet should take to bypass security measures, should discard all source routed packets

• **Tiny fragment attacks:** intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate fragment to circumvent filtering rules needing full header info, can enforce minimum fragment size to include full header.

Firewalls - Application Level Gateway (or Proxy)

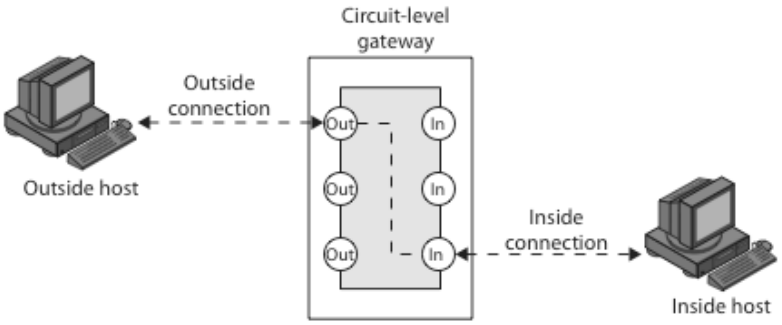
An application-level gateway (or proxy server), acts as a relay of application-level traffic. A user contacts the gateway to access some service, provides details of the service, remote host & authentication details, contacts the application on the remote host and relays all data between the two endpoints. If the gateway does not implement the proxy code for a specific application, then it is not supported and cannot be used. Note that some services naturally support proxying, whilst others are more problematic. Application-level gateways tend to be more secure than packet filters, &can log and audit traffic at application level.



(b) Application-level gateway

Firewalls - Circuit Level Gateway

A circuit-level gateway relays two TCP connections, one between itself and an inside TCP user, and the other between itself and a TCP user on an outside host. Once the two connections are established, it relays TCP data from one connection to the other without examining its contents. The security function consists of determining which connections will be allowed. It is typically used when internal users are trusted to decide what external services to access. One of the most common circuit-level gateways is SOCKS, defined in RFC 1928. It consists of a SOCKS server on the firewall, and a SOCKS library & SOCKS-aware applications on internal clients.



(c) Circuit-level gateway

5.7 EMAIL SECURITY

E-mail is one of the most widely used and regarded network services

Currently message contents are not secure

May be inspected either in transit

Or by suitably privileged users on destination system

In virtually all distributed environments, electronic mail is the most heavily used network-based application. But current email services are roughly like "postcards", anyone who wants could pick it up and have a look as its in transit or sitting in the recipients mailbox.

Email Security Enhancements

confidentiality

protection from disclosure

authentication

sender of message

message integrity

protection from modification

non-repudiation of origin

protection from denial by sender

With the explosively growing reliance on electronic mail for every conceivable purpose, there grows a demand for authentication and confidentiality services. What we want is something more akin to standard mail (contents protected inside an envelope) if not registered mail (have confidence about the sender of the mail and its contents). That is, the "classic" security services listed are desired.

5.8 PRETTY GOOD PRIVACY (PGP)

Widely used de facto secure email

Developed by Phil Zimmermann

Selected best available crypto algorithms to use

Integrated into a single program

On Unix, PC, Macintosh and other systems

Originally free, now also have commercial versions available

The Pretty Good Privacy (PGP) secure email program, is a remarkable phenomenon, has grown explosively and is now widely used. Largely the effort of a single person, Phil Zimmermann, who selected the best available crypto algorithms to use & integrated them into a single program, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. It runs on a wide range of systems, in both free & commercial versions.

PGP Operation – Authentication

Sender creates message

Use SHA-1 to generate 160-bit hash of message

Signed hash with RSA using sender's private key, and is attached to message

Receiver uses RSA with sender's public key to decrypt and recover hash code

Receiver verifies received message using hash of it and compares with decrypted hash code

The actual operation of PGP consists of five services: authentication, confidentiality, compression, e-mail compatibility, and segmentation. Here see the digital signature service provided by PGP, using the steps as shown. Note this assumes use of RSA digital signatures; recent PGP versions also support the use of DSS signatures. Signatures can also be detached from a message/file and sent/stored separately.

PGP Operation – Confidentiality

Sender generates message and 128-bit random number as session key for it

Encrypt message using CAST-128 / IDEA / 3DES in CBC mode with session key

Session key encrypted using RSA with recipient's public key, & attached to message

Receiver uses RSA with private key to decrypt and recover session key

Session key is used to decrypt message

Another basic service provided by PGP is confidentiality, provided by encrypting messages to be transmitted or to be stored locally as files, using symmetric encryption algorithms CAST-128, IDEA or 3DES in 64-bit cipher feedback (CFB) mode. The randomly chosen session key used for this is sent encrypted using the recipient's public RSA key. The steps used in this process are as shown. Recent PGP versions also support the use of ElGamal (a Diffie-Hellman variant) for session-key exchange.

PGP Operation – Confidentiality & Authentication

Can use both services on same message

Create signature & attach to message

Encrypt both message & signature

Attach RSA/ElGamal encrypted session key

Both confidentiality & authentication services may be used for the same message. Firstly a signature is generated for the plaintext message and prepended to the it. Then the plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal).

PGP Operation – Compression

by default PGP compresses message after signing but before encrypting

so can store uncompressed message & signature for later verification

& because compression is non deterministic

uses ZIP compression algorithm

By default PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage. The signature is generated before compression for the reasons shown. The compression algorithm used is ZIP.

PGP Operation – Email Compatibility

When using PGP will have binary data to send (encrypted message etc)

However email was designed only for text

Hence PGP must encode raw binary data into printable ASCII characters

Uses radix-64 algorithm

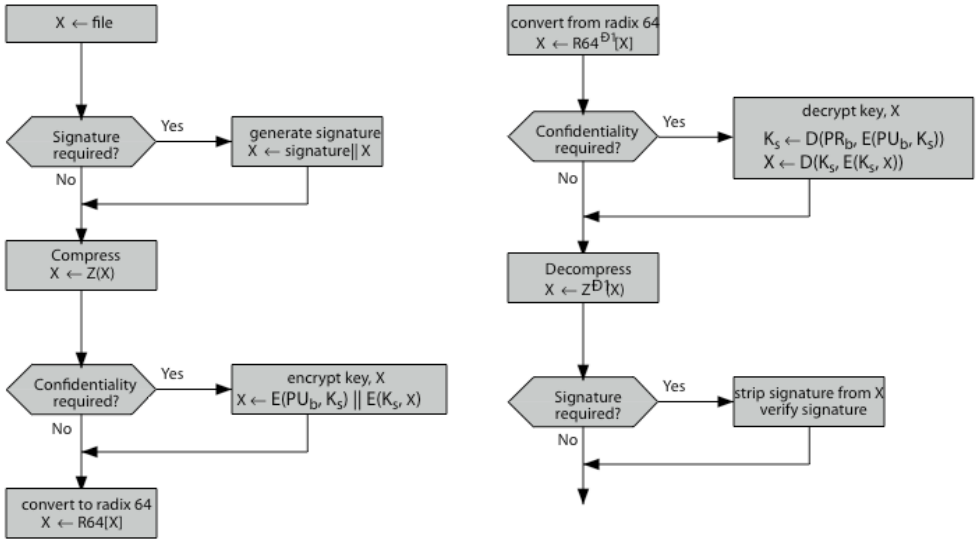
maps 3 bytes to 4 printable chars

also appends a CRC

PGP also segments messages if too big

When PGP is used, at least part of the block to be transmitted is encrypted, and thus consists of a stream of arbitrary 8-bit octets. However many electronic mail systems only permit the use of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. It uses radix-64 conversion, in which each group of three octets of binary data is mapped into four ASCII characters. This format also appends a CRC to detect transmission errors. PGP also automatically subdivides a message that is too large for a single email, into segments that are small enough to send.

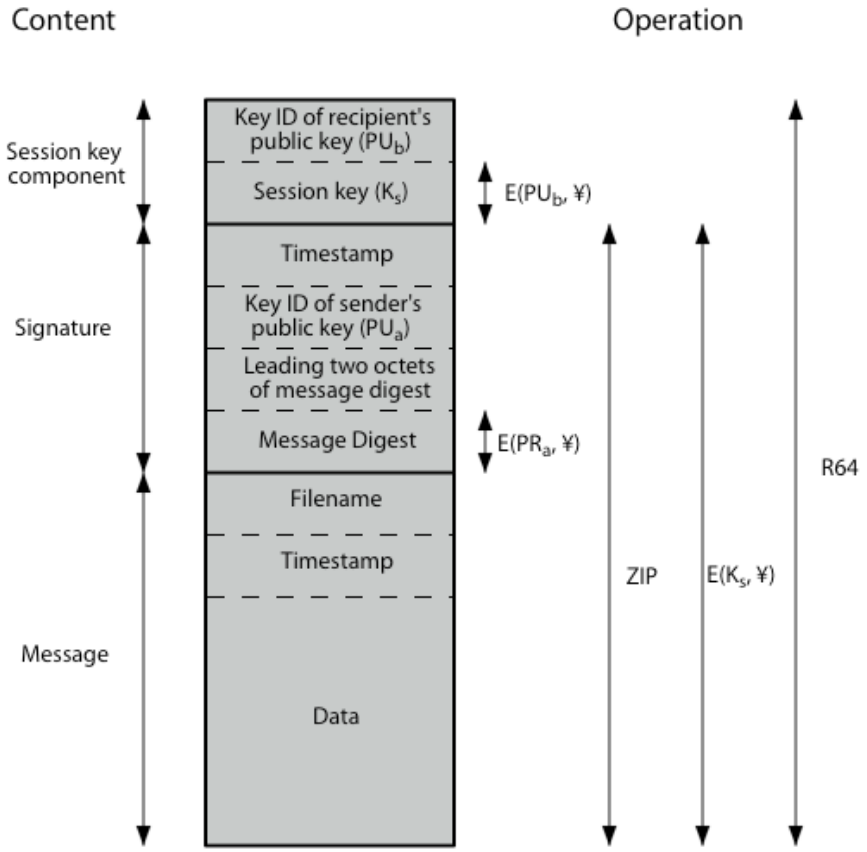
PGP Operation – Summary



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

PGP Message Format



A message consists of three components: The message component, A signature (optional), and A session key component (optional).

5.9 S/MIME (SECURE/MULTIPURPOSE INTERNET MAIL EXTENSIONS)

Security enhancement to MIME email.

Original Internet RFC822 email was text only

MIME provided support for varying content types and multi-part messages

With encoding of binary data to textual form

S/MIME added security enhancements

Have S/MIME support in many mail agents

eg MS Outlook, Mozilla, Mac Mail etc

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, which in turn provided support for varying content types and multi-part messages over the text only support in the original Internet RFC822 email standard. MIME allows

encoding of binary data to textual form for transport over traditional RFC822 email systems. S/MIME support is now included in many modern mail agents.

S/MIME Functions

Enveloped data
encrypted content and associated keys
Signed data
encoded message + signed digest
Clear-signed data
cleartext message + encoded signed digest
Signed & enveloped data
nesting of signed & encrypted entities

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages. S/MIME provides the functions shown.

S/MIME Cryptographic Algorithms

Digital signatures: DSS & RSA
Hash functions: SHA-1 & MD5
Session key encryption: ElGamal & RSA
Message encryption: AES, Triple-DES, RC2/40 and others
MAC: HMAC with SHA-1
Have process to decide which algorithms to use

S/MIME Messages

S/MIME secures a MIME entity with a signature, encryption, or both
Forming a MIME wrapped PKCS object
Have a range of content-types:
enveloped data
signed data
clear-signed data
registration request
certificate only message

S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message or one or more of the subparts of the message. The MIME entity plus some security related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce a PKCS, which refers to a set of public-key cryptography specifications issued by RSA

Laboratories. A PKCS object is then treated as message content and wrapped in MIME.

S/MIME Certificate Processing

S/MIME uses X.509 v3 certificates

Managed using a hybrid of a strict X.509 CA hierarchy & PGP's web of trust

Each client has a list of trusted CA's certs

Own public/private key pairs & certs

Certificates must be signed by trusted CA's

S/MIME uses public-key certificates that conform to version 3 of X.509 (see Chapter 14). The key-management scheme used by S/MIME is in some ways a hybrid between a strict X.509 certification hierarchy and PGP's web of trust. S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists, needed to verify incoming signatures and to encrypt outgoing messages. But certificates are signed by trusted certification authorities.

5.10 IP SECURITY

Have a range of application specific security mechanisms

eg. S/MIME, PGP, Kerberos, SSL/HTTPS

However there are security concerns that cut across protocol layers

Would like security implemented by the network for all applications

The Internet community has developed application-specific security mechanisms in a number of application areas, including electronic mail (S/MIME, PGP), client/server (Kerberos), Web access (Secure Sockets Layer), and others. However users have some security concerns that cut across protocol layers. By implementing security at the IP level, an organization can ensure secure networking not only for applications that have security mechanisms but also for the many security-ignorant applications.

IPSec

General IP Security mechanisms

Provides

authentication

confidentiality

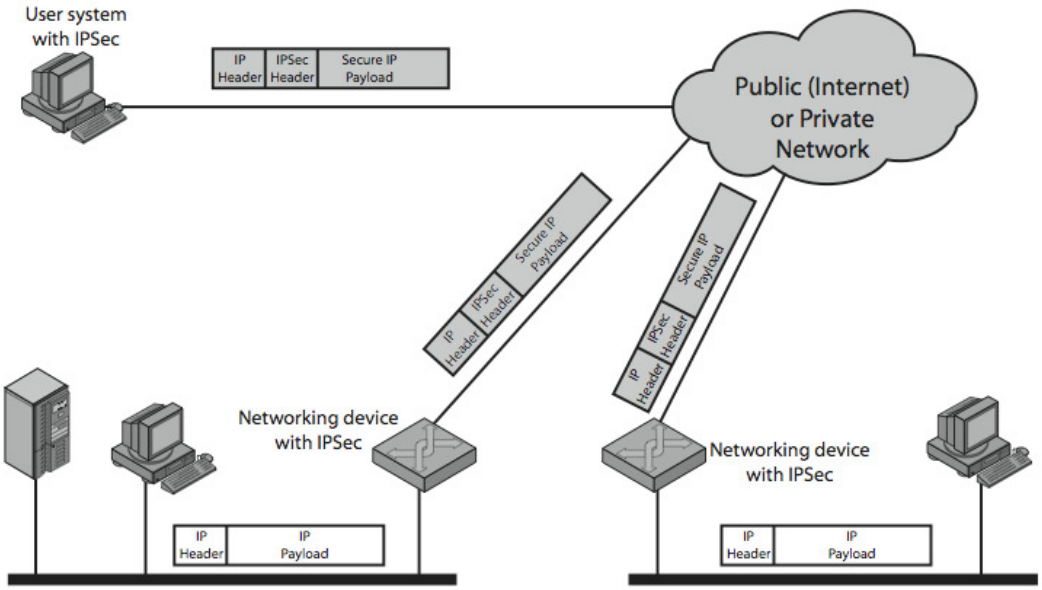
key management

Applicable to use over LANs, across public & private WANs, & for the Internet

IP-level security encompasses three functional areas: authentication, confidentiality, and key management. The authentication mechanism assures that a received packet was transmitted by the party identified as the source in the packet header, and that the packet has not been altered in transit. The confidentiality facility enables communicating nodes to encrypt messages to prevent eavesdropping by third parties. The key management facility is concerned with the secure exchange of keys. IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet.

IPSec Uses

An organization maintains LANs at dispersed locations. Nonsecure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPSec protocols are used. These protocols operate in networking devices, such as a router or firewall that connect each LAN to the outside world. The IPSec networking device will typically encrypt and compress all traffic going into the WAN, and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPSec protocols to provide security.



Benefits of IPSec

in a firewall/router provides strong security to all traffic crossing the perimeter
in a firewall/router is resistant to bypass
is below transport layer, hence transparent to applications
can be transparent to end users
can provide security for individual users
secures routing architecture

IP Security Architecture

specification is quite complex
defined in numerous RFC's
incl. RFC 2401/2402/2406/2408
many others, grouped by category
mandatory in IPv6, optional in IPv4
have two security header extensions:
Authentication Header (AH)
Encapsulating Security Payload (ESP)

The IPSec specification has become quite complex. The IPSec specification consists of numerous documents. The most important of these, issued in November of 1998, are

RFC 2401: An overview of security architecture
RFC 2402: Description of a packet authentication extension to IPv4 and IPv6
RFC 2406: Description of a packet encryption extension to IPv4 and IPv6
RFC 2408: Specification of key management capabilities

In addition to these four RFCs, a number of additional drafts have been published by the IP Security Protocol Working Group set up by the IETF. The documents are divided into seven groups. Support for these features is mandatory for IPv6 and optional for IPv4. In both cases, the security features are implemented as extension headers that follow the main IP header. The extension header for authentication is known as the Authentication Header (AH); that for encryption is known as the Encapsulating Security Payload (ESP) header.

