Name of the Paper: Cryptography and Network Security
Paper Code: CS704C
Contact (Periods/Week):3L/Week
Credit Point: 3
No. of Lectures: 35

Prerequisite:

- Knowledge of Computer Networksand Operating Systems fundamentals
- Understanding of Discreet Mathematics concepts

Course Objective(s)

- To impart concepts on cryptography and Network security
- To gain knowledge of the standard algorithms used to provide confidentiality, integrity, and authenticity
- To recognize the various key distribution and management systems for security of a cryptosystem

Course Outcomes

CS704C.1 To understand the basic concepts in cryptography
CS704C.2 To apply the deployment of different encryption techniques to secure messages in transit across data networks
CS704C.3 To discuss various techniques used to assure Integrity and Authentication
CS704C.4 To analyze diverse security measures and issues in practice

Module -1: [6L]
INTRODUCTION AND NUMBER THEORY
Introduction - Services, Mechanisms, and Attacks, OSI security architecture, Network security model[1L]
Classical Encryption techniques (Symmetric cipher model, substitution techniques, transposition techniques, steganography)[1L]
Finite Fields and Number Theory: Groups, Rings, Fields, Modular arithmetic, Euclid's algorithm[2L]
Finite fields, Polynomial Arithmetic, Prime numbers, Fermat‟s and Euler‟s theorem[1L]
Testing for primality -The Chinese remainder theorem - Discrete logarithms [1L]

Module -2: [8L]
BLOCK CIPHERS AND PUBLIC KEY CRYPTOGRAPHY Data Encryption Standard- Block cipher principles, block cipher modes of operation[2L]
Advanced Encryption Standard (AES), Triple DES, Blowfish, RC5 algorithm[2L]
Public key cryptography: Principles of public key cryptosystems, The RSA algorithm[2L]
Key management - Diffie Hellman Key exchange, Elliptic curve arithmetic, Elliptic curve cryptography [2L]

Module 3: [6L]
HASH FUNCTIONS AND DIGITAL SIGNATURES
Authentication requirement, Authentication function, MAC, Hash function[2L]

Security of hash function and MAC, MD5, SHA, HMAC, CMAC[2L]
Digital signature and authentication protocols, DSS, ElGamal, Schnorr [2L]

Module -4: [9L]
SECURITY PRACTICE AND SYSTEM SECURITY
Authentication applications, Kerberos, X.509[1L]
Authentication services, Internet Firewalls for Trusted System: Roles of Firewalls[1L]
Firewall related terminology- Types of Firewalls[1L]
Firewall designs principles, SET for E-Commerce Transactions[2L]
Intruder, Intrusion detection system[1L]
Virus and related threats, Countermeasures[1L]
Trusted systems, Practical implementation of cryptography and security [2L]

Module -5: [6L]
E-MAIL, IP,AND WEB SECURITY
E-mail Security: Security Services for E-mail-attacks possible through E-mail, Establishing keys privacy, authentication of the source [1L]
Message Integrity, Non-repudiation, Pretty Good Privacy, S/MIME[1L]
IP Security: Overview of IPSec, IPv4 and IPv6-Authentication Header, Encapsulation Security Payload (ESP)[1L]
Internet Key Exchange (Phases of IKE, ISAKMP/IKE Encoding)[1L]
Web Security: SSL/TLS Basic Protocol, computing the keys, client authentication[1L]
PKI as deployed by SSL Attacks fixed in v3, Exportability, Encoding, Secure Electronic Transaction[1L]

Text Books:
1. Atul Kahate, “Cryptography and Network Security”, Third edition, McGraw Hill Education

Recommended books:
2. William Stallings, “Cryptography and Network Security: Principles and Practice”, Sixth edition, Pearson
3. Behrouz A. Forouzan, Debdeep Mukhopadhyay, “Cryptography and Network Security”, Second edition, McGraw Hill Education

CO-PO Mapping

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CS704C.1	3		2			3						
CS704C.2	3	2	3	2	3							
CS704C.3		3	2	3								
CS704C.4	2	3		3		3						

Module -1: [6L]

INTRODUCTION AND NUMBER THEORY

LECTURE 1:

1. Introduction

When computer data travels in a network, there exist several threats to the data such as modification, forging, etc. Cryptography is the method of transforming a message at the sender to unreadable format to protect it from unauthorized access in transit. At the receiver, the unreadable message is returned back to its original form.

There are three aspects of information security which needs to be considered:

- **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 the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks and they make use of one or more security mechanisms to provide the service

1.1. Security Services

- **Confidentiality** – This principle ensures that only the sender and the intended receiver(s) have access to the content of the message. Interception causes loss of message confidentiality.
- **Authentication –** This process ensures that the origin of a message or electronic document is correctly identified. Hence, it helps establish the proof of identities. Fabrication is possible in absence of proper authentication mechanisms.
- **Integrity –** This ensures that the intended recipient receives the message as send by the sender with no change. Modification of the message results in loss of message integrity.
- **Non repudiation –** Does not allow the sender of a message to refuse the claim of not sending that message.
- **Access control –** The principle of access control specifies and controls who can access what information. Access control is broadly related to two areas: role management which concentrates on the user side (which user can do what) and rule management which focuses on the resources side (which resource is accessible under what circumstances).
- **Availability –** The principle of availability states that resources should be available to authorized people at all times. Interruption is an attack on availability.