IPSec Services

Access control

Connectionless integrity

Data origin authentication

Rejection of replayed packets

a form of partial sequence integrity

Confidentiality (encryption)

Limited traffic flow confidentiality

IPSec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. The security services supported are as shown above. See Stallings Table 16.1 for the services provided by AH & ESP respectively. For ESP, there are two cases: with and without the authentication option. Both AH and ESP are vehicles for access control, based on the distribution of cryptographic keys and the management of traffic flows relative to these security protocols.

Security Associations

a one-way relationship between sender & receiver that affords security for traffic flow

defined by 3 parameters:

Security Parameters Index (SPI)

IP Destination Address

Security Protocol Identifier

has a number of other parameters

seq no, AH & EH info, lifetime etc

have a database of Security Associations

A key concept that appears in both the authentication and confidentiality mechanisms for IP is the security association (SA). An association is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed, for two-way secure exchange, then two security associations are required. Security services are afforded to an SA for the use of AH or ESP, but not both. A security association is uniquely identified by three parameters:

Security Parameters Index (SPI): A bit string assigned to this SA and having local significance only

IP Destination Address: this is the address of the destination endpoint of the SA
Security Protocol Identifier: This indicates whether the association is an AH or ESP security association.

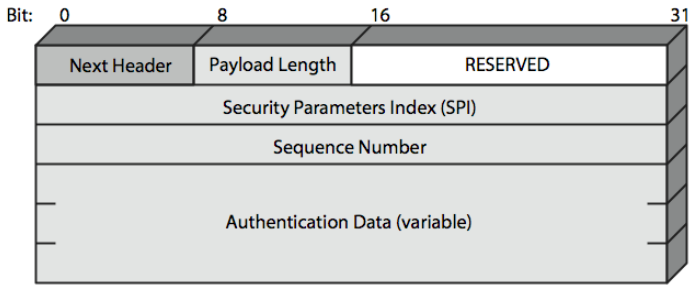
A SA may also have a number of other parameters. In each IPSec implementation, there is a Security Association Database that defines the parameters associated with each SA.

Authentication Header (AH)

provides support for data integrity & authentication of IP packets
end system/router can authenticate user/app
prevents address spoofing attacks by tracking sequence numbers
based on use of a MAC
HMAC-MD5-96 or HMAC-SHA-1-96
parties must share a secret key

The Authentication Header provides support for data integrity and authentication of IP packets. The data integrity feature ensures that undetected modification to a packet's content in transit is not possible. The authentication feature enables an end system or network device to authenticate the user or application and filter traffic accordingly; it also prevents address spoofing attacks and replay attacks. Authentication is based on the use of a message authentication code (MAC), hence the two parties must share a secret key. AH supports MACs using HMAC-MD5-96 or HMAC-SHA-1-96. Both of these use the HMAC algorithm, the first with the MD5 hash code and the second with the SHA-1 hash code. In both cases, the full HMAC value is calculated but then truncated by using the first 96bits, which is the default length for the Authentication Data field.

Authentication Header



Next Header (8 bits): Identifies the type of header immediately following this header

Payload Length (8 bits): Length of Authentication Header in 32-bit words, minus 2.

Reserved (16 bits): For future use

Security Parameters Index (32 bits): Identifies a security association

Sequence Number (32 bits): A monotonically increasing counter value

Authentication Data (variable): A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC,for this packet

Transport & Tunnel Modes

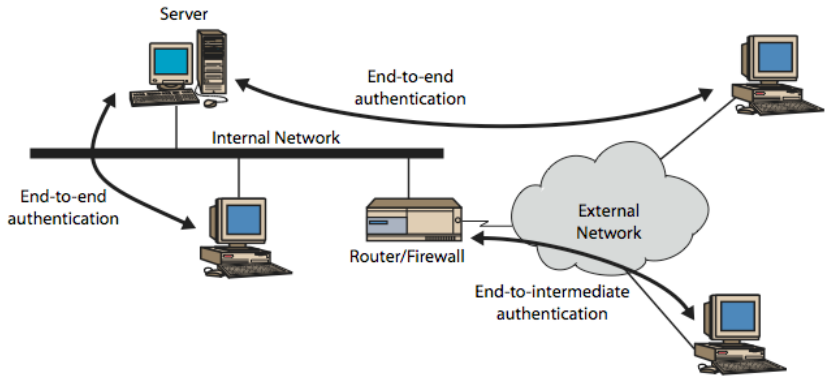


Figure shows the difference between end-to-end (transport) mode and end-to-intermediate (tunnel) mode.

Transport mode provides protection primarily for upper-layer protocol payloads, by inserting the AH after the original IP header and before the IP payload. Typically, transport mode is used for end-to-end communication between two hosts.

Tunnel mode provides protection to the entire IP, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new “outer”IP packet with a new outer IP header. Tunnel mode is used when one or both ends of an SA are a security gateway, such as a firewall or router that implements IPSec.

Encapsulating Security Payload (ESP)

provides message content confidentiality & limited traffic flow confidentiality

can optionally provide the same authentication services as AH

supports range of ciphers, modes, padding

incl. DES, Triple-DES, RC5, IDEA, CAST etc

CBC & other modes

padding needed to fill blocksize, fields, for traffic flow

The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service, with the same MACs as AH. ESP supports range of ciphers, modes, and padding, as shown.

Encapsulating Security Payload

It contains the following fields:

Security Parameters Index (32 bits): Identifies a security association

Sequence Number (32 bits): A monotonically increasing counter value; this provides an anti-replay function ,as discussed for AH

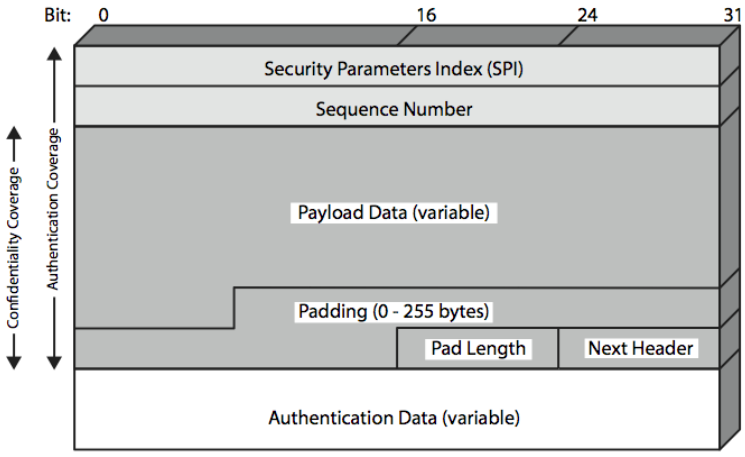
Payload Data (variable): This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption

Padding (0–255 bytes): for various reasons

Pad Length (8 bits): Indicates the number of pad bytes immediately preceding this field

Next Header (8 bits): Identifies the type of data contained in the payload data field by identifying the first header in that payload

Authentication Data (variable): A variable-length field that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field



5.11 WEB SECURITY

Web now widely used by business, government, individuals but Internet & Web are vulnerable have a variety of threats integrity confidentiality denial of service Authentication need added security mechanisms

The World Wide Web is widely used by businesses, government agencies, and many individuals. But the Internet and the Web are extremely vulnerable to compromises of various sorts, with a range of threats as shown. These can be described as passive attacks including eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted, and active attacks including impersonating another user, altering messages in transit between client and server, and altering information on a Web site. The web needs added security mechanisms to address these threats.

SSL (Secure Socket Layer)

transport layer security service

originally developed by Netscape

version 3 designed with public input

subsequently became Internet standard known as TLS (Transport Layer Security)

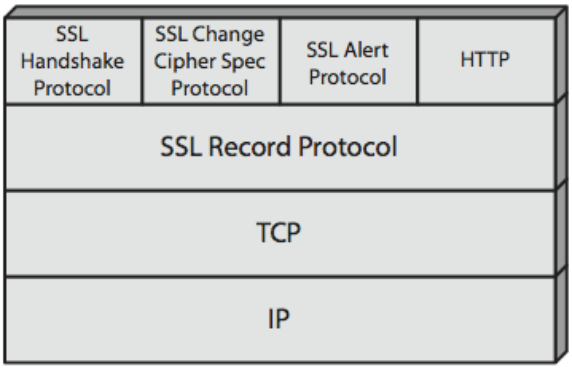
uses TCP to provide a reliable end-to-end service

SSL has two layers of protocols

SSL probably most widely used Web security mechanism. Its implemented at the Transport layer; cf IPSec at Network layer; or various Application layer mechanisms eg. S/MIME & SET (later).

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. Netscape originated SSL. Version 3 of the protocol was designed with public review and input from industry and was published as an Internet draft document. Subsequently, the IETF TLS working group was formed to develop a common standard. SSL is not a single protocol but rather two layers of protocol, as shown next.

SSL Architecture



The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are also defined as part of SSL: the Handshake Protocol, Change Cipher Spec Protocol, and Alert Protocol. These SSL-specific protocols are used in the management of SSL exchanges.

SSL Architecture

SSL connection

a transient, peer-to-peer, communications link

associated with 1 SSL session

SSL session

an association between client & server

created by the Handshake Protocol

define a set of cryptographic parameters

may be shared by multiple SSL connections

Two important SSL concepts are the SSL connection and the SSL session:

Connection: A connection is a network transport that provides a suitable type of service, such connections are transient, peer-to-peer relationships, associated with one session

Session: An SSL session is an association between a client and a server, created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

SSL Record Protocol Services

message integrity

using a MAC with shared secret key

similar to HMAC but with different padding

confidentiality

using symmetric encryption with a shared secret key defined by Handshake Protocol

AES, IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128

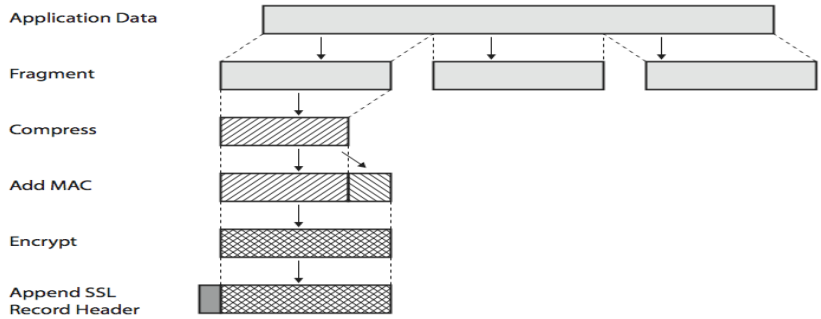
message is compressed before encryption

SSL Record Protocol defines two services for SSL connections:

Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC), which is similar to HMAC

Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. The message is compressed before being concatenated with the MAC and encrypted, with a range of ciphers being supported as shown.

SSL Record Protocol Operation



The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled and then delivered to higher-layer applications.

SSL Change Cipher Spec Protocol

one of 3 SSL specific protocols which use the SSL Record protocol

a single message

causes pending state to become current

hence updating the cipher suite in use

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest, consisting of a single message. Its purpose is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

SSL Alert Protocol

conveys SSL-related alerts to peer entity

severity

warning or fatal

specific alert

fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter

warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown

compressed & encrypted like all SSL data

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol consists of two bytes; the first takes the value warning (1) or fatal (2) to convey the severity of the message. The second byte contains a code that indicates the specific alert. The first group shown is the fatal alerts, the others are warnings.

SSL Handshake Protocol

allows server & client to:

authenticate each other

to negotiate encryption & MAC algorithms

to negotiate cryptographic keys to be used

comprises a series of messages in phases

Establish Security Capabilities

Server Authentication and Key Exchange

Client Authentication and Key Exchange

Finish

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by client and server, which can be viewed in 4 phases:

Phase 1. Establish Security Capabilities - this phase is used by the client to initiate a logical connection and to establish the security capabilities that will be associated with it

Phase 2. Server Authentication and Key Exchange - the server begins this phase by sending its certificate if it needs to be authenticated.

Phase 3. Client Authentication and Key Exchange - the client should verify that the server provided a valid certificate if required and check that the server_hello parameters are acceptable

Phase 4. Finish - this phase completes the setting up of a secure connection. The client sends a change_cipher_spec message and copies the pending Cipher Spec into the current Cipher Spec

TLS (Transport Layer Security)

IETF standard RFC 2246 similar to SSLv3

with minor differences

in record format version number

uses HMAC for MAC

a pseudo-random function expands secrets

has additional alert codes

some changes in supported ciphers

changes in certificate types & negotiations

changes in crypto computations & padding

TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL. TLS is defined as a Proposed Internet Standard in RFC 2246. RFC 2246 is very similar to SSLv3, but with a number of minor differences.

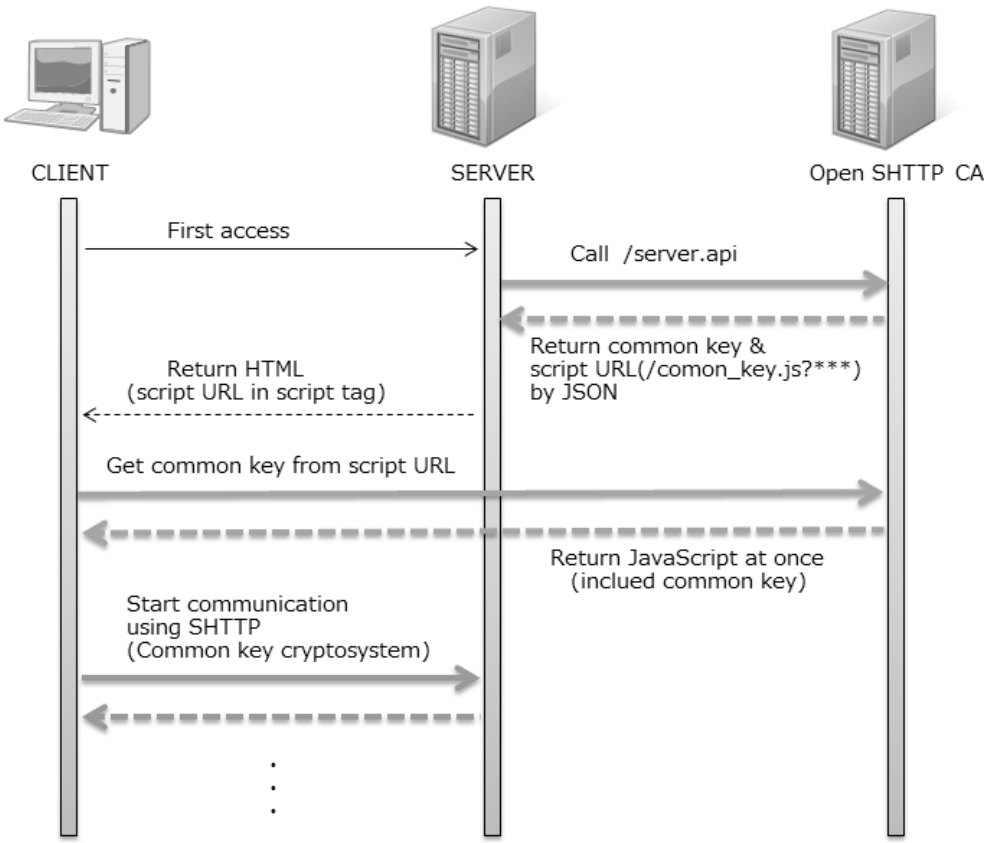
Secure Hypertext Transfer Protocol

Introduction:

Secure HTTP provides secure communication mechanisms between an HTTP client-server pair in order to enable spontaneous commercial transactions for a wide range of applications. It is to provide a flexible protocol that supports multiple orthogonal operation modes, key management mechanisms, trust models, cryptographic algorithms and encapsulation formats through option negotiation between parties for each transaction.

Secure HTTP is a secure message-oriented communications protocol designed for use in conjunction with HTTP. It is designed to coexist with HTTP's messaging model and to be easily integrated with HTTP applications. Secure HTTP provides a variety of security mechanisms to HTTP clients and servers, providing the security service options appropriate to the wide range of potential end uses possible for the World-Wide Web. The protocol provides symmetric capabilities to both client and server (in that equal treatment is given to both requests and replies, as well as for the preferences of both parties) while preserving the transaction model and implementation characteristics of HTTP.

S-HTTP provides broad support for implementing different types of cryptographic algorithms and key management systems. Although S-HTTP systems can make use of digital certificates and public keys, messages can also be encrypted on a per-transaction basis using symmetric session keys.



Message Preparation:

The creation of an S-HTTP message can be thought of as a function with three inputs:

1. The cleartext message. This is either an HTTP message or some other data object. Note that since the cleartext message is carried transparently, headers and all, any version of HTTP can be carried within an S-HTTP wrapper.
2. The receiver's cryptographic preferences and keying material. This is either explicitly specified by the receiver or subject to some default set of preferences.
3. The sender's cryptographic preferences and keying material. This input to the function can be thought of as implicit since it exists only in the memory of the sender.