1.2. Security Mechanisms

A security mechanism is designed to detect, prevent, or recover from a security attack as no single mechanism exists which will support all functions required. Our main focus of security mechanisms in use is **Cryptographic Techniques**. Some of the mechanisms are:

1. Encipherment 2. Digital Signature 3. Access Control

Security Mechanisms (X.800)

- Specific security mechanisms - encipherment, digital signatures, access controls, data integrity, authentication exchange, traffic padding, routing control, notarization
- Pervasive security mechanisms - trusted functionality, security labels, event detection, security audit trails, security recovery

1.3. Security Attacks

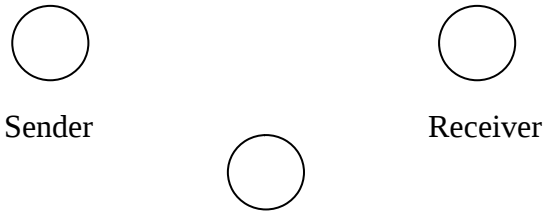
There are four general categories of attack which are listed below.

- **Interruption**

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

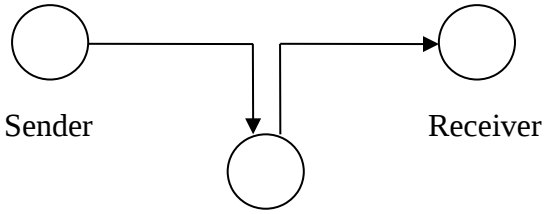
- **Interception**

  An unauthorized party gains access to an asset. This is an attack on confidentiality. Unauthorized party could be a person, a program or a computer. Examples - wire tapping to capture data in the network, illicit copying of files



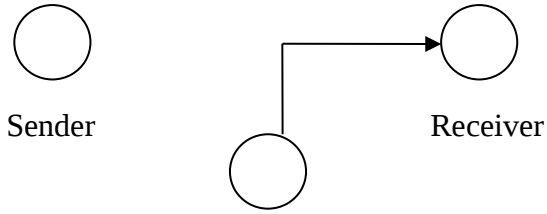
- **Modification**

  An unauthorized party not only gains access to but tampers with an asset. This is an attack on integrity. Examples - changing values in data file, altering a program, modifying the contents of messages being transmitted in a network.



- **Fabrication**

  An unauthorized party inserts counterfeit objects into the system. This is an attack on authenticity. Examples - insertion of spurious message in a network or addition of records to a file.



1.4. Cryptographic Attacks

1. **Passive Attacks**

   Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Passive attacks are of two types:

- **Release of message contents**: A telephone conversation, an e-mail message and a transferred file may contain sensitive or confidential information. We would like to prevent the opponent from learning the contents of these transmissions.
- **Traffic analysis**: Attempts of analyzing encoded messages to come up with likely patterns Passive attacks are very difficult to detect because they do not involve any alteration of data. However, it is feasible to prevent the success of these attacks.

2. Active attacks

These attacks involve some modification of the data stream or the creation of a false stream. These attacks can be classified in to four categories:

- **Masquerade** – One entity pretending to be an another entity
- **Modification attacks** – Are classified into replay attacks and alteration of messages
  Replay attacks - Involves passive capture of a data unit and its subsequent transmission to produce an unauthorized effect.
  Alteration of messages – Involves passive capture of a data unit and its subsequent modification
- **Denial of service** – Prevents or inhibits the normal use or management of communication facilities. Another form of service denial is the disruption of an entire network, either by disabling the network or overloading it with messages so as to degrade performance.

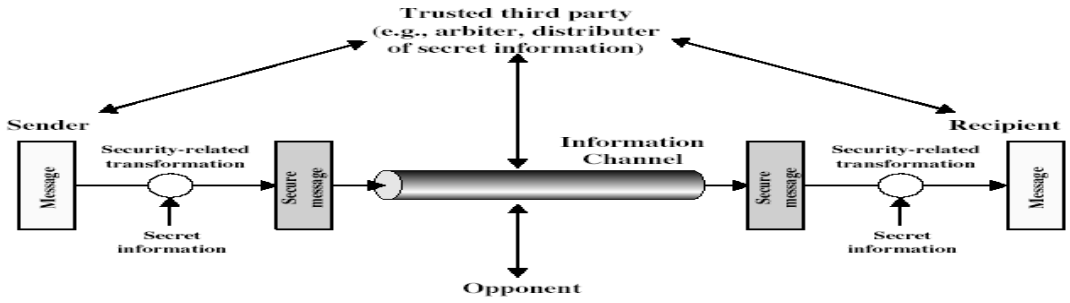
It is quite difficult to prevent active attacks absolutely, because to do so would require physical protection of all communication facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them.

1.5. OSI Security Architecture

ITU-T X.800 Security Architecture for OSI defines a systematic way of defining and providing security requirements. This also defines seven layers of security in the form of:

- Authentication
- Access Control
- Non repudiation
- Data Integrity
- Confidentiality
- Assurance or Availability
- Notarization or Signature

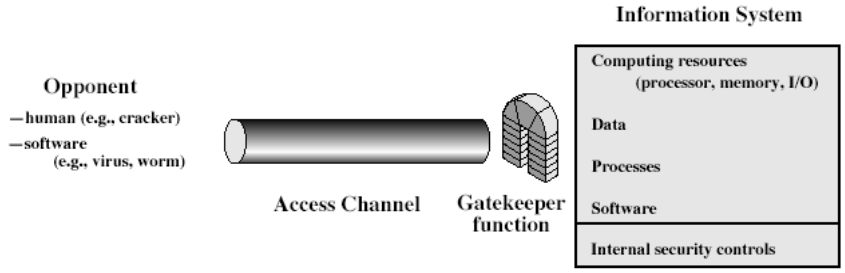
1.6. Network Security Model



This model requires the following:

- Design a suitable algorithm for the security transformation
- Generate the secret information (keys) used by the algorithm
- Develop methods to distribute and share the secret information

– Specify a protocol enabling the principals to use the transformation and secret information for a security service



- Using this model requires us to:
  – select appropriate gatekeeper functions to identify users
  – implement security controls to ensure only authorised users access designated information or resources
- Trusted computer systems can be used to implement this model

LECTURE 2:

There are two basic building blocks of all encryption techniques: substitution and transposition.

2.1 Substitution Techniques

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns.

2.1.1 Caesar cipher (or) shift cipher

The earliest known use of a substitution cipher and the simplest was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet.

Plain Text : pay more money

Cipher text: SDB PRUH PRQHB

Note that the alphabet is wrapped around, so that letter following “z” is “a”.

For each plaintext letter **p**, substitute the cipher text letter **c** such that

c = E(p) = (p+3) mod 26

A shift may be any amount, so that general Caesar algorithm
is **c = E(p) = (p+k) mod 26**
where **k** takes on a value in the range 1 to 25. The decryption algorithm is simply
p = D(c) = (c-k) mod 26

2.1.2 Playfair cipher

The best known multiple letter encryption cipher is the playfair.

The playfair algorithm is based on the use of 5x5 matrix of letters constructed using a keyword. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetical order.
The letter **i** and **j** are counted as one letter. Two letters at a time are encrypted according to the following rules:

- Repeating plaintext letters that would fall in the same pair are separated with a Filler letter such as **x**
- Plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row following the last.
- Plaintext letters that fall in the same column are replaced by the letter beneath, with the top element of the column following the last.
- Otherwise, each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

Example

Keyword - MONARCHY
Plain Text - meet me at the school house

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

Splitting two letters as a unit => me et me at th es ch ox ol ho us ex
Corresponding cipher text => CL KL CL RS PD IL HY AV MP HF XL IU

Strength of playfair cipher

Playfair cipher is a great advance over simple mono alphabetic ciphers. Since there are 26 letters, 26x26 = 676 diagrams are possible, so identification of individual diagram is more difficult.

2.1.3 Polyalphabetic ciphers

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is polyalphabetic cipher. All the techniques have the following features in common.

- A set of related monoalphabetic substitution rules are used
- A key determines which particular rule is chosen for a given transformation

2.1.4 Vigenere cipher

In this scheme, the set of related monoalphabetic substitution rules consisting of 26 caesar ciphers with shifts of 0 through 25. Each cipher is denoted by a key letter. e.g., Caesar cipher with a shift of **3** is denoted by the key value **d** (since a=0, b=1, c=2 and so on). To aid in understanding the scheme, a matrix known as **Vigenere** tableau is constructed.

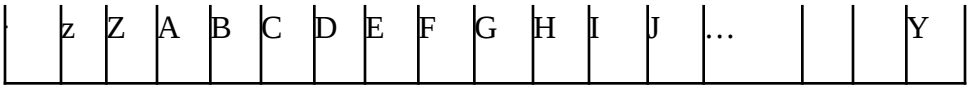
Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left. A normal alphabet for the plaintext runs across the top. The process of encryption is simple:
Given a key letter x and a plaintext letter y, the cipher text is at the intersection of the row labeled x and the column labeled y; in this case, the ciphertext is V.

To encrypt a message, **a key is needed that is as long as the message**. Usually, the key is a repeating keyword.
Example
Key = d e c e p t i v e d e c e p t i v e d e c e p t i v e
Plain Text = we are discovered save yourself
Cipher Text = ZICVTWQNGRZGVTWAVZHCQYGLMGJ

	PLAIN TEXT															
K		a	b	c	d	e	f	g	h	i	j	k	…	x	y	z
E	a	A	B	C	D	E	F	G	H	I	J	K	…	X	Y	Z
Y	b	B	C	D	E	F	G	H	I	J	K	L	…	Y	Z	A
	c	C	D	E	F	G	H	I	J	K	L	M	…	Z	A	B
L	d	D	E	F	G	H	I	J	K	L	M	N	…	A	B	C
E	e	E	F	G	H	I	J	K	L	M	N	O	…	B	C	D
T	f	F	G	H	I	J	K	L	M	N	O	P	…	C	D	E
T	g	G	H	I	J	K	L	M	N	O	P	Q	…	D	E	F
E R	: :	: :	: :	: :	: :	: :	: :	: :	: :	: :	: :	: :	…	: :	: :	: :
S	x	X	Y	Z	A	B	C	D	E	F	G	H	…			W
	y	Y	Z	A	B	C	D	E	F	G	H	I	…			X

.