In order to create an S-HTTP message, then, the sender integrates the sender's preferences with the receiver's preferences. The result of this is a list of cryptographic enhancements to be applied and keying material to be used to apply them.

Message Recovery:

The recovery of an S-HTTP message can be thought of as a function of four distinct inputs:

1. The S-HTTP message.
2. The receiver's stated cryptographic preferences and keying material. The receiver has the opportunity to remember what cryptographic preferences it provided in order for this document to be dereferenced.
3. The receiver's current cryptographic preferences and keying material.
4. The sender's previously stated cryptographic options. The sender may have stated that he would perform certain cryptographic operations in this message.

In order to recover an S-HTTP message, the receiver needs to read the headers to discover which cryptographic transformations were performed on the message, then remove the transformations using some combination of the sender's and receiver's keying material, while taking note of which enhancements were applied.

Modes of Operation:

Message protection may be provided on three orthogonal axes: signature, authentication, and encryption. Any message may be signed, authenticated, encrypted, or any combination of these (including no protection). Multiple key management mechanisms are supported, including password-style manually shared secrets and public-key key exchange. In particular, provision has been made for prearranged (in an earlier transaction or out of band) symmetric session keys in order to send confidential messages to those who have no public key pair. Additionally, a challenge-response mechanism is provided to allow parties to assure themselves of transaction freshness.

1. **Signature:** If the digital signature enhancement is applied, an appropriate certificate may either be attached to the message (possibly along with a certificate chain) or the sender may expect the recipient to obtain the required certificate (chain) independently.
2. **Key Exchange and Encryption:** S-HTTP defines two key transfer mechanisms, one using public-key enveloped key exchange and another with externally arranged keys. In the former case, the symmetric-key cryptosystem parameter is passed encrypted under the receiver's public key.
3. **Message Integrity and Sender Authentication:** Secure HTTP provides a means to verify message integrity and sender authenticity for a message via the computation of a Message Authentication Code (MAC). This mechanism is also useful for cases where it is appropriate to allow parties to identify each other reliably in a transaction without providing (third-party) non-repudiability for the transactions themselves.

Summary:

1. Secure Hypertext Transfer Protocol (S-HTTP) uses Cryptographic Message Syntax (CMS) Algorithm to secure each individual message. Since each message is secured separately each message can have different level or type of security depending on the need. Each web page that uses S-HTTP has the required information incorporated into the pages HTML defining its cryptographic level and any needed information (keys, etc.). When the browser dereferences the link and sends a request it is securely encrypted using the method defined.
2. Works identically to HTTP, except that the request and response messages are transmitted using SSL (Secure Sockets Layer) or its successor TLS (Transport Layer Security). HTTPS is used automatically for any URL beginning with "https:" instead of "http:"
3. The request and response messages are transmitted between the browser and server in encrypted form. This prevents snoopers on the network from accessing private information in the messages, such as passwords or credit card numbers. A certificate exchange allows the browser to identify the server it is communicating with. HTTPS doesn't enable the server to identify the browser.
4. HTTPS does not guarantee that the browser and server can trust each other. You just know that no-one else is listening. HTTPS requires additional server setup: must create a certificate that identifies the server to the browser. In designing Web applications you must use HTTPS whenever private data is being transmitted, such as passwords or credit card numbers.

Security in 3G

Introduction:

Communications on shared media like radio communication are no longer private. Privacy and authentication are lost unless some method is established to regain it. Cryptography provides the solution to regain control over privacy and authentication. All digital mobile systems provide security through some kind of encryption system. Data can be encrypted in many ways, but algorithms used for secure data transfer fall into one of the two broad categories: Symmetric and Asymmetric. Both rely on performing mathematical operations using a secret number known as a key.

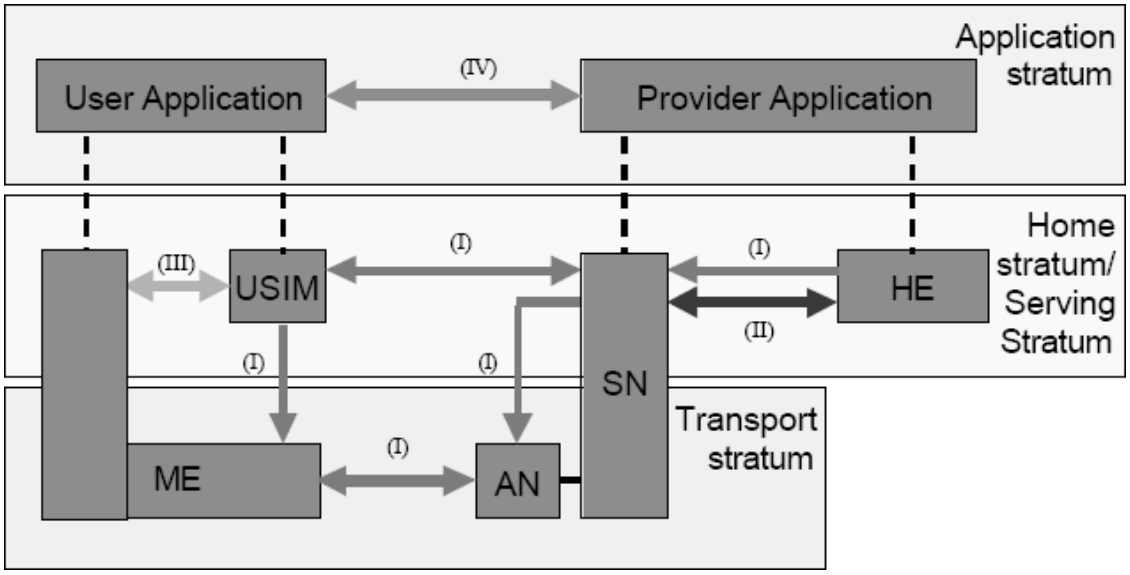
Symmetric algorithms depend on both parties knowing the keys. Larger key means better encryption. DES and A5 are examples of symmetric algorithms. The difficulty with symmetric algorithms is that both parties need to have a copy of the key. To transmit the key freely over the air would render the whole exercise pointless.

Asymmetric algorithms use two separate keys for encryption and decryption. Usually, the encryption key can be publicly distributed, while the recipient holds the decryption key securely. RSA is an example of asymmetric algorithm. The privacy needs of a wireless system can be described in the following areas:

1. **Privacy of call set up information:** During call set up, the mobile station (MS) will communicate information to the network. .Some of the information that a user could send is calling number, calling card number, type of service requested etc. The system must send all this information in a secure way.
2. **Privacy of speech:** The system must encrypt all spoken communication so that hackers cannot intercept the signals by listening on the air waves.
3. **Privacy of the data:** The system must encrypt all user data communications so that hackers cannot intercept the data by listening on the airwaves.
4. **Privacy of user location:** A user should not transmit any information that enables a listener to determine the user's location. The usual method to meet this is to encrypt the user ID. Three levels of protection are needed:
    a. Radio link eavesdropping
    b. Unauthorized access by hackers to the user location information stored in the network at the HLR and VLR.
    c. Unauthorized access by insiders to the user location information stored in the network. This level is difficult to achieve, but not impossible.

5. **Privacy of user identification**: When a user interacts with the network, the user ID is sent in a way that does not show the user ID. This prevents analysis of calling patterns based on user ID.
6. **Privacy of calling patterns:** No information must be sent from MS that enables a listener of the radio interface to do traffic analysis on the mobile user. Typical traffic analysis information is calling number, frequency of user of MSs, caller identity and privacy of financial transaction.

In 1996, when the 3rd Generation system known as UMTS was being developed in ETSI (European Telecommunications Standards Institute), the opportunity was taken to review the basis for security in existing mobile systems and to develop new security architecture specifically to be used in UMTS.



The above figure shows how these elements are retained to provide five specific aspects of security for operators and end users of the 3G network:

I. Network access security provides users with secure access to 3G services and protects against attacks on the (radio) access link;

II. Network domain security enables nodes in the provider domain to exchange signaling data securely;

III. User domain security allows users to have secure access to mobile stations;

IV. Application domain security enables applications in the user and in the provider domain to exchange messages securely;

V. Visibility and configurability of security informs the user if a security feature is in operation or not, allowing appropriate use of the service.

General objectives for 3G security features:

1. To ensure that information generated by or relating to a user is adequately protected against misuse.
2. To ensure that the resources and services provided by serving networks and home environments are adequately protected against misuse.
3. To ensure that the security features standardized are compatible with worldwide availability.
4. To ensure that the security features are adequately standardized to ensure world- wide interoperability and roaming between different serving networks.
5. To ensure that the level of protection afforded to users and providers of services is better than that provided in contemporary fixed and mobile networks.
6. To ensure that the implementation of 3G security features and mechanisms can be extended and enhanced as required by new threats and services.

Introduction

In July 2010, GSM Association announced that GSM has more than 5 billion connections. It doesn't show direct number of user because some people use multiple SIM card. It is still certain that, a lot of people use this system for communication. In GSM, there is one important issue people normally don't think, which is security.

Most of computer users know at least basic concepts of the computer security, such as virus, antivirus, trojan… Besides, more people use at least one cheap anti-virus program for security.

However, people do not even have any idea about GSM security. They blindly trust GSM systems and they do not usually think that there may be an insecure communication. People do not have to know all security mechanisms in technology but they should at least have an idea about weak and strong points in GSM and how opponents can use these weak points against them. [1]

GSM Overview

GSM History

1876 - First telephone was invented by Alexander Bell.

1921 - The first car mounted radio telephone worked.

1946 - First commercial mobile radio-telephone service was started by Bell and AT&T in Saint Louis, USA.

1973 - First handheld cellular phone was released by Motorola.

1978 - First cellular network was setup in Bahrain

1982 - The European Conference of Post and Telecommunications Administrations (CEPT) formed a group called Group Spéciale Mobile (GSM) to develop a European cellular system that would replace the many existing incompatible cellular systems already in place in Europe.

1987 – A milestone was achieved with the signing of the GSM Memorandum of Understanding (MoU) by operators, agreeing to implement cellular networks, based on the

GSM specifications. While it was clear from the start that GSM would be a digital system, it was officially announced in 1987.

1991 - GSM service started. In the same year, GSM was renamed to Global System for Mobile Communications from Group Spéciale Mobile.

Although GSM was initially developed as a European digital communication standard to allow users to use their cellular devices seamlessly across Europe, it soon developed into a standard that would see unprecedented growth globally. [2, 3]

GSM is a lot better system than old analog systems. Key features of GSM can be written as;

- International Roaming - single subscriber number worldwide
- Superior speech quality - better than existing analog cellular technology
- High level of security - user's information is safe and secure
- Universal and Inexpensive Mobile handsets
- Digital Convenience - talk time is doubled per battery life and digital networks can handle higher volume of calls at any one time that analog networks
- New services - such as call waiting, call forwarding, Short Message Service (SMS), GSM
- Packet Radio Service (GPRS)
- Digital compatibility - easily interfaces with existing digital networks like Integrated with Services Digital Network (ISDN)

  [3]

GSM Architecture

Cellular communication means that there are a lot of different areas, looks like cell, contain communication system devices, such as antennas, base stations… If the base station antenna power is high, it means that it serves large areas. Otherwise, if the antenna power is low, it means that it serves little areas. However there are other parameters which affect the convergence. For example, to achieve a good communication in crowded areas, convergence should be reduced and the channel capacity should be increased.

A lot of adjacent cells create the clusters. Clusters can have different amount of cells and each cell uses different frequency to avoid interference. These cluster structures repeat itself in different communication areas.

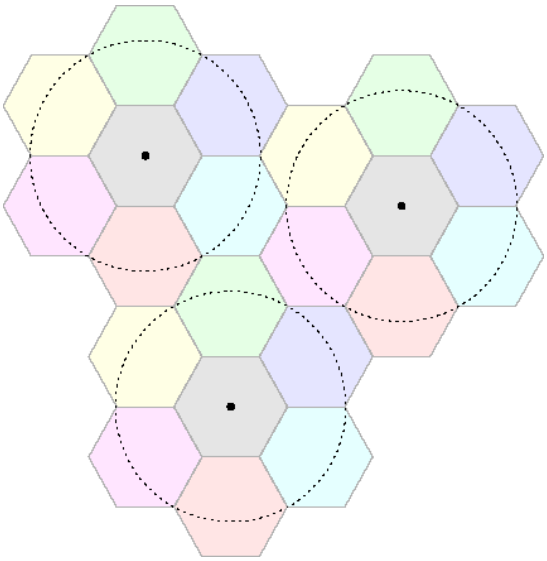


Figure 1: Cell and Cluster Structures

As it can be seen in the figure 1, different cluster cells can use the same frequency since they are far away each other and they don't make any interference.

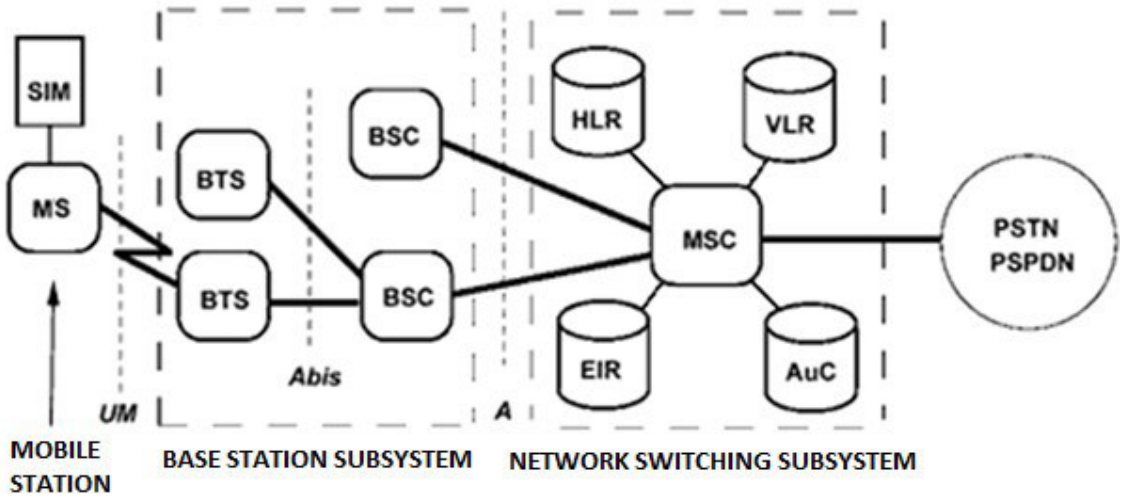


Figure 2: GSM Structure [History GSM]

Figure 2 shows each device in the GSM system. Let's take closer look each of them. [4, 5]

Subscriber Identity Module (SIM) Card

It is operator dependent smart card which contains A3/8 algorithms, IMSI and Ki. There will be god information about SIM card in the security chapter.

Mobile Equipment (ME)

It is operator independent communication device. It contains A5 algorithm. It is useless without SIM card. It never sees A3/8 algorithms and Ki.

Base Transceiver Station (BTS)

The Base Transceiver Station belonging to a PLMN serving the MS. Base stations form a patchwork of radio cells over a given geographic coverage area. Base Stations are connected to base station controllers (BSC).

Base Station Controller (BSC)

It is a node controlling a number of BTS, coordinating handovers and performing BS co-ordination not related to switching. The BSC to BTS link is in many cases a point to point microwave link. BSC are also connected to mobile switching centers (MSC) via fixed or microware links.

Mobile Switching Center (MSC)

It is a node controlling a number of BSC. It is center device and has a lot of function in GSM system. It makes switching, authentication, registering and links the nodes each other. It is connected to PSTN.

Home Location Register (HLR)

It is used for recording the most recent known location of all MS belonging to MS's home area. It contains all administrative information about each registered user of a GSM network along with the current location of the MS.

Visited Location Register (VLR)

It is used for recording information about all MS when they are at the "visiting" area. It tracks mobiles that are out of their home network, so that the network will know where to find them.

Authentication Centre (AuC)

It is used by a HLR to generate random challenges (RAND) and to store secret key information (Ki) relating to each of its MS. The AuC can be integrated with other network functions, e.g. with the HLR. The AUC database contains; International Mobile Subscriber Identity (IMSI), Temporary Mobile Subscriber Identity (TMSI), Location Area Identity (LAI), Authentication Key (Ki).

Equipment Identity Register (EIR)

The EIR is a database that keeps tracks of handsets on the network using the IMEI. There is only one EIR per network. It is composed of three lists; the white list, the gray list, and the black list. The black list is a list if IMEIs that are to be denied service by the network for some reason. Reasons include the IMEI being listed as stolen or cloned or if the handset is malfunctioning or doesn't have the technical capabilities to operate on the network. The gray list is a list of IMEIs that are to be monitored for suspicious activity. This could include handsets that are behaving oddly or not performing as the network expects it to. The white list is an unpopulated list. That means if an IMEI is not on the black list or on the gray list, then it is considered good and is "on the white list". [4, 5]

Subsystems and Working Principles

GSM structure has subsystems, Base Station Subsystem, Network Subsystem and Network Management Subsystem. Moreover, it has 3 interfaces; **UM** interface (air interface), **Abis** interface (between BTS and BSC), **A** interface (between BSC and MSC). [2]

Mobile Station

Every GSM mobile phone has a Subscriber Identity Module (SIM). The SIM provides the mobile phone with a unique identity through the use of the International Mobile Subscriber Identity (IMSI). The SIM is like a key, without which the mobile phone can't function. It is capable of storing personal phone numbers and short messages. It also stores security related information such as the A3 authentication algorithm, the A8 ciphering key generating algorithm, the authentication key (Ki) and IMSI. The mobile station stores the A5 ciphering algorithm. The SIM is removable, which allows users to travel abroad taking with them only their SIM card. They would need to inform their local provider, which countries they would be visiting, and prior to their departure. At their destination, they can simply plug the SIM into a rental cellular phone and make use of the mobile unit. The SIM can be

protected with a Personal Identification Number (PIN) chosen by the subscriber. The PIN is stored on the card and if entered incorrectly thrice, the card blocks itself. At this point, you'll have to contact your cellular provider who can unblock your mobile phone, by entering an eight digit Personal Unblocking Key (PUK), which is also stored on the card. [3]

Base Station Subsystem (BSS)

The role of the Base Station Subsystem (BSS) is to connect the user on a mobile phone with other landline or mobile users. The Base Transceiver Station (BTS) is in direct contact with the mobile phones via the air interface and can be thought of as a complex radio modem. The Base Station Controller (BSC) is responsible for the control of the several BTS. It monitors each call and decides when to handover the call from one BTS to another, as well as manages radio frequencies allocated for the calls through the BTS. [3]

Network Subsystem (NSS)

It is a complete exchange, capable of routing calls from a fixed network via the BSC and BTS to an individual mobile station. The Mobile Services Switching Center (MSC) interconnects the cellular network with the Public Switched Telephone Network (PSTN). The MSC also serves to co-ordinate setting up calls to and from GSM users. The Home Location Register (HLR) stores information of all subscribers belonging to an area served by a MSC. It stores permanent data such as the IMSI, services subscribed by the user, subscriber's number from a public network, Ki and some other temporary data. The HLR has to provide the MSC with all the necessary information when the call is coming from a public network. The Visitor Location Register (VLR) contains relevant information for all mobiles currently served by a MSC. The permanent data stored in the VLR is also stored in the HLR. In addition, it also stores the Temporary Mobile Subscriber Identity (TMSI), which is used for limited intervals to prevent the transmission of the IMSI via the air interface. (See section on GSM Security: Anonymity) The VLR has to support the MSC during call establishment and authentication when the call originates from a mobile station. The Equipment Identity Register (EIR) stores all the International Mobile Equipment Identities (IMEI) of mobile equipment and their rights on the network. The EIR maintains a white, gray and black list. Those on the white list are permitted on the network while those on the black list are blocked from the network. The gray list consists of faulty equipment that may pose a problem on the network but are still permitted to participate on the network. The IMEI reveals the serial number of the mobile station, manufacturer, type approval and country of production. The Authentication Center

(AuC) is a protective database that houses the KI, the A3 authentication algorithm, the A5 ciphering algorithm and the A8 ciphering key generating algorithm. It is responsible for creating the sets of random numbers (RAND), Signed Response (SRES) and the Cipher key (KC), though the created sets are stored in the HLR and VLR. [3]

Network Management Subsystem (NMS)

The Network Management Subsystem (NMS) is the third subsystem of the GSM network in addition to the Network Switching Subsystem (NSS) and Base Station Subsystem (BSS). The purpose of the NMS is to monitor various functions and elements of the network.

The operator workstations are connected to the database and communication servers via a Local Area Network (LAN). The database server stores the management information about the network. The communications server takes care of the data communications between the NMS and the equipment in the GSM network known as “network elements”. These communications are carried over a Data Communications Network (DCN), which connects to the NMS via a router. The DCN is normally implemented using an X.25 Packet Switching Network.

The NMS functions can be divided into three categories; fault management, configuration management and performance management. [6]

GSM Security Principles

Security Goals & Concerns

Security mechanisms should have some features to become efficient. First of all, security concerns can be defined in two side; operator side and customer side.

Operators;

- Should bill the right person
- Should provide systems to avoid fraud
- Should protect their services against attacks

Customers;

- Should have privacy, nobody should be able to detect their identification or their location

- Communication on the air should be encrypted to avoid eavesdropping
- Should be able to change mobile equipment independently

Security mechanisms;

- Shouldn't add much load to the voice calls or datacommunication
- Shouldn't need to increase the channel bandwidth
- Shouldn't increase the bit error rate
- Shouldn't bring expensive complexity to the system
- Should be useful and cost efficient
- Should be able to detect suspicious mobile equipment

[7, 8]

Security Mechanisms

GSM has a lot of security systems to build safe communication. It includes a lot of different types of algorithms and different type of devices.

The main security measurements of GSM security can be written in 4 principles;

Authentication of a user; it provides the ability for mobile equipment to prove that it has access to a particular account with the operator.

Ciphering of the data and signaling; it requires that all signaling and user data (such as text messages and speech) are protected against interception by means of ciphering.

Confidentiality of a user identity; it provides IMSI's (international mobile subscriber identity) security. GSM communication uses IMSI rarely, it uses TMSI (Temporary Mobile Subscriber Identity) to provide more secure communication and to avoid disclosing of user's identity. This means someone intercepting communications should not be able to learn if a particular mobile user is in the area.

Using SIM as security module; Incase SIM card was taken by opponent, there is still PIN code measurement. [4, 5, 9]

A3 and A8 Algorithms

A3 and A8 algorithms are A3 and A8 algorithms are symmetric algorithms which the encryption and decryption use the same key. Both of the algorithms are one way function, it means that output can be found if the inputs are known but it is mostly impossible to find inputs incase the output is known. A3 and A8 algorithms are kept and implemented in SIM card. [10, 11]

Many users of GSM will be familiar with the SIM (Subscriber Identity Module) the small smartcard which is inserted into a GSM phone as you can see in the figure 3.

Figure 3: Sample SIM Card

The SIM itself is protected by an optional PIN code. The PIN is entered on the phone's keypad, and passed to the SIM for verification. If the code does not match with the PIN stored by the SIM, the SIM informs the user that code was invalid, and refuses to perform authentication functions until the correct PIN is entered. To further enhance security, the SIM normally "locks out" the PIN after a number of invalid attempts (normally 3). After this, a PUK (PIN Unlock) code is required to be entered, which must be obtained from the operator. If the PUK is entered incorrectly a number of times (normally 10), the SIM refuses local access to privileged information (and authentication functions) permanently, rendering the SIM useless.

Typical SIM features can be lined as below:

- 8 bit CPU
- 16 K ROM
- 256 bytes RAM
- 4K EEPROM
- Cost: $5-50

On its own, the phone has no association with any particular network. The appropriate account with a network is selected by inserting the SIM into the phone. Therefore the SIM card contains all of the details necessary to obtain access to a particular account. It contains 4 important information; IMSI, Ki, A3 and A8 algorithms. [9]

IMSI (International Mobile Subscriber Identity): Unique number for every subscriber in the world. It includes information about the home network of the subscriber and the country of issue. This information can be read from the SIM provided there is local access to the SIM (normally protected by a simple PIN code). The IMSI is a sequence of up to 15 decimal digits, the first 5 or 6 of which specify the network and country.

Ki: Root encryption key. This is a randomly generated 128-bit number allocated to a particular subscriber that seeds the generation of all keys and challenges used in the GSM system. The Ki is highly protected, and is only known in the SIM and the network's AuC (Authentication Centre). The phone itself never learns of the Ki, and simply feeds the SIM the information it needs to know to perform the authentication or generate ciphering keys. Authentication and key generation is performed in the SIM, which is possible because the SIM is an intelligent device with a microprocessor.

A3 Algorithm: It provides authentication to the user that it has privilege to access the system. The network authenticates the subscriber through the use of a challenge-response method.

Firstly, a 128 bit random number (RAND) is transmitted to the mobile station over the air interface. The RAND is passed to the SIM card, where it is sent through the A3 authentication algorithm together with the KI. The output of the A3 algorithm, the signed response (SRES) is transmitted via the air interface from the mobile station back to the network. On the network, the AuC compares its value of SRES with the value of SRES it has received from the mobile station. If the two values of SRES match, authentication is successful and the subscriber joins the network. The AuC actually doesn't store a copy of SRES but queries the HLR or the VLR for it, as needed. [3]

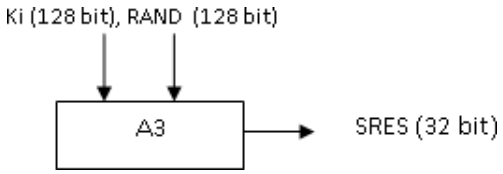


Figure 4: A3 Algorithm

Figure 4 illustrates working principle of A3 algorithm.

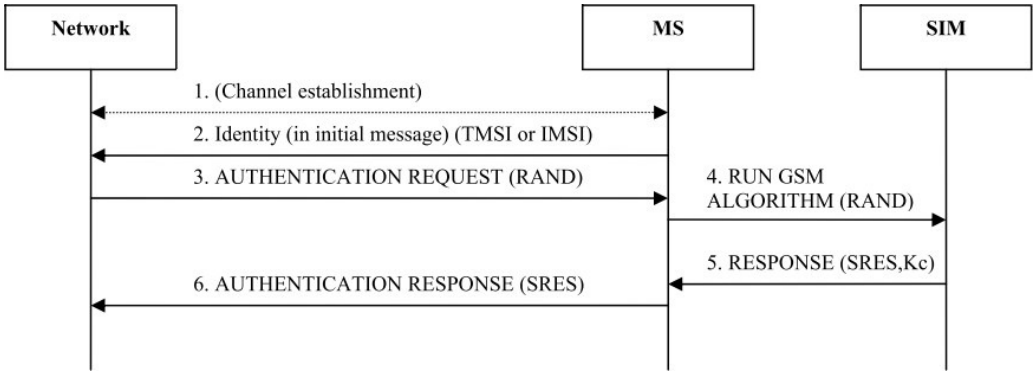


Figure 5: A3 Algorithm Request Order

Figure 5 shows the request order between mobile station and operator network in A3 algorithm. This figure can be explained as;

1) Some connection is attempted between the phone and the network.
2) The phone submits its identity. All potential messages used at the start of a connection contain an identity field. Where possible, it avoids sending its IMSI in plaintext (to prevent eavesdroppers knowing the particular subscriber is attempting a connection). Instead, it uses its TMSI (Temporary Mobile Subscriber Identity). This will be discussed later in this article.
3) The network sends the AUTHENTICATION REQUEST message containing the RAND.
4) The phone receives the RAND, and passes it to the SIM, in the RUN GSM ALGORITHM command.
5) The SIM runs the A3 algorithm, and returns the SRES to the phone.

6) The phone transmits the SRES to the network in the AUTHENTICATION RESPONSE message.
7) The network compares the SRES with its own SRES. If they match, the transaction may proceed. Otherwise, the network either decides to repeat the authentication procedure with IMSI if the TMSI was used, or returns an AUTHENTICATION REJECT message.

A8 Algorithm: GSM makes use of a ciphering key to protect both user data and signaling on the vulnerable air interface. Once the user is authenticated, the RAND (delivered from the network) together with the Ki (from the SIM) is sent through the A8 ciphering key generating algorithm, to produce a ciphering key (Kc). The A8 algorithm is stored on the SIM card. The Kc created by the A8 algorithm, is then used with the A5 ciphering algorithm to encipher or decipher the data. The A5 algorithm is implemented in the hardware of the mobile phone, as it has to encrypt and decrypt data on the air.[3]

Whenever the A3 algorithm runs to generate SRES, the A8 algorithm is run as well The A8 algorithm uses the RAND and Ki as input to generate a 64-bit ciphering key, the Kc, which is then stored in the SIM and readable by the phone. The network also generates the Kc and distributes it to the base station (BTS) handling the connection [9].

Figure 6 shows A8 algorithm working principle;

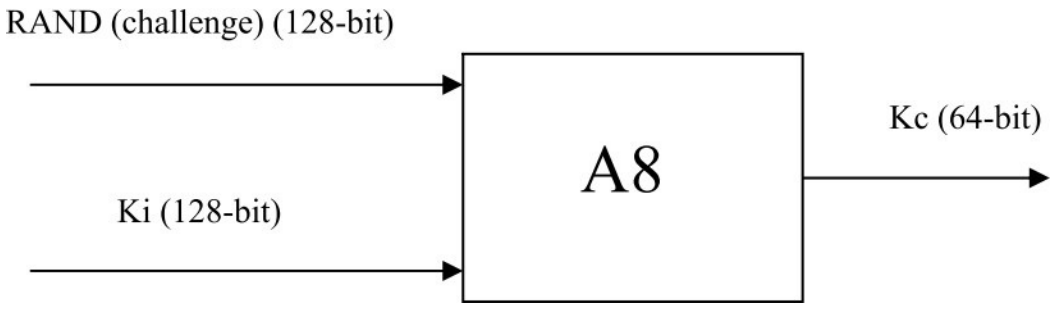


Figure 6: A8 Algorithm

COMP128

COMP128 is hash function which is an implementation of the A3 and A8 algorithms in the GSM standard.

The Algorithm Expert Group was held in 1987 and designed GSM encryption algorithms. They created 2 algorithms; the first one was COMP128 which is to provide authentication and derive the cipher key (A3/8), the second was A5 algorithm. GSM allowed every operator to use its own A3/8 algorithm and the all system support this without transferring between networks, also during roaming. However, most of the operators do not have expertise to make their own A3/8 algorithm design and they use example COMP128 design. [12]

The COMP128 takes the RAND and the Ki as input; it generates 128 bits of output. The first 32 bits of the 128 bits form the SRES response; the last 54 bits of the COMP128 output form the session key, Kc. Note that the key length at this point is 54 bits instead of 64 bits, which is the length of the key given as input to the A5 algorithm. Ten zero-bits are appended to the key generated by the COMP128 algorithm. Thus, the key of 64 bits with the last ten bits zeroed out. This effectively reduces the key space from 64 bits to 54 bits. This is done in all A8 implementations, including those that do not use COMP128 for key generation, and seems to be a deliberate feature of the A8 algorithm implementations. [13]

A5 Algorithm

A5 is a stream cipher which can be implemented very efficiently on hardware. There exist several implementations of this algorithm, the most commonly used ones are A5/0, A5/1 and A5/2 (A5/3 is used in 3G systems). The reason for the different implementations is due to export restrictions of encryption technologies. A5/1 is the strongest version and is used widely in Western Europe and America, while the A5/2 is commonly used in Asia. Countries under UN Sanctions and certain third world countries use the A5/0, which comes with no encryption. [3, 14]

As a stream cipher, A5 works on a bit by bit basis (and not on blocks, as DES and AES). So, an error in the received cipher text will only result in the corresponding plaintext bit being in error.

None of the algorithms are published by GSM Association. They are all discovered by using reverse engineering methods.

A5/1 algorithm uses the structure which can be seen in figure 8. [9]

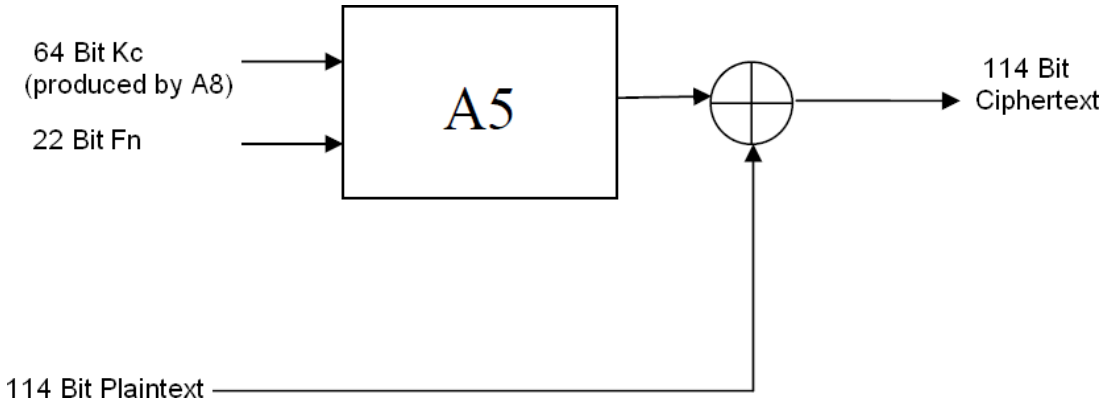


Figure 7: A5 Structure

Kc is the key which was produced by A8 algorithm. Plaintext is the data which is wanted to transmit. Fn is the frame bits which come from **LFSR** (Linear Feedback Shift Register) process.

To understand A5/1 algorithm, **LFSR** (Linear Feedback Shift Register) structure should be introduced firstly.