.



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

Strength of Vigenere cipher

- There are multiple cipher text letters for each plaintext letter
- Letter frequency information is obscured

2.1.5 One Time Pad Cipher

It is an unbreakable cryptosystem. It represents the message as a sequence of 0s and 1s. This can be accomplished by writing all numbers in binary, for example, or by using ASCII. The key is a random sequence of 0s and 1s of same length as the message. Once a key is used, it is discarded and never used again. The system can be expressed as follows:

$C_i = P_i \oplus K_i$ where

C_i - i^{th} binary digit of cipher text

P_i - i^{th} binary digit of plaintext

K_i - i^{th} binary digit of key

Thus, the cipher text is generated by performing the bitwise XOR of the plaintext and the key. Decryption uses the same key. Because of the properties of XOR, decryption simply involves the same bitwise operation:

$$P_i = C_i \oplus K_i$$

Example -

Plain Text – 00101001

Key – 10101100

Cipher Text – 10000101

Advantage:

Encryption method is completely unbreakable for a ciphertext only attack.

Disadvantages

It requires a very long key which is expensive to produce and expensive to transmit.

Once a key is used, it is dangerous to reuse it for a second message; any knowledge on the first message would give knowledge of the second.

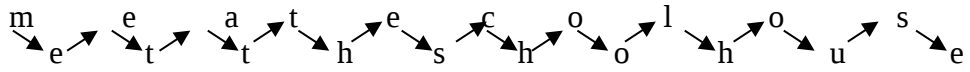
2.2 TRANSPOSITION TECHNIQUES

All the techniques examined so far involve the substitution of a cipher text symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

2.2.1 Rail fence

It is simplest of such cipher, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

The encrypted message is
Plain Text – meet at the school house
To encipher this message with a rail fence of depth 2, rewrite the message as follows:



MEATECOLOSETTHSHOHUE

2.2.2 Row Transposition Ciphers-
A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of columns then becomes the key of the algorithm.
Example

Plaintext	= meet at the school house							
Key	=	4	3	1	2	5	6	7
		m	e	e	t	a	t	t
		h	e	s	c	h	o	o
		l	h	o	u	s	e	

CT = ESOTCUEEHMHLAHSTOETO

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is more complex permutation that is not easily reconstructed.

2.2.3 Feistel cipher structure
The input to the encryption algorithm are a plaintext block of length **2w** bits and **a key K**. the plaintext block is divided into two halves $\mathbf{L_0}$ and $\mathbf{R_0}$. The two halves of the data pass through **"n"** rounds of processing and then combine to produce the ciphertext block. Each round **"i"** has inputs $\mathbf{L_{i-1}}$ and $\mathbf{R_{i-1}}$, derived from the previous round, as well as the sub key $\mathbf{K_i}$, derived from the overall key **K**. in general, each sub key $\mathbf{K_i}$ are different from **K** and from each other.

All rounds have the same structure. A substitution is performed on the left half of the data (as similar to S-DES). This is done by applying a round function **F** to the right half of the data and then taking the **XOR** of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round sub key $\mathbf{K_i}$. Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network. The exact realization of a Feistel network depends on the choice of the following parameters and design features:

Block size - Increasing size improves security, but slows cipher

Key size - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher

Number of rounds - Increasing number improves security, but slows cipher

Sub key generation - Greater complexity can make analysis harder, but slows cipher

Round function - Greater complexity can make analysis harder, but slows cipher

Fast software en/decryption and ease of analysis - are more recent concerns for practical use and testing.