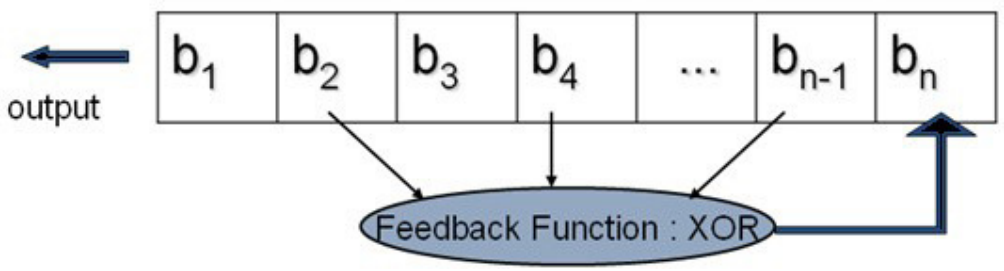


Figure 8: LFSR Structure

As it can be seen in the figure 7, in LFSR structure, there is certain amount of bits which has some special bits (taps). These special taps are XOR ed, all the bits shit one bit left and the result was put to the first bit. [14]

In A5/1 algorithm, LFSR structure uses 3 register bits. These bits are shown in the figure 9.

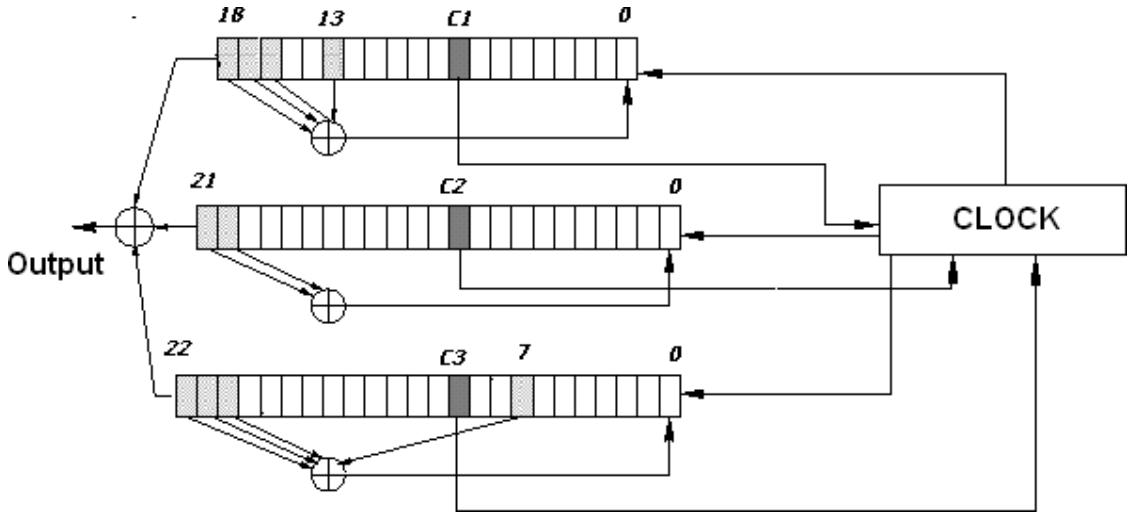


Figure 9: LFSR Structure in A5/1 Algorithm

A5/1 is built from three short linear feedback shift registers (LFSR) of lengths 19, 22, and 23 bits, which are denoted by *R*1, *R*2 and *R*3 respectively. The rightmost bit in each register is labeled as bit zero. The taps of *R*1 are at bit positions 13, 16, 17, 18; the taps of *R*2 are at bit positions 20, 21; and the taps of *R*3 are at bit positions 7, 20, 21, 22. When a register is clocked, its taps are XOR ed together, and the result is stored in the rightmost bit of the left-shifted register. The three registers are clocked in a stop/go fashion using the following majority rule: Each register has a single "clocking" tap (bit 8 for *R*1, bit 10 for *R*2, and bit 10 for for *R*3); each clock cycle, the majority function of the clocking taps is calculated and only those registers whose clocking taps agree with the majority bit are actually clocked. Note that at each step either two or three registers are clocked, and that each register moves with probability 3/4 and stops with probability 1/4.

The process of generating pseudo random bits from the session key Kc and the frame counter Fn is carried out in four steps:

1. The three registers are zeroed, and then clocked for 64 cycles (ignoring the stop/go clock control). During this period each bit of Kc (from least significant bit to most significant bit) is XOR ed in parallel into the lsb's of the three registers.
2. The three registers are clocked for 22 additional cycles (ignoring the stop/go clock control). During this period the successive bits of *Fn* (from lsb to msb)

are again XOR'ed in parallel into the lsb's of the three registers. The contents of the three registers at the end of this step are called the initial state of the frame.

3. The three registers are clocked for 100 additional clock cycles with the stop/go clock control but without producing any outputs.
4. The three registers are clocked for 228 additional clock cycles with the stop/go clock control in order to produce the 228 output bits. At each clock cycle, one output bit is produced as the XOR of the msb's of the three registers. 114 bits of these 228 bits are used in MS-BTS communication and the rest 114 bits are used in BTS-MS communication. [15]

A5/2 is the weak version of A5/1. Different from A5/1, it contains 4 LFSR bits Time complexity of the A5/1 is 2^{54} (if the last 10 bits of the Kc is not zero, then 2^{64}). However, time complexity of A5/2 is 2^{16}. A5/0 is the weakest version between these 3 algorithms. It doesn't make any encryption. [11]

GPRS Security

The security in GPRS is based on the same mechanisms as of GSM. However, GPRS uses a different encryption key to provide security. To authenticate the customer, the same A3/8 algorithms are used with the same Ki, different RAND. The resulting Kc is different than voice communication key and this Kc is used to encrypt GPRS data. This Kc is refered GPRS-Kc to make it different from voice communication Kc. Similarly, SRES and RAND are referred as GPRS-SRES and GPRS-RAND. GPRS cipher is also referred to GPRS A5 or GEA (GPRS Encryption Algorithm). [16]

Weakness of GSM Security

Weak sides of the Security Mechanisms

GSM doesn't have perfect security system. Opponents can eavesdrop the channel in real time. The weak sides of GSM Security mechanisms will be discussed in this chapter.

First of all, most of the operators do not have expertise enough to create new A3/8 algorithms. So they use COMP128 function without even changing it. This is big security problem because all the COMP128 function has found by reverse engineering.

The bit size of the algorithms is weak. A5/1 algorithm uses 64 bit Kc in the best case. Most operators use COMP128 which has 54 bit Kc and last 10 bits are always zero. Also A5/2 is weaker than A5/1.

Moreover, authentication query only exists BTS-MS communication. There is no authentication for MS-BTS. It means that, fake base stations can behave like real BTS and MS will answer each SRES request from them. The network does not authenticate itself to a phone. This is the most serious fault in GSM security, which allows a man-in-the-middle attack. This weakness was known for GSM constructors at the time of the GSM design, but it was expected that building a false BTS would be too expensive and it would be difficult to make those attacks cost effective. However, after 20 years the situation changed significantly. Today there are companies that product short range BTS, so an attacker can simply buy a BTS at a reasonable price.

Another serious vulnerability of the GSM is the lack of proper Caller ID or Sender ID verification. In other words, the caller number or SMS sender number could be spoofed. The caller ID and the voice is transmitted in different channels. So, Called ID or SMS ID can be spoofed.

Another weakness attackers can exploit is vulnerability in the IMSI protection mechanism. As mentioned before, networks use TMSI to protect IMSI but if the network somehow loses track of a particular TMSI it must then ask the subscriber their IMSI over a radio link. The connection cannot be ciphered because the network does not know the identity of the user, and thus the IMSI is sent in plain text. The attacker can thus check whether a particular user (IMSI) is in the vicinity. [1]

History of Cracking Algorithms

In April 1998, the Smartcard Developer Association (SDA) together with two U.C. Berkeley researchers claimed to have cracked the COMP128 algorithm stored on the SIM. By sending large number of challenges to the authorization module, they were able to deduce the K_i within several hours. They also discovered that K_C uses only 54 bits of the 64 bits. The remaining 10 bits are replaced by zeros, which makes the cipher key purposefully weaker. They feel this is due to government interference. A weaker ciphering key, could potentially allow governments to monitor conversations.

The SDA had the SIM in their physical presence when they cracked the algorithm. However they fear "an over the air attack" is not farfetched. Unfortunately, they are unable to confirm their suspicions, as the equipment required to carry out such an attack is illegal in the US. The GSM Alliance responded to the incident, stating even if a SIM could be cloned it would serve no purpose, as the GSM network would only allow only one call from any phone number at any one time. GSM networks are also capable of detecting and shutting down duplicate SIM codes found on multiple phones.

In August 1999, an American group of researchers claimed to have cracked the weaker A5/2 algorithm commonly used in Asia, using a single PC within seconds.

In December 1999, two leading Israeli cryptographers claimed to have cracked the strong A5/1 algorithm responsible for encrypting conversations. They admit the version they cracked may not be the exact version used in GSM handsets, as GSM operators are allowed to make small modifications to the GSM algorithms. The researchers used a digital scanner and a high end PC to crack the code. Within two minutes of intercepting a call with a digital scanner, the researchers were able to listen to the conversation. In the US, digital scanners are illegal. The GSM Alliance of North America has claimed that none of its members use the A5/1 algorithm, opting for more recently developed algorithms.

In May 2002, The IBM Research group discovered a new way to quickly extract the COMP128 keys using side channels. As it can be seen in the figure 10 and figure 11, side channels are the natural features of the devices such as power consumption, electromagnetic radiation, timing, errors… [3, 8]

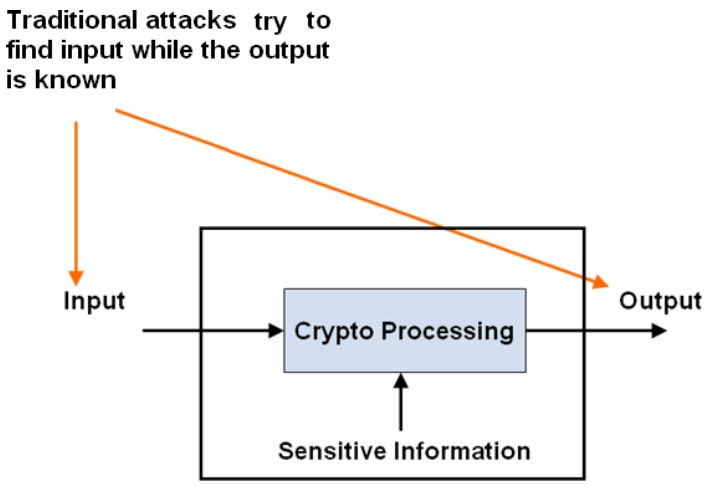


Figure 10: Traditional Attacks

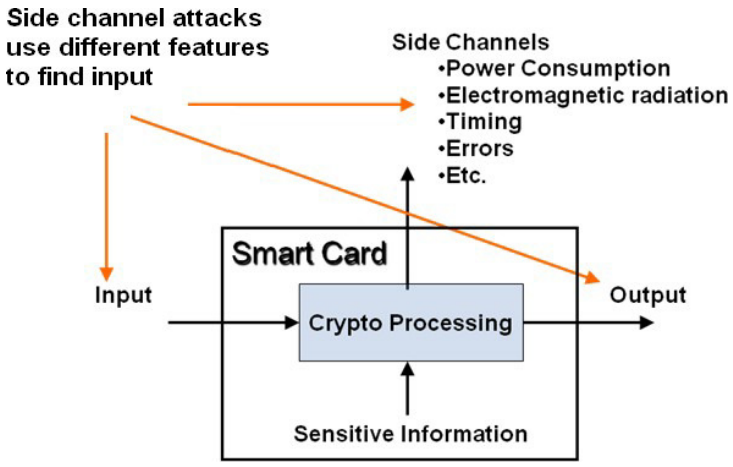


Figure 10: Side Channel Attacks

Popular Attacks Types

In this chapter, it will be described the significant attacks to the anonymity, authentication and confidentiality.

Capturing One or Several Mobile Stations

In many of the attacks described in this chapter, the attacker needs to pretend the network to the MS or pretending the MS to the network or both in a so called man-in-the middle attack. An attacker impersonating BS and MS to each other can eavesdrop, modify, and erase, order, replay, spoof and relaying signals/user data between two communicating entities. The required equipment is an adjusted and modified BTS and MS bundle. The modified BTS behaves as the identity the network to the MS, while the modified MS impersonates the MS to the network.

This mechanism can be seen in figure 11 below.

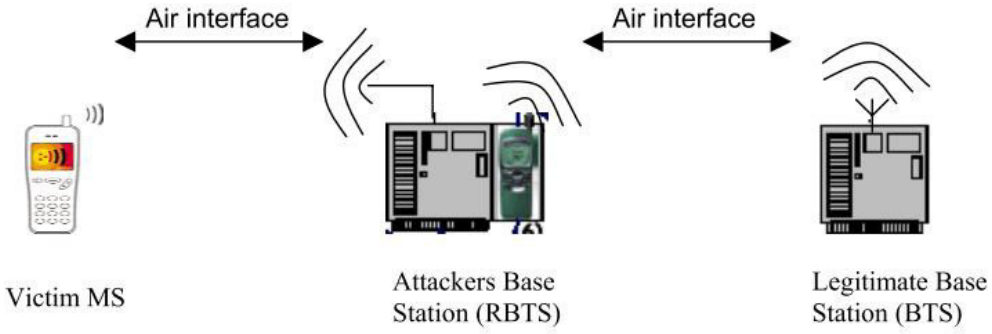


Figure 11: Man-in-Middle mechanism

Before an active attack, the attacker may have to eavesdrop the MS. The attacker may want to learn the information consists of cell identity, network identity, and control channel structure, list of channels in use and details of the access protocol. In this manner, an attacker with a Fake BTS, providing higher power levels than the BTS, between the victim MS and legitimate BTS, forces the MS use the FBTS. The MS captured by the attacker who controls what messages go between the MS and BTS as well as messages flowing in the other direction. After capturing MS identity, the attacker will then use this to provide fabricated messages on behalf of a legitimate subscriber.

Attacks on the Anonymity of GSM Users

The anonymity in GSM is provided by using temporary identifier TMSI, which is like a nickname of subscriber locally. An attacker may want some subscriber's movements and/or pursue call samples and so must have the IMSI and the TMSI of the MS. This information may also be used to attack other security assets than anonymity, for instance eavesdropping on a specific person. If the attacker can get the IMSI of subscriber or associated current TMSI of a specific person then the anonymity of the user is imperiled.

Passive Monitoring:

Every time a MS is powered on, MS is required to introduce itself to the network. This is performed by an IMSI attach. IMSI attach occurs in case of location update. Since the IMSI is not registered in the network and there is not yet any authentication, an encryption cannot

be applied. Therefore the IMSI is sent in the clear. An attacker listening to the air traffic can extract the IMSI and associated subscriber's being active.

Passively track GSM users and eavesdropping on the users' permanent identity (IMSI) is possible and easy. This information provides the attacker with a functional IMSI and the knowledge of that the owner of IMSI is in the present area. Passive monitoring is however inefficient and time-consuming since the attacker needs to either wait for MSs to perform IMSI attach when it is powered on or for a database failure to occur in the network, which probably does not happen so frequently.

Active Monitoring:

To track a GSM subscriber, the attacker can make use of the identification procedure. The network may initiate an identification procedure, if the network cannot identify the MS using its TMSI. The identification procedure begins with transmitting an IDENTITY REQUEST message to MS so as to ask it to transmit an identification parameter. The network can request IMSI, IMEI or TMSI. Since GSM does not use message authentication to check message origin on the radio link, an attacker with sufficient base station functionality can use these messages to retrieve the same information as a legitimate base station by deceiving the target MS s.

It should be possible to request from a subscriber (whose IMSI is known by attacker) for his/her TMSI abusing the identification procedure. When the attacker knows the IMSI/TMSI bundle, it is possible to locate a specific subscriber. The attacker simply pages the MS with the specific IMSI/TMSI.

Attacks on the Authentication Algorithm

Many GSM operators use the design specification given in the GSM MoU, COMP128, instead of designing their own algorithm for authentication (A3) and session key generation (A8). The difficulty in starting to create new algorithm is that subscribers who bought SIMs before eventual introduction of a different algorithm, are forced to use their old SIMs with the old algorithm. Other reason to change/revise the algorithm is the cost of changing software in database etc. On the other hand, it is possible to utilize new and more secure versions of COMP128 in new SIMs that are given to new subscribers.

The design of COMP128 was never made public, but the design has been reverse engineered and cryptanalyzed. Today, it is quite easy to find software implementation of

COMP128 by simple search on internet. Since the GSM specification for SIM cards is widely available, all that is needed to clone a SIM card is the 128 bit COMP128 secret key Ki and the IMSI which is coded in the SIM. [17]

By copying Ki and IMSI into an empty SIM (easy to buy from web) the attacker can authenticate himself to the network as the legitimate subscriber and thus call by charging. The attacker can even, instead of using the subscription, use the captured key Ki for decrypting all the calls from and to the subscriber.

Cloning can be done either by physical access to the SIM to be cloned, or over the air. The following subsections will examine the two cases:

Cloning with Physical Access to the SIM Module:

If the attacker has physical access to the SIM module, several attacks can be launched in order to clone the module. Some of these attacks base on using flows in the cryptographic algorithm resided in the smart card, while others use vulnerabilities in the smart card itself.

The most popular attack to SIM modules is the attacks to the cryptographic algorithm (COMP128) itself. It is a chosen-challenge attack and use flows in the hashing function to deduce the secret key Ki. The attacker creates a number of specially-chosen challenges and queries the SIM for each one. The SIM applies COMP128 to its secret key and the chosen challenge, returning a response back. After analyzing the responses, the attacker can determine the Ki. The result of this attack is thus that the attacker gains access to the secret key Ki of the MS. The attack exploits a lack of diffusion, which means that some parts of the output hash depend only on some parts of the input to the algorithm. Mounting this attack requires, apart from having physical access to the target SIM, an off-the-shelf smartcard reader, and a computer to direct the operation. The attack requires one to query the SIM about 150,000 times; an average SIM reader can issue 6.25 queries per second, so the whole attack takes approximately 8 hours. By overclocking the SIM or using a higher frequency oscillator on the SIM card reader the processing time could be reduced considerably. This increases however the risk of failure and damage to the original SIM. [18, 19]

Cloning over the Air:

The attacker can even perform the attack over the air, making use of a fake base station. Apart from this equipment, the attacker needs to know the target IMSI or TMSI. The

captured MSs will be immediately forced to make a location update request which is conducted. After the channel allocation is completed the attacker initiates an authentication process. Immediately after the attacker has a challenge-response pair, he/she initiates a new authentication procedure. The MS is required to respond to every challenge made by the GSM network. This process continues until the attacker has got the required number of pairs to be able to initiate the cloning procedure.

It is assumed that the channel establishment stage only has to be done once. The number of frames exchanged between the network and an MS, for one authentication process, are approximately 66 frames. Since the duration of one TDMA frame is 4.610 ms, the duration of the whole signaling sequence is 4.615 ms/frame x 66 frames = 0.30459 s. The time it takes to get the number of challenge-response pairs needed for the attack can be calculated. It is known that the cryptographic attack requires approximately 150 000 challenge-response pairs. This means that the attack takes approximately 45,689 seconds (150 000 challenges x 0.30459 s), that is approximately 13 hours. This means that the MS has to be available to the attacker over the air for the whole time it takes to gather the information. This is quite unrealistic, because people use their mobiles to make calls or receive calls in addition to the fact that such a bombardment with challenges may cause the battery of the MS to run out, which would make the victim suspicious. To get rid of these problems, the attack can be performed in parts; instead of performing a 13-hour attack, the attacker could interrogate the MS for 30 minutes every day. In that way, the battery would not run out and there would be less risk of making the owner or the legitimate network suspicious.

The defense against cloning over the air is to limit the number of times a SIM can be authenticated to a number significantly smaller than 150 000. The SIM locks up if the limit is exceeded. The drawback about this solution is that a new SIM module has to be issued and distributed to the subscriber, which results in costs both for the subscriber and the operator. [12]

Attacks on the Confidentiality of GSM

As mentioned before, the over-the-air privacy of GSM telephone conversations is protected using the A5 stream cipher. This algorithm has two main variants: A5/1 is the “strong” export-limited version used by CEPT-countries, and A5/2 is the “weak” version that has no export limitations. The exact design of both A5/1 and A5/2 was reverse engineered by Briceno from an actual GSM telephone in 1999. [21]

In the following subsections attacks are classified into three: brute-force attacks, crypto analytical attacks, and non-crypto analytical attacks.

Brute-Force Attacks:

The confidentiality of GSM is protected by the secrecy of Kc. Kc is 64 bits although the last 10 bits are set to zero. This reduces the key space from 2^64 to 2^54. A5/2 was developed with assistance from the NSA, and can be broken in real time with a work factor of approximately 2^16. A5/1, the stronger of the two variants, is however susceptible to attacks that can break it with a work factor of 2^40.

Pentium 4 chip has nearly 60 million transistors and the implementation of one set of LFSRs (A5/1) would require about 2000 transistors, 30.000 parallel A5/1 implementations on one chip can be done. If the chip was clocked to 3.2 GHz (a rather ambitious assumption) and each A5/1 implementation would generate one output bit for each clock cycle then it is needed to generate 100+114+114 output bits, hence approximately 10M keys per second per A5/1 implementation can be used. A key space of 2^54 would thus require about 18 hours, using all of the parallel implementations on the chip. If the attack in the average case succeeds after searching half of the key space, the key is found in about 9 hours. Further optimization by giving up on a specific key after the first invalid key stream bit and distributing the computation between multiple chips will decrease the computation time by several magnitudes. This, still in the worst case, means several hours/many minutes of processing and is far away from a real-time attack. Bear in mind that the complexity of the attack is even greater due to the fact that it is quite difficult to determine when the key is found due to the nature of the plaintext. [22, 23]

To conclude, it is too difficult to succeed in a brute-force attack in real-time, but it is fully possible to find a key given a couple of hours. Entities with enough resources (computation power) can probably cut the processing time greatly.

Even though a brute-force attack may not be used as a real-time attack on the A5 algorithm, it could easily be used to find the key used in a specific conversation “offline”. The attacker intercepts and records the interesting conversation and decrypts it at a later time.

Crypto Analytical Attacks:

There exist several crypto analytical attacks against the algorithms protecting different aspects of GSM. The algorithm used by many operators to authenticate subscribers (COMP128) is broken due to flaws in the design of the hash function. The result is the ability of intruders to clone subscriptions either by having physical access to the target SIM or over the air. The most popular attack requires physical access to the SIM to clone and is completed in about 8 hours. It can be speeded up with the risk of damaging the SIM. The most efficient way to clone a GSM smartcard is a partitioning attack proposed by a team from IBM. It requires challenging the target SIM only 8 times in the best case, which means that cloning can be done in minutes or even seconds. The equipment needed to mount this attack (a specially designed smartcard reader and software) is however only available in laboratories yet. Newer versions of COMP128 has been developed and distributed. It is however not known to what extent these stronger versions have been adapted by operators. A guess is that many operators still use the old algorithm due to the costs involved in upgrading. What is known for sure is that users who had COMP128 inside their SIMs when they bought a subscription are still using COMP128.

There exist several crypto analytical propositions on how to attack the encryption algorithms used for confidentiality protection that break these algorithms in real-time. Several attacks against A5/1 and A5/2 exist, although most of them have only theoretical value. Most of the attacks require that the attacker knows portions of the key stream. It is possible to obtain small portions of plaintext because the attacker often knows the structure and content of the signaling messages (especially if the attacker is impersonating the network to the victim MS and is thereby able to query the MS for information) in addition to the fact that channel coding is applied to the data before encryption. An attacker mounting a man-in-the-middle attack may ask the victim subscriber to transmit certain signaling messages (of which the content is known or almost known) after encryption has started. The attacker then has access to the cipher-text in addition to the known portions of the plaintext and can thereby derive portions of the key stream used in the encryption process. It is, however, hard to obtain the amounts of the known plaintext that some of these attacks require. The attack that requires least known plaintext is an attack against A5/2. It requires that the attacker knows the plaintext of two frames approximately six seconds apart from each other and finds the session key in about 10 ms. The known plaintext requirement may be possible to satisfy using the method mentioned earlier, therefore this attack on A5/2 has been used in one of the attacks on

confidentiality. It is worth mentioning that it only took a couple of hours to crack A5/2, which illustrates the weaknesses of this algorithm.

The most recent attack on A5/1 is a cipher-text-only attack. This is an impressive attack only requiring knowledge of a small number of encrypted frames, enabling the attacker to listen to the "encrypted" conversation data, in real-time. Further the authors (of [15]) propose a ciphertext-only attack on A5/2 that improves the previous attack on A5/2 to a ciphertext-only attack. The problem of known plaintext is no longer a concern using this attack. This is however an attack requiring huge amounts of computation power. Figure 12 below gives an idea of the computation requirements in the proposed ciphertext-only attack on A5/1. [15]

Available data (Ciphertext)	Pre-processing steps	Number of PCs to complete pre-processing in one year	Number of 200 GB disks	T	Number of PCs to complete attack in real-time
2^{12} (appr 5 min)	2^{52}	140	22	2^{28}	1
$2^{6.7}$ (appr 8 sec)	2^{41}	5000	176	$2^{32.6}$	1000
$2^{6.7}$ (appr 8 sec)	2^{42}	5000	350	$2^{30.6}$	200
2^{14} (appr 20 min)	2^{35}	35	3	2^{30}	1

Figure 12: Three possible tradeoff points in the attacks on A5/1

Since this is a ciphertext only attack, no plaintext is required in order to find the session key in real-time. However, the computation and storage requirements for this attack are very high making it very unlikely that an individual hacker would have the needed resources to mount the attack. The requirements for the ciphertext-only cryptanalysis of A5/2 are however fulfilled by most personal computers of today.

Non-Crypto Analytical Attacks:

It is common knowledge that most GSM mobile phones can communicate with most different base stations and networks. This is possible because all of the different manufacturers follow the specifications and standards of how GSM should function. These specifications are developed by the European Telecommunications Standards Institute (ETSI). It is possible to study the specifications on how the communication between the network and the MS is conducted, and get detailed information on the communication protocols and mechanisms used when a MS is to be authenticated by the network.

The same key Kc is used for the different encryption algorithms A5/1, A5/2, and A5/3. This means that breaking one of this three algorithms and retrieving the session key threatens the confidentiality of the conversation even when the stronger versions of the algorithm are used later.

A base station does not need to authenticate itself to the MS it is communicating with. Furthermore messages are not authenticated and their integrity is not protected.

Denial of Service (DoS) Attacks

DoS attacks can be performed by physically disturbing radio signals or by logical means. These two possibilities will be explained further in the two sections below.

Denial of Service – Physical Intervention

The physical attacks are the most straight forward attacks. The attacker prevents user or signaling traffic from being transmitted on any system interface, whether wired or wireless, by physical means. An example of physical intervention on a wired interface is wire cutting. The attacker could for example cut the wire leaving a base station. An example of physical intervention on a wireless interface is jamming. Having the equipment that jam GSM radio signals is sufficient. The equipment is placed in the area where traffic is to be disturbed and the GSM equipment within the device's range will not function properly. Note that the frequency hopping makes the jamming moredifficult than usual.

There are examples of jamming causing problems for GSM operators. Recently a GSM operator in Moldova suffered heavily from jamming activities that effectively caused a drop rate of lost calls of about 7 %. The operator and the authorities had major problems in stopping the attacks. [25]

Denial of Service – Logical Intervention

An attacker can perform DoS attacks by logical means also as the following examples show:

- The attacker spoofs a de-registration request (IMSI-detach) to the network. The network de-registers the subscriber from the visited location area and instructs the HLR to do the same. The user is subsequently unreachable for other subscribers. The attacker needs a modified MS and the IMSI of the user to de-register.

Introduction

Electronic commerce, as exemplified by the popularity of the Internet, is going to have an enormous impact on the financial services industry. No financial institution will be left unaffected by the explosion of electronic commerce. Even though SSL is extremely effective and widely accepted as the online payment standard, it requires the customer and merchant to trust each other: an undesirable requirement even in face-to-face transactions, and across the Internet it admits unacceptable risks.

Visa and MasterCard and a consortium of 11 technology companies made a promise to banks, merchants, and consumers: they would make the Internet safe for credit card transactions and send electronic commerce revenues skyward. With great fanfare, they introduced the Secure Electronic Transaction protocol for processing online credit card purchases [1].

Overview of SET Protocol

Secure payment systems are critical to the success of E-commerce. There are four essential security requirements for safe electronic payments (Authentication, Encryption, Integrity and Non-repudiation). Encryption is the key security schemes adopted for electronic payment systems, which is used in protocols like SSL and SET.

Problem with SSL

The SSL protocol, widely deployed today on the Internet, has helped create a basic level of security sufficient for some hearty souls to begin conducting business over the Web. SSL is implemented in most major Web browsers used by consumers, as well as in merchant server software, which supports the seller's virtual storefront in cyberspace. Hundreds of millions of dollars are already changing hands when cybershoppers enter their credit card numbers on Web pages secured with SSL technology.

In this sense, SSL provides a secure channel to between the consumer and the merchant for exchanging payment information. This means any data sent through this channel is encrypted, so that no one other than these two parties will be able to read it. In other words, SSL can give us confidential communications, it also introduces huge risks:

The cardholder is protected from eavesdroppers but not from the merchant. Some merchants are dishonest: pornographers have charged more than advertised price, expecting their customers to be too embarrassed to complain. Some others are just hackers who put up a snazzy illegal Web site and profess to be the XYZ Corp., or impersonate the XYZ Corp. and collecting credit card numbers for personal use.

The merchant has not protected from dishonest customers who supply an invalid credit card number or who claim a refund from their bank without cause. Contrary to popular belief, it is not the cardholder but the merchant who has the most to lose from fraud. Legislation in most countries protects the consumer.

Purpose

The purpose of the SET protocol is to establish payment transactions that

provide confidentiality of information;
ensure the integrity of payment instructions for goods and services order data;
authenticate both the cardholder and the merchant .

Main Entities

There are four main entities in SET:

Cardholder (customer)
Merchant (web server)
Merchant's Bank (payment gateway, acquirer): payment gateway is a device operated by an acquirer. Sometime, separate these two entities.
Issuer (cardholder's bank)

How it Works

Both cardholders and merchants must register with CA (certificate authority) first, before they can buy or sell on the Internet, which we will talk about later. Once registration is done, cardholder and merchant can start to do transactions, which involve 9 basic steps in this protocol, which is simplified.
Customer browses website and decides on what to purchase
Customer sends order and payment information, which includes 2 parts in one message:
    Purchase Order – this part is for merchant
    Card Information – this pat is for merchant's bank only.
Merchant forwards card information (part b) to their bank
Merchant's bank checks with Issuer for payment authorization
Issuer send authorization to Merchant's bank
Merchant's bank send authorization to merchant
Merchant completes the order and sends confirmation to the customer
Merchant captures the transaction from their bank
Issuer prints credit card bill (invoice) to customer

Protocol Overview

SET (Secure Electronic Transaction) is a very comprehensive security protocol, which utilizes cryptography to provide confidentiality of information, ensure payment integrity, and enable identity authentication. For authentication purposes, cardholders, merchants, and acquirers will be issued digital certificates by their sponsoring organizations.

It relies on cryptography and digital certificate to ensure message confidentiality and security. Digital envelop is widely used in this protocol. Message data is encrypted using a randomly generated key that is further encrypted using the recipient's public key. This is

referred to 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.

Digital certificates, which are also called electronic credentials or digital IDs, are digital documents attesting to the binding of a public key to an individual or entity. Both cardholders and merchants must register with a *certificate authority* (CA) before they can engage in transactions. The cardholder thereby obtains electronic credentials to prove that he is trustworthy. The merchant similarly registers and obtains credentials. These credentials do not contain sensitive details such as credit card numbers. Later, when the customer wants to make purchases, he and the merchant exchange their credentials. If both parties are satisfied then they can proceed with the transaction. Credentials must be renewed every few years, and presumably are not available to known fraudsters.

Secure Electronic Transactions (SET) relies on the science of cryptography – the encoding and decoding messages. There are two primary encryption methods in use today: secret-key cryptography and public-key cryptography. Secret-key cryptography is impractical for exchanging messages with a large group of previously unknown correspondents over a public network. For a merchant to conduct transactions securely with millions of subscribers, each consumer would need a distinct key assigned by that merchant and transmitted over a separate secure channel. However, by using public-key cryptography, that same merchant could create a public/private key pair and publish the public key, allowing any consumer to send a secure message to that merchant. This is why SET uses both methods in its encryption process. The secret-key cryptography used in SET is the well-known Data Encryption Standard (DES), which is used by financial institutions to encrypt PINs (personal identification numbers). And the public-key cryptography used in SET is RSA. In the following section, the usage of symmetric (secret-key) and asymmetric (public-key) key encryption in SET will be discussed.

Use of Symmetric Key

In SET, message data is encrypted using a randomly generated symmetric key (a DES 56-bit key). This key, in turn, is encrypted using the message recipient’s public key (RSA). The result is the so called “digital envelope” of the message. This combines the encryption speed of DES with the key management advantages of RSA public-key encryption. After encryption, the envelope and the encrypted message itself are sent to the recipient. After receiving the encrypted data, the recipient decrypts the digital envelope first using his or her private key to obtain the randomly generated symmetric key and then uses the symmetric key to unlock the original message.