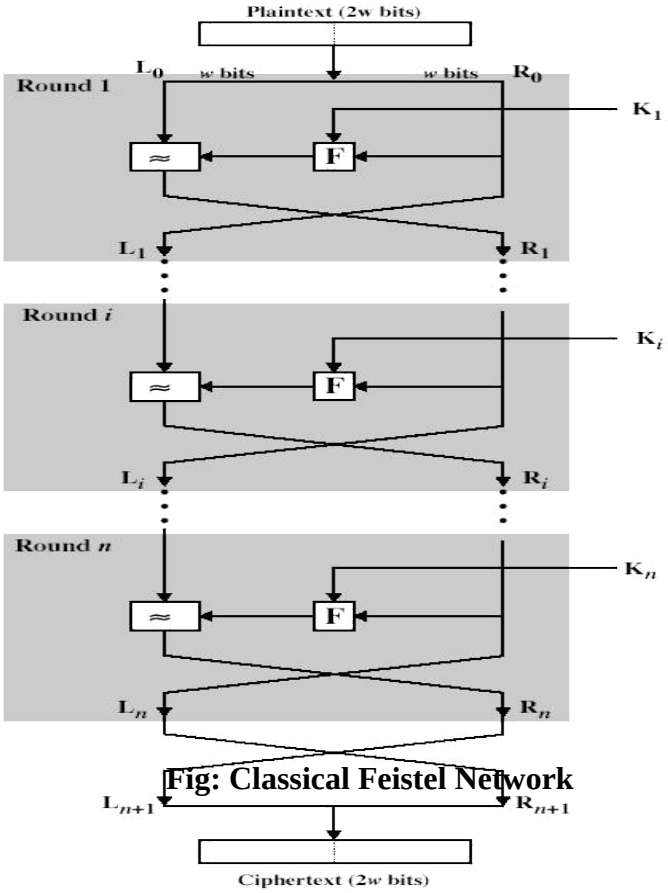


Fig: Classical Feistel Network

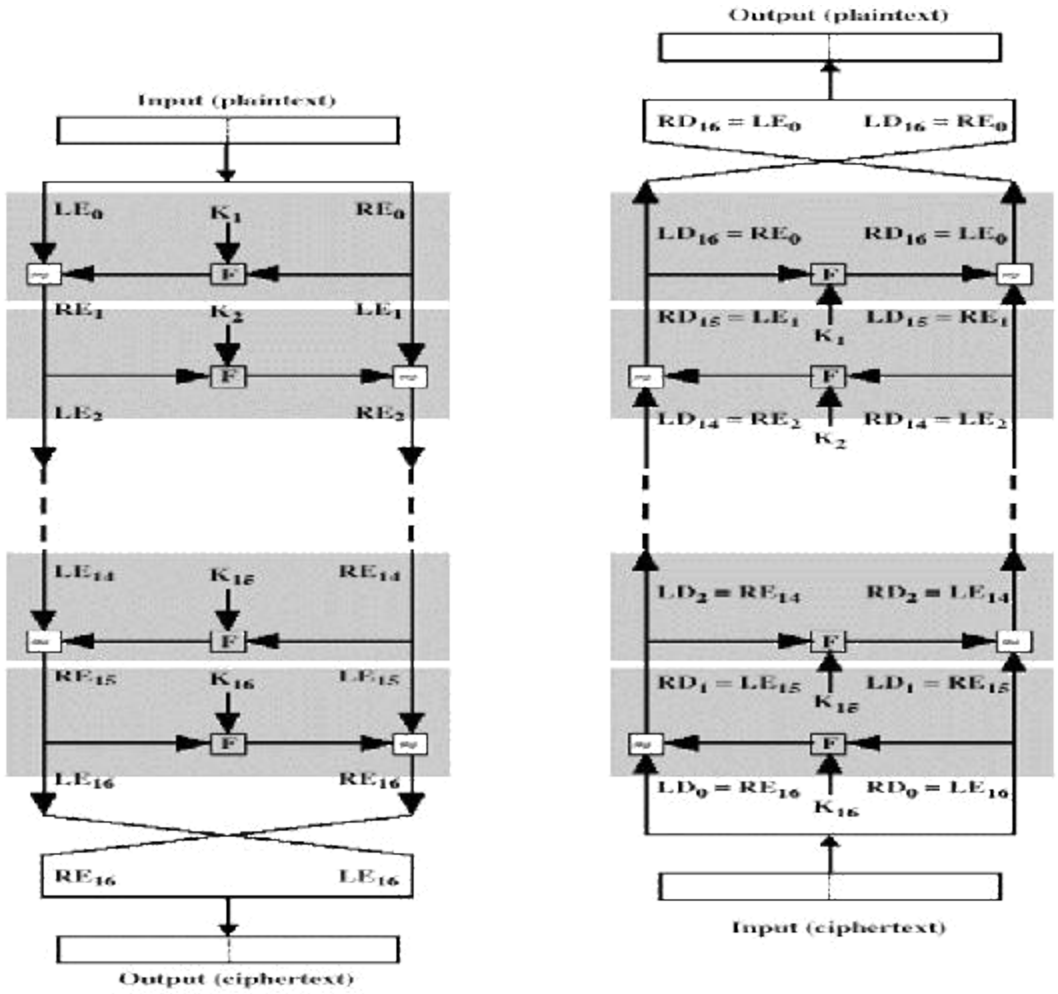


Fig: Feistel encryption and decryption

The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the sub key $\mathbf{K_i}$ in reverse order. i.e., $\mathbf{K_n}$ in the first round, $\mathbf{K_{n-1}}$ in second round and so on. For clarity, we use the notation $\mathbf{LE_i}$ and $\mathbf{RE_i}$ for data traveling through the decryption algorithm. The diagram below indicates that, at each round, the intermediate value of the decryption process is same (equal) to the corresponding value of the encryption process with two halves of the value swapped.

i.e., $\mathbf{RE_i} \,||\, \mathbf{LE_i}$ (or) equivalently $\mathbf{RD_{16-i}} \,||\, \mathbf{LD_{16-i}}$

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is $\mathbf{RE_{16}} \,||\, \mathbf{LE_{16}}$. The output of that round is the cipher text. Now take the cipher text and use it as input to the same algorithm.

The input to the first round is $\mathbf{RE_{16}} \,||\, \mathbf{LE_{16}}$, which is equal to the 32-bit swap of the output of the sixteenth round of the encryption process.

Now we will see how the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process.

In general, for the i^{th} iteration of the encryption algorithm,

$\mathbf{LE_i = RE_{i-1}}$

$\mathbf{RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)}$

Finally, the output of the last round of the decryption process is $\mathbf{RE_0 \| LE_0}$. A 32-bit swap recovers the original plaintext.

LECTURE 3:
These are basic notions of abstract algebra, which is widely used in cryptography.

3.1 Groups
A group **G**, sometimes denoted by **{G, $\bullet$ }**, is a set of elements with a binary operation, denoted by $\bullet$, that associates to each ordered pair **(a,b)** of elements in **G** an element **(a $\bullet$ b)** in **G**, such that the following axioms are obeyed:

(A1) Closure: If a and belong to G, then a $\bullet$ b is also in G
(A2) Associative: a $\bullet$ (b $\bullet$ c)= (a $\bullet$ b) $\bullet$ c for all a,b,c in G
(A3) Identity element: There is an element e in G such that a $\bullet$ e= e $\bullet$ a=a for all a in G
(A4) Inverse element: For each a in G there is an element a' in G such that a $\bullet$ a' = a' $\bullet$ a=e

Example*: set S_N of permutations on the set {1,2,..,N} with operation $\bullet$ - composition of permutations is a group with e=(1,2,..,N). For N=3, (123) $\bullet$ (321)=(321); (213) $\bullet$ (132)=(312)*

If a group has a finite number of elements, it is referred to as a **finite group**, and the **order** of the group is equal to the number of elements in the group. Otherwise, the group is **infinite group.**

A group is said to be **abelian** if it satisfies the following additional condition:
(A5) Commutative: a $\bullet$ b=b $\bullet$ a for all a,b in G

The set of integers (positive, negative, and 0) under addition is an abelian group. The set of real numbers under multiplication is an abelian group.
The set S_N of permutations is not an abelian group.

When the group operation is addition, the identity element is 0; the inverse element of a is –a; and the subtraction is defined as: a-b=a+(-b).

Exponentiation within a group is defined as repeated application of the group operation, so that $a^3 = a \bullet a \bullet a$. We also define $a^0 = e$, the **identity element**, and $a^{-n} = (a')^n$, where a' is **inverse element** for a.

A **group G is cyclic** if every element of G is a power a^k (k – integer) of a fixed element $a \in G$. The element a is said to **generate** the group G, or to be a **generator** of G.

A cyclic group is always abelian, and may be finite or infinite.

3.2 Rings

A **ring** R, sometimes denoted by **{R,+, ×}**, is a set of elements with two binary operations, called addition and multiplication, such that for all **a, b, c** in **R** the following axioms are obeyed:

(A1-A5) R is an abelian group with respect to addition; that is, R satisfies axioms A1 through A5. For this case of an additive group we denote the identity element as 0 and the inverse of a as –a.

(M1) Closure under multiplication: If a and b belong to R, then ab is also in R (multiplication, as usually, is shown by concatenation of its operands)
(M2) Associativity of multiplication: a(bc)=(ab)c
(M3) Distributive laws: a(b+c)=ab+ac
(a+b)c=ac+bc
With respect to addition and multiplication, the set of all n-square matrices over the real numbers is a ring R.

The ring is said to be commutative if it satisfies the following additional condition:
(M4) Commutativity of multiplication: ab=ba

Let S be the set of all even integers under the usual operations of addition and multiplication. S is a commutative ring. The set of all n-square matrices over the real numbers is not a commutative ring.
We define integral domain, which is commutative ring that obeys the following axioms:

(M5) Multiplicative identity: There is an element 1 such that a1=1a=a for all a in R

(M6) No zero divisors: If a,b in R and ab=0, then, either a=0 or b=0.
Let S be the set of integers, positive, negative, and 0, under the usual operations of addition and multiplication. S is an integral domain.