This level of encryption, using DES, can be easily cracked using modern hardware. In 1993, a brute-force DES cracking machine was designed by Michael Wiener – one which was massively parallel. For less than a million dollars, a 56-bit DES key could be cracked in average time of 3.5 hours. For a billion dollars, a parallel machine can be constructed that cracks 56-bit DES in a second (Schneier, 1996). Obviously, this is of great concern since DES encrypts the majority of a SET transaction.

In SET, the public key cryptography is only used to encrypt DES keys and for

authentication (digital signature) but not for the main body of the transaction. In SET, the RSA modulus is 1024 bits in length (Using the latest factoring results it appears that factoring a 1024-bit modulus would require over 100,000,000,000 MY of computational effort). To generate the digital signature, SET uses a distinct public/private key. Each SET participant possesses two asymmetric key pairs: a “key exchange” pair, which is used in the process of section key encryption and decryption, and a “signature” pair for the creation and verification of digital signatures (160-bit message digests).

The algorithm is such that changing a single bit in the message will change, on average, half of the bits in the message digest. Approximately, the possibility of two messages having the same message digest is one in 1,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000, which means it is computationally unfeasible to generate two different messages that have the same message digest.

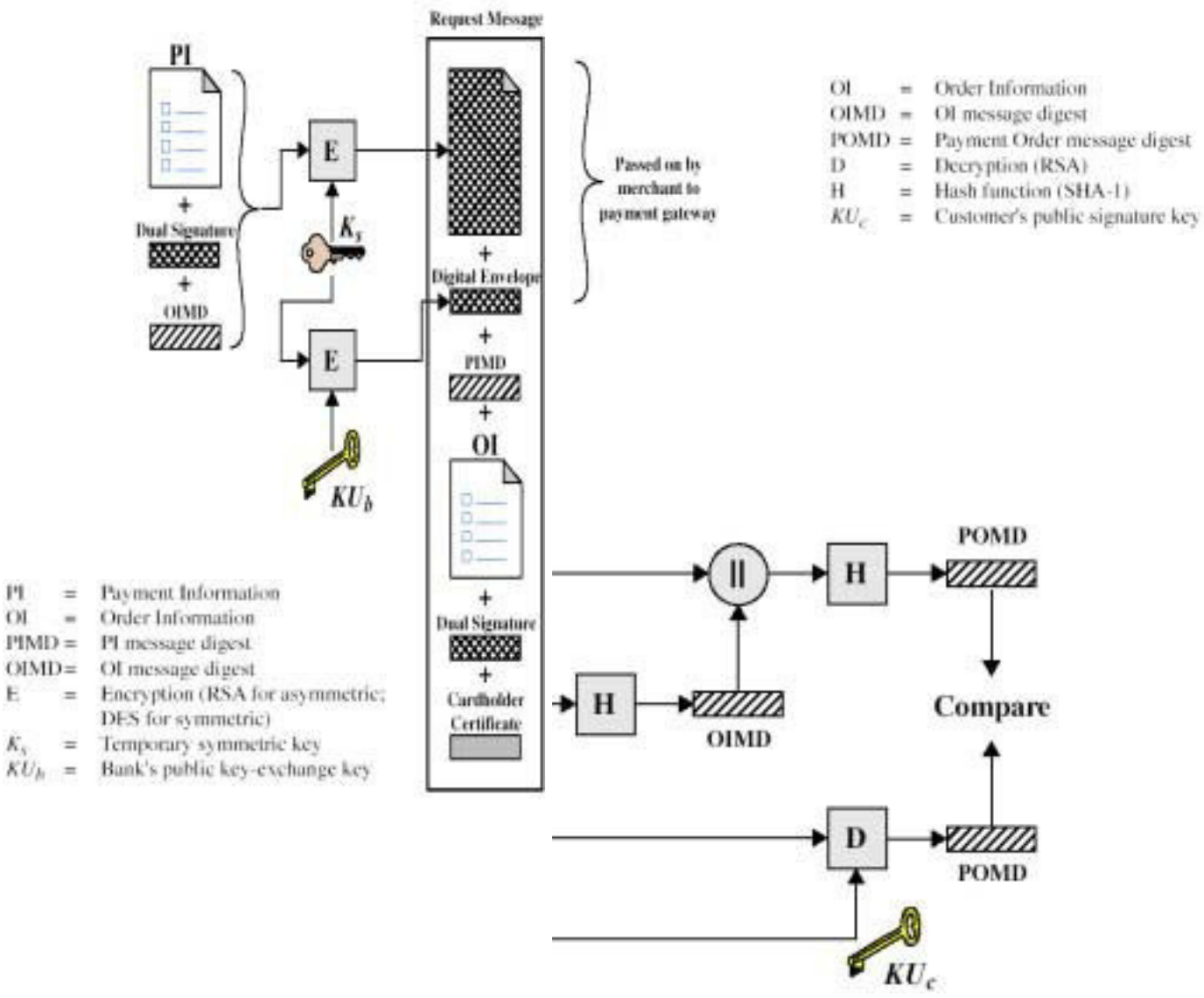


Figure 1- Cardholder Sends Purchase Request / Merchant Verifies Customer Purchase Request

RSA-OAEP

RSA-OAEP (RSA Encryption Scheme - Optimal Asymmetric Encryption Padding) was proposed by Bel-lare and Rogaway in 1994 which is one of the innovations of SET. RSA-OAEP public-key encryption scheme combines the encoding method of OAEP with the encryption primitive RSA. RSA-OAEP takes a plaintext as input, transforms it into an encoded message via OAEP and apply RSAEP (RSA encryption primitive) to the result

(interpreted as an integer) using an RSA public key. RSA-OAEP is intended to be both efficient and secure and is designed to encrypt only short messages--typically secret keys for symmetric encryption or MAC algorithms. OAEP ties the security of RSA encryption closely to that of the basic RSA operation. The version of OAEP used in SET is a more advanced version of the original scheme. While existing message formatting methods for RSA encryption have no known flaw, the provable security aspects of OAEP are very appealing. OAEP is very new but already it is a part of the IEEE P1363 standards effort.

RSA-OAEP encryption scheme has been proven to be semantically secure against adaptive chosen-ciphertext attacks in the random oracle model under the RSA assumption. However, the reduction is not tight, and thus it is not clear what security assurances the proof provides. It is recommended that RSA-OAEP be modified to RSA-OAEP+ that has a tighter security reduction, and furthermore can be easily modified to allow encryption of arbitrarily-long messages. Furthermore, the RSA-KEM encryption scheme of which has a tight reduction should be considered as a replacement for RSA-OAEP.

Dual Signatures

A new application of digital signatures is introduced in SET, namely the concept of dual signatures. Dual signatures is needed when two messages are need to be linked securely but only one party is allowed to read each. The following picture shows the process of generating dual signatures.

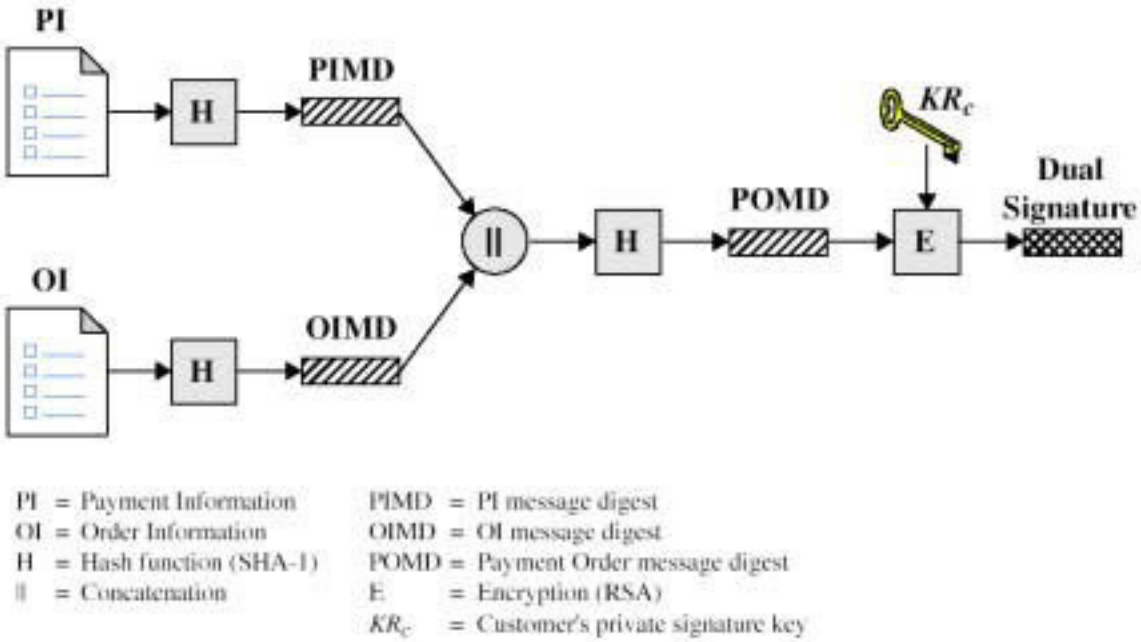


Figure 2- Construction of Dual Signatures in SET

In SET, dual signatures are used to link an order message sent to the merchant with the payment instructions containing account information sent to the acquirer (merchant bank). When the merchant sends an authorization request to the acquirer, it includes the payment instructions sent to it by the cardholder and the message digest of the order information. The acquirer uses the message digest from the merchant and computes the message digest of the payment instructions to check the dual signatures.

SET Process

The SET protocol utilizes cryptography to provide confidentiality of information, ensure payment integrity, and enable identity authentication. For authentication purposes, cardholders, merchants, and acquirers will be issued digital certificates by their sponsoring organizations. It also use *dual signature*, which hides the customer's credit card information from merchants, and also hides the order information to banks, to protect privacy.

Process Steps

1). Merchant sends invoice and unique transaction ID (XID)
2). Merchant sends merchant certificate and bank certificate (encrypted with CA's private key)
3). Customer decrypts certificates, obtains public keys
4). Customer generates order information (OI) and payment info (PI) encrypted with different session keys and dual-signed
5). Merchant sends payment request to bank encrypted with bank-merchant session key, PI, digest of OI and merchant's certificate
6). Bank verifies that the XID matches the one in the PI
7). Bank sends authorization request to issuing bank via card network 8). Bank sends approval to merchant
9). Merchant sends acknowledgement to customer

Payment Initialization

The Purpose of the payment initialization is to allow customer to get certificate from the merchant. The initialization request is represented as PinitReq which carries eight fields of information (Table 1). The security field in SET order information is listed in Table 2.

Table 1- Fields in Payment Initialization

Field	Information
RRPID	Request/Response Pair ID
Language	Customer's Language
LID_C	Customer's Local ID
[LID_M]	Merchant's Local ID
Chall_C	Customer's challenge salt to Merchant's signature freshness
BrandID	Card Brand (VISA, Master etc.)
BIN	Bank ID Number
Thumbs	Thumbnails (hashes) of of certificates known to Customer

Table 2- Security Field in Order Information

Field	Information	Security
OIData {TransIDs, RRPID, Chall-C, HOD, ODSalt, [Chall-M], BrandID, BIN, [ODExtOIDs], [OIExtensions]}		
TransIDs	TRANSACTION Ids copied from PinitREs	Globally unique
RRPID	Request/response pair ID	
Chall-C	Copied from corresponding PInitReq	
HOD	DD(HODInput) Links OIData to PurchAmt without compying PurchAmt into OIData, which would create confidentiality problems.	Hash of order data
ODSalt	Copied from HODInput	Order data hash (to guard against dictionary attack on order data hash)
Chall-M	Merchant's challenge to Cardholder's signature freshness	Merchant's challenge to cardholder signature freshness
BrandID	Cardholder's chosen payment card brand BIN Bank identification number from the cardholder's account number	
ODExtOIDs	List of object identifiers from ODExtensions in the same order as the extensions appeared in ODExtensions	
OIExtensions	The data in an extension to the OI should relate to the merchant's processing of the order	

Certificates Insurance

Before two parties use public-key cryptography to conduct business, each wants to be sure that the other party is authenticated. One way to be sure that the public key belongs to the right party is to receive it over a secure channel directly from the same place. However, in most circumstances this solution is not practical.

An alternative to secure transmission of the key is to use a trusted third party to authenticate that the public key belongs to Alice. Such a party is known as a *Certificate Authority* (CA). Because SET participants have two key pairs, they also have two certificates. Both certificates are created and signed at the same time by the Certificate Authority.

Cardholder certificates function as an electronic representation of the payment card. Because they are digitally signed by a financial institution, they cannot be altered by a third party and and can only be generated by a financial institution. A cardholder certificate does not contain the account number and expiration date. Instead the account information and a secret value known only to the ardholder's software are encoded using a one-way hashing algorithm. If the account number, expiration date, and the secret value are known, the link to the certificate can be proven, but the information cannot be derived by looking at the certificate. Within the SET protocol, the cardholder supplies the account information and the secret value to the payment gateway where the link is verified.
A certificate is only issued to the cardholder when the cardholder's issuing financial institution approves it. By requesting a certificate, a cardholder has indicated the intent to perform commerce via electronic means. This certificate is transmitted to merchants with purchase requests and encrypted payment instructions. Upon receipt of the cardholder's certificate, a merchant can be assured, at a minimum, that the account number has been validated by the card- issuing financial institution or its agent. In this specification, cardholder certificates are optional at the payment card brand's discretion.

Merchant certificates

Merchant certificates function as an electronic substitute for the payment brand decal that appears in the store window—the decal itself is a representation that the merchant has a relationship with a financial institution allowing it to accept the payment card brand. Because they are digitally signed by the merchant's financial institution, merchant certificates cannot be altered by a third party and can only be generated by a financial institution. These certificates are approved by the acquiring financial institution and provide assurance that the merchant holds a valid agreement with an Acquirer. A merchant must have at least one pair of certificates to participate in the SET environment, but there may be multiple certificate pairs per merchant. A merchant will have a pair of certificates for each payment card brand that it accepts.

Payment Gateway Certificates

Payment gateway certificates are obtained by Acquirers or their processors for the systems that process authorization and capture messages. The gateway's encryption key, which the cardholder gets from this certificate, is used to protect the cardholder's account

information. Payment gateway certificates are issued to the Acquirer by the payment brand.

Acuirer Certificates

An Acquirer must have certificates in order to operate a Certificate Authority that can accept and process certificate requests directly from merchants over public and private networks. Those Acquirers that choose to have the payment card brand process certificate requests on their behalf will not require certificates because they are not processing SET messages. Acquirers receive their certificates from the payment card brand.

Issuer Certificates

An Issuer must have certificates in order to operate a Certificate Authority that can accept and process certificate requests directly from cardholders over public and private networks. Those Issuers that choose to have the payment card brand process certificate requests on their behalf will not require certificates because they are not processing SET messages. Issuers receive their certificates from the payment card brand.

Cardholder Registration

This process comprised 6 messages between two parties: cardholder and Issuer (CA).

1. The cardholder initiates request to the CA.
2. After the CA receives message 1 from the cardholder, the CA replies. The message includes the CA's public key-exchange key certification signed by root CA, CA's signature certificate and the initial request encrypted using CA's private key.
3. The cardholder request a registration form in this message. He randomly generates a symmetric key K1, which is used to encrypt the request, and sends this along with a digital envelop including key K1 and his credit card number.
4. The CA determines the cardholder's issuing bank by the credit card number and returns the appropriate the form, which is signed by the CA and along with CA's signature certificate.
5. The cardholder generates a public/private signature key pair, two symmetric keys K2, K3 and a random number S1. He creates a message with his filled registration form, public key, and K2, and its digital signature. This message is encrypted using K3 and sent with a digital envelop including K3 and card number.
6. The CA verifies the information, then issue a digital ID to CA. The CA generates a secret value using the random number S2 generated by the CA and S1. This secret value, the account number and the expiration date further feed into a one-way hashing to generate a secret number. The CA signs the certificate includes this secret number and the cardholder's public signature key. Then, CA sends this certificate encrypted using K2 along with and its signature certificate.

This registration process includes 3 steps. The first two messages are about to get CA's public key. Once the cardholder has CA's key-exchange key, he can request a registration form in message 3 and 4. The certificate is in the last 2 messages.

Merchant Registration

The Merchant' registration is simpler than cardholder's, which include 4 messages. The first two messages are almost same as cardholder's, except in the second message the registration form has been sent. The merchant has to generate two public/private key pairs – one is for signature, the other is for key- exchange—instead of one pair compared to the cardholder.

The registration protocol has been proved to be secure [3]. But there are two risks to cause insecure. The first is that the cardholder is not required to generate a fresh signature key pair, but may register an old one. There is a risk that the old one could be compromised. And another problem is that the secret value generation mentioned above which is the exclusive-OR of numbers (S1, S2) chosen by two parties. Since exclusive-OR is invertible, a criminal working for a CA can give every cardholder the same secret value. This combination introduces some risk that a criminal can impersonate the cardholder.

These two problems are fixable. The first insecurity can be repaired in the cardholder's implementation. The second one can be fixed by replacing exclusive-OR by one-way hashing.