3.3 Field:

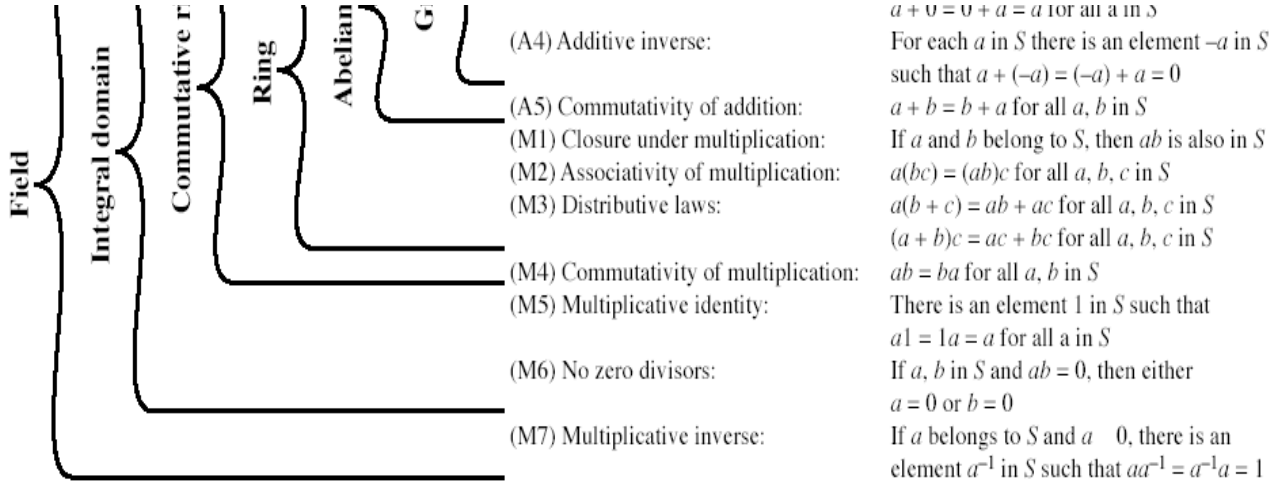
A field F, sometimes denoted by {F,+, ×}, is a set of elements with two operations, called addition and multiplication, such that for all a, b, c in F the following axioms are obeyed:

(A1-M6) F is an integral domain; that is, F satisfies axioms A1 through A5 and M1 through M6.
(M7) Multiplicative inverse: For each a in F, except 0, there is an element a^{-1} *in F, such that* $aa^{-1} = a^{-1}a = 1$

In essence, a field is a set in which we can do addition, subtraction, multiplication and division without leaving the set. Division is defined as: $a / b = a(b^{-1})$

Examples of fields are the rational numbers, real numbers, complex numbers. Set of all integers is not a field, because not every element of the set has a multiplicative inverse; in fact, only the elements 1 and -1 have multiplicative inverses in integers.

Fig. summarises axioms that define groups, rings and fields



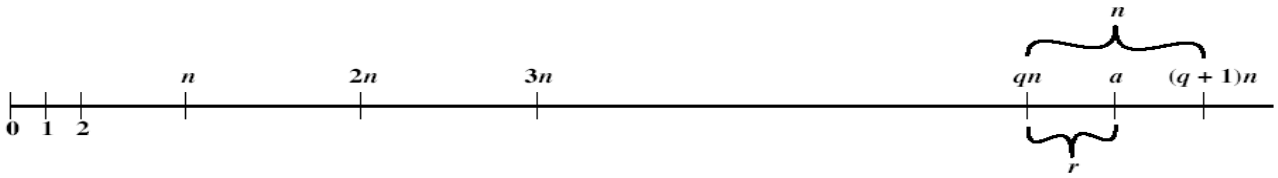
3.4 Modular Arithmetic

3.4.1 Introduction

Given any positive integer n and any integer a, if we divide a by n, we get an integer quotient q and an integer remainder r that obey the following relationship:

a=qn+r $0 \le r < n; q = \lfloor a/n \rfloor$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x.



The remainder r is often referred to as a residue.

a=11, n=7, 11=1x7+4, r=4

a=-11, n=7, -11=(-2)x7+3, r=3

If **a** is an integer and **n** is a positive integer, we define **a mod n** to be the remainder when **a** is divided by **n**, Thus, for any integer **a**, we can always write

$a = \lfloor a/n \rfloor \times n + (a \bmod n)$

11mod7=4, -11mod7=3

Two integers **a** and **b** are said to be congruent modulo **n**, if **a mod n** = **b mod n**. This is written as $a \equiv b \bmod n$.

$73 \equiv 4 \bmod 23; 21 \equiv -9 \bmod 10$

3.4.2 Divisors

We say that a nonzero **b** divides **a** if **a=mb** for some **m**, where **a, b**, and **m** are integers. That is, **b** divides **a** if there is no remainder on division. The notation **b|a** is commonly used to mean **b** divides **a**. Also, if **b|a**, we say that **b** is a divisor of **a**.

Example -

The positive divisors of 24 are 1,2,3,4,6,8,12,24.

The following relations hold:

- If a|1 then a= ± 1
- If a|b and b|a, then a= $\pm$b
- Any b $\neq$0 divides 0
- If b|g and b|h then b|(mg+nh) for arbitrary integers m and n
  Proof:
  If b|g, then g=b $\times$g1 for some integer g1
  If b|h, then h=b $\times$h1 for some integer h1
  So,
  mg+nh=mbg1+nbh1=b $\times$(mg1+nh1) and therefore b divides mg+nh

Example:
B=7,g=14, h=63, m=3, n=2
7|14 and 7|63. To show: 7|(3 $\times$14+2 $\times$63)
We have (3 $\times$14+2 $\times$63)=7(3 $\times$2+2 $\times$9)
And it is obvious that 7|7(3 $\times$2+2 $\times$9)

Note that if a $\equiv$0modn, then n|a.