Security of SET

Cryptography Algorithm in SET

- Symmetric encryption
  - DES (Data Encryption Standard) : 56bit key, protect financial data
  - CDMF (Commercial Data Masking Facility) : 40 bit key, protect acquire-to cardholder message
- Asymmetric encryption and digital signature : RSA
- Hash function : SHA-1
- Message Authentication Code : HMAC (based on SHA-1)

Security Technology in SET

- Digital envelopes, nonces, salt and Dual signatures
- Two public-private key pairs for each party
- One for digital signatures; one for key exchange messages
- 160-bit message digests
- Statistically globally unique IDs (XIDs)
- Certificates (5 kinds)
- Cardholder, Merchant, Acquirer, Issuer, Payment Gateway
- Hardware cryptographic modules (for high security)
- Idempotency (message can be received many times but is only processed once) $f(f(x)) = f(x)$
- Complex protocol. Over 600 pages of detail

SET Security Recommendations

SET is not secure if its servers are not secure:

• Dedicate a machine to the Merchant Server and POS software.
• Use a firewall to insulate it from the Internet and intranet. Do not allow FTP or telnet on other ports.
• Remove all unnecessary software from the Merchant Server.
• Only SET-defined protocol ports should be open to computers outside the firewall.
• Merchant Server software should interface with POS software only through APIs.
• Need to protect transaction databases against access/alteration.

Future of SET

SET can work in Real Time or be a store and forward transfer, and is industry backed by the major credit card companies and banks. Its transaction can be accomplished over the WEB or via email. It provides confidentiality, integrity, authentication, and, or non-repudiation.

Table 3- SET Future Technology

Algorithm	Now	Near-Future	Future
Symmetric(encrypts order instruction)	DES	Triple DES	(AES)
Hash (digests message)	SHA-1	?	?
Asymmetric (data integrity for authentication; key management)	RSA	ECC (ElGamal+Diffie Hellman+DSA)	?

Confidentiality

- payment info is secure
- but order info is not secure

Data Integrity

- Uses mathematical techniques to minimize corruption or detect malicious tamper

Client Authentication

- Digital ID (certificate) used to identify costumer
- Digital ID (certificate) checked via the card's Issuer

Merchant Authentication

- Digital certificate again used as a back check for confirming the merchant is valid
- The check generally against a dB held by the issuer of the card

Interoperability

- Has not been achieved
- IBM and VeriFone (Hewlett-Packard) are working together to make their individual products interoperable.
- Results in many different "interoperable" versions of SET, instead a single protocol

SET is safe since it addresses all the parties involved in typical credit card transactions: consumers, merchants, and the banks. Besides the interoperability problem, it has difficulties to spread since it needs all the participants to have some part of the software, even very expensive hardware. It may be clearly in the interests of the credit card companies and banks, but it looks quite different from the perspective of merchants and consumers. In order to process SET transactions, the merchants have to spend several million dollars in equipment and services when they already have what are arguably sufficient security provisions in SSL. To consumers, they have to install software, “Anything that requires consumers to take an extra step deters them from adopting it,” Vernon Keenan, a senior analyst at Zona Research argues. SET is a very comprehensive and very complicated security protocol. It has to be simplified to be adopted by every parties involved, otherwise, it might be abandoned.

9. The access control server returns the user authentication information to the merchant plug-in running in the acquirer domain by redirecting the user to the merchant site. It also sends the information to the repository where the history of the user authentication is kept for legal purpose.
10. The plug-in receives the response of the access control server through the user's browser. This contains the digital signature of the access control server.
11. The plug-in validates the digital signature of the response and the response from the access control server.
12. If the authentication was successful and the digital signature of the access control server is validated, the merchant sends the authorization information to its bank (i.e., the acquirer bank).

12.4 ELECTRONIC MONEY

12.4.1 Introduction

Electronic money, which is also called as electronic cash or digital cash is one more way of making payments on the Internet. Electronic money is nothing but money represented by computer files. In other words, the physical form of money is converted into binary form of computer data. Let us first understand how one can obtain and use electronic money. For this, first take a look at Fig. 12.21. The figure shows the conceptual steps involved in electronic money processing.

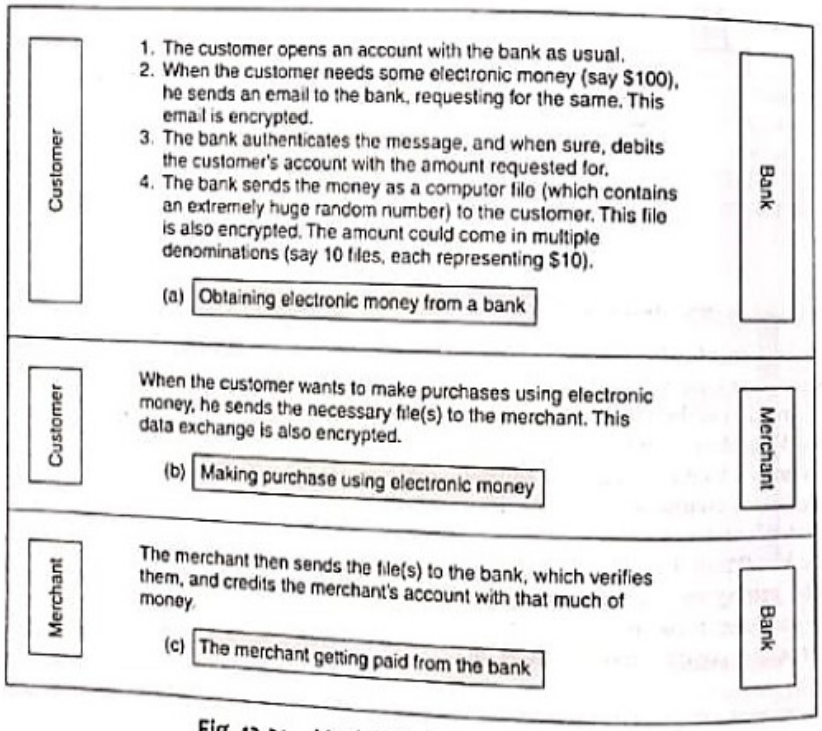


Fig. 12.21 Model of electronic money

As the figure shows, the customer obtains electronic money (which is nothing but one or more computer files) from a bank in exchange of physical money (from his account with the bank). When the customer wants to make any purchases in an electronic commerce transaction and make payments using electronic money, the customer sends these files representing electronic money to the merchant. The merchant forwards these files to the same bank, which verifies the electronic money and credits the merchant's account with the actual money equivalent to the value of the electronic money.

12.4.2 Security Mechanisms in Electronic Money

The security mechanisms in these procedures are similar to all the previous mechanisms described earlier. Let us study the process of the customer obtaining the money in the form of files from the bank. The same principles would apply in other transactions (e.g., a customer buying something from a merchant and then sending these files to him).

Step 1 Bank sends the electronic money to the customer, as shown in Fig. 12.22.

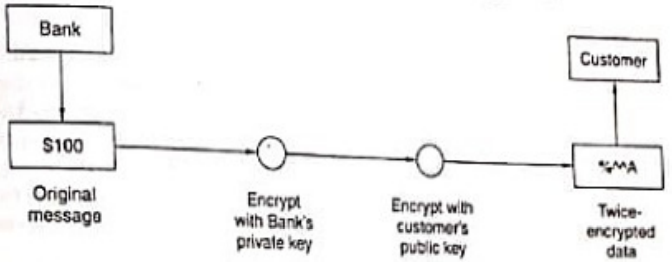


Fig. 12.22 Bank sends electronic money to the customer after encrypting it twice

As the figure shows, the bank first encrypts the original message with its own private key. It then encrypts this encrypted message further, this time with the customer's public key. Thus, the original message is encrypted twice. The bank sends this twice-encrypted message to the customer.

Step 2 The customer receives the money and decrypts it, as shown in Fig. 12.23.

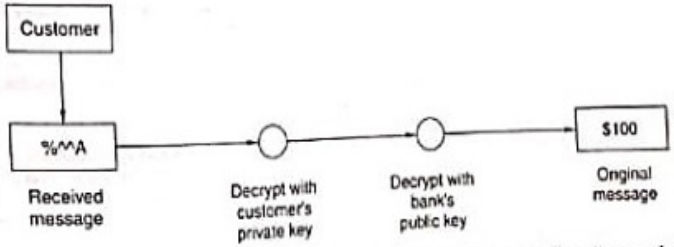


Fig. 12.23 Customer decrypts the bank's message twice to get the electronic money

Here, the customer first decrypts the received message with its own private key. Further, it decrypts the once-decrypted message using the bank's public key. Thus, the customer gets the original message back (which is $100). To ensure authentication, techniques of digital signatures and certificates may also be used in addition to these steps. We shall not describe those, as we have studied them earlier.

12.4.3 Types of Electronic Money

Electronic money can be classified in two ways. In the first classification, the types of electronic money are decided based on whether the electronic money is tracked or not. In this classification, electronic money can be of two types, **identified electronic money** and **anonymous electronic money**. In the other method of classification, it is based on whether or not the transaction is real-time. In this classification, electronic money can be either **online electronic money** or **offline electronic money**. We shall study these types now.

Classification based on the tracking of money

This classification is based on whether the electronic money is tracked throughout its lifetime. Accordingly, it can be classified as follows.

Identified electronic money Identified electronic money more or less works like a credit card. The progress of the identified electronic money from the very first time it is issued by a bank to one of its customer, up to its final return to the bank can be easily tracked by the bank. As a result, the bank can precisely know how and when the money was spent by the customer. Consequently, the bank knows who is the original customer that had requested for this money and how he spent it. How is this possible? For making electronic money identifiable like this, the file containing the information about the electronic money contains a unique serial number that is generated by the bank itself. Therefore, the bank has a list of these serial numbers vis-à-vis the customer who requested for that money.

Now, suppose the serial number generated by the bank for electronic money worth $100 is say SR100. Suppose the customer who requested for this electronic money now spends these $100 by sending the corresponding files to a merchant. The merchant would go back to the bank to redeem the electronic money and get real money, instead. At this point, the bank again has the electronic money with the serial number SR100. Therefore, it knows that the customer has bought something worth $100 from a specific merchant on a specific date. This is shown in Fig. 12.24.

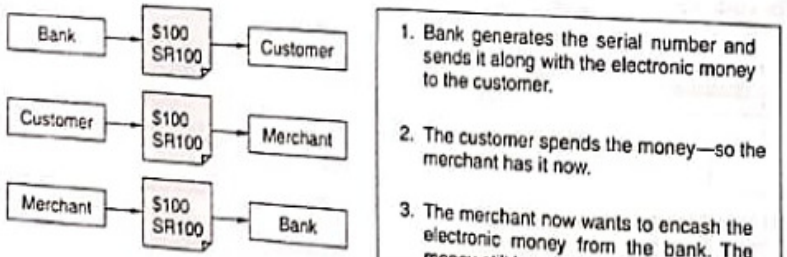


Fig. 12.24 Steps involved in identified electronic money

Since the entire journey of identified electronic money is traceable, this can create privacy issues.

Anonymous electronic money The anonymous electronic money (also called as blinded money) works like real hard cash. There is no trace of how the money was spent. There is no trail of the transactions involved in this type of electronic money. Products like *DigiCash* provide this kind of electronic money to Internet users to spend, by tying up with banks.

The key difference between identified electronic money and anonymous electronic money (which creates the anonymity) is the fact that whereas in case of the identified electronic money the bank creates the serial number, in case of the anonymous electronic money, it is the customer who creates the serial number. The process of the customer generating the random number is as follows.

1. The customer generates a random number by some mathematical algorithm. The customer then multiplies it by another huge number (called as the blinding factor).
2. The customer sends the resulting number, called as blinded number to the bank.
3. The bank does not know about the original number of step (1).
4. Bank signs (i.e., encrypts) the blinded number and sends it back to the customer.
5. The customer converts the blinded number back to the original number using some algorithm.
6. The customer then uses the original number (and not the blinded number) when making any transaction with a merchant.
7. The merchant's encashment request to the bank is also with the original number.
8. The bank cannot trace this electronic money as it does not know the relationship between the original number and the blinded number.

This process is shown in Fig. 12.25.

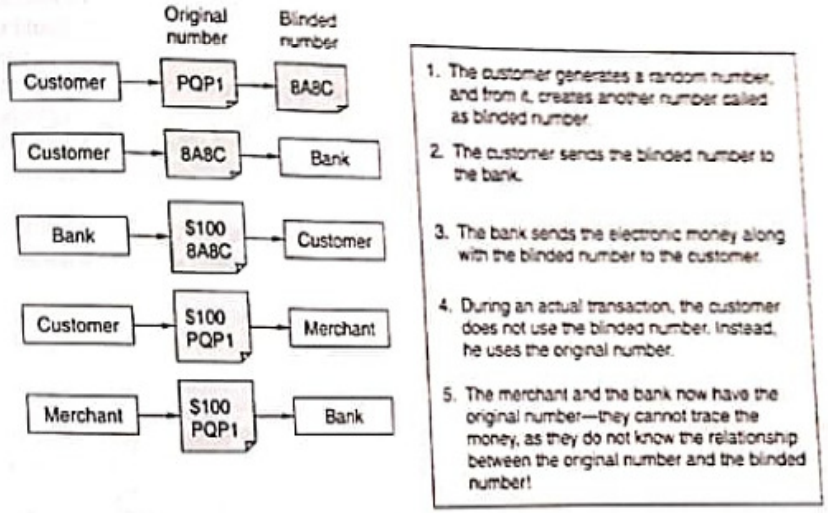


Fig. 12.25 Steps involved in anonymous electronic money

In the case of identified electronic money, the chances of a customer trying to spend the same money more than once can be easily caught or prevented. This is possible because the bank maintains a list of the issued and spent serial numbers. Therefore, it can catch attempts of spending the same piece of electronic money more than once.

Classification based on the involvement of the bank in the transaction

Based on involvement (or otherwise) of the bank in the actual transaction, electronic money can be further classified into two categories: **online electronic money** and **offline electronic money.**

Online electronic money In this type, the bank must actively participate in the transaction between the customer and the merchant. That is, before the purchase transaction of the customer can complete, the merchant would confirm from the bank in real time as to whether the electronic money offered by the customer is acceptable (e.g., ensuring that it is not already spent, or that the serial number for it is valid).

Offline electronic money In this type, the bank does not participate in the transaction between the customer and the merchant. That is, the customer purchases something from the merchant and offers to pay by electronic money. The merchant accepts the electronic money, but does not validate it online. The merchant might collect a group of such electronic money transactions and process them together at a fixed time every day.

12.4.4 The Double Spending Problem

Now, if we combine the two ways of classifying electronic money, we have four possibilities:

1. Identified Online electronic money
2. Identified Offline electronic money
3. Anonymous Online electronic money
4. Anonymous Offline electronic money

Of the four, the last type can create the double spending problem. More specifically, a customer could arrange for anonymous electronic money by using the blinded money concept. Later on, he could spend it offline more than once in quick succession (say, in the same hour) with two different merchants. Since the bank is not involved in any of the two online transactions, the fact that the same piece of money is being spent cannot be prevented. Moreover, when it is realized that the same piece of money is spent more than once (when both merchants send their daily transaction lists to the bank), the bank cannot determine which customer spent it more than once, because of the blinding factor (recall our discussion of anonymous electronic money). Consequently, *anonymous offline electronic money* is of little practical use.

Double spending problem can happen in case of *identified offline electronic money* as well. However, upon detection, the customer under question can be easily tracked from the serial numbers of the electronic money. This is shown in Fig. 12.26.

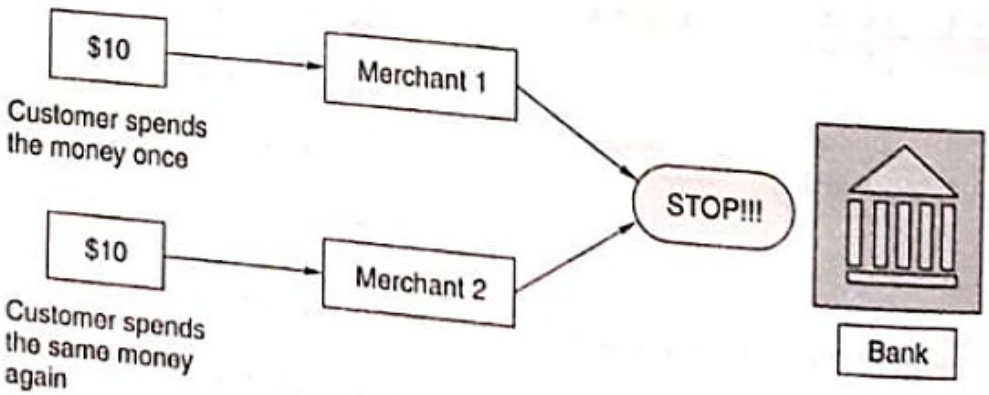


Fig. 12.26 Detection of double spending problem