3.4.3 Properties of the Modulo operator

1. a $\equiv$b mod n if n|(a-b)
2. a $\equiv$b mod n implies b $\equiv$a mod n
3. a $\equiv$b mod n and b $\equiv$c mod n imply a $\equiv$c mod n

To demonstrate the 1st point, if n|(a-b), then a-b=kn for some k. So we can write a=b+kn. Therefore, (a mod n) (remainder when b+kn is divided by n) = (remainder when b is divided by n) = (b mod n)

Example:
23 $\equiv$8 mod 5 because 23-8=15=5x3
-11 $\equiv$5 mod 8 because -11-5=-16=8x(-2)
81 $\equiv$0 mod 27 because 81-0=81=27x3

3.4.4 Modular arithmetic operations

Properties of modular arithmetic, working over {0,1,.., n-1}:

1. [(a mod n) + (b mod n)] mod n = (a+b) mod n
2. [(a mod n) - (b mod n)] mod n = (a-b) mod n
3. [(a mod n) x (b mod n)] mod n = (ab) mod n

1st property is demonstrated below.
Let (a mod n) = r_a and (b mod n) = r_b.
So, a=r_a+jn for some integer j and b= r_b+kn for some integer k, can be written.
Then, (a+b) mod n = (r_a+jn+ r_b+kn) mod n = (r_a+ r_b+(k+j)n) mod n = (r_a+ r_b) mod n = [(a mod n) + (b mod n)] mod n

11 mod 8 =3, 15 mod 8 =7
[(11 mod 8) + (15 mod 8)] mod 8 = 10 mod 8 =2

(11+15) mod 8 = 26 mod 8 = 2
[(11 mod 8) - (15 mod 8)] mod 8 = -4 mod 8 =4
(11-15) mod 8 = -4 mod 8 = -4
[(11 mod 8) x (15 mod 8)] mod 8 = 21 mod 8 =5
(11x15) mod 8 = 165 mod 8 = 5

Exponentiation is performed, as in ordinary arithmetic
To find 11^7 mod 13, we can proceed as follows:

$$11^2 = 121 \equiv 4 \bmod 13$$

$$11^4 \equiv 4^2 \equiv 3 \bmod 13$$

$$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \bmod 13$$

Thus, the rules for ordinary arithmetic involving addition, subtraction, and multiplication carry over into modular arithmetic.

3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Additive and multiplicative inverses modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

Table 3.1 introduces arithmetic modulo 8.
We see that not for all elements exist multiplicative inverses (for 2, 4, 6).

LECTURE 4

4.1 Properties of modular arithmetic

Let Zn ={0,1,..,n-1}. This is referred to as the set of residues, or residue class modulo n. To be more precise, each integer in Zn represents a residue class. We can label the residue classes modulo n as [0], [1],.., [n-1], where [r]={a: a is integer, a r mod n.

Of all the integers in the residue class, the smallest nonnegative integer is the one usually used to represent the residue class. Finding the smallest nonnegative integer to which k is congruent modulo n is called reducing k modulo n.

If we perform modulo arithmetic within Zn, the properties shown in Table 4.1 hold for integers in Zn. Thus, Zn is a commutative ring with a multiplicative identity element.

Commutative laws	$(w \times x) \bmod n = (x \times w) \bmod n$
Associative laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive laws	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$ $[w + (x \times y)] \bmod n = [(w + x) \times (w + y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive inverse ($-w$)	For each $w \in Z_n$, there exists a z such that $w + z = 0 \bmod n$

Table 4.1: Properties of Modular Arithmetic for Integers in Z_n

There is one peculiarity of modular arithmetic that sets it apart from ordinary arithmetic. First, observe that, as in ordinary arithmetic, we can write

If (a+b) (a+c) mod n, then b c mod n (4.1)
(5+23) (5+7) mod 8, then 23 7 mod 8

Equation (4.1) is consistent with the existence of an additive inverse. Adding the additive inverse of a to both sided of (4.1), we have

((-a)+a+b) ((-a)+a+c) mod n
b c mod n

However, the following statement is true only with the attached condition:

If (a b) (a c) mod n then b c mod n if a is relatively prime to b (4.2)

Where the term relatively prime is defined as follows: Two integers are relatively prime if their only common positive integer factor is 1. Similar to the case of equation (4.1), we can say that (4.2) is consistent with the existence of a multiplicative inverse of a. Applying the multiplicative inverse of a to both sides of (4.2), we have

To see this, consider an example, in which condition does not hold:

6x3=18 2 mod 8
6x7=42 2 mod 8
Yet mod 8 because 6 and 8 are not relatively prime
With a=6 and n=8,

Z8	0 1 2 3 4 5 6 7
Multiply by 6:	0 6 12 18 24 30 36 42
Residues:	0 6 4 2 0 6 4 2

However, if we take a=5 and n=8, whose only common factor is 1,

Z8	0 1 2 3 4 5 6 7
Multiply by 5:	0 5 10 15 20 25 30 35
Residues:	0 5 2 7 4 1 6 3

The line of residues contains all integers in Z8, in a different order.
In general, an integer has a multiplicative inverse in Zn, if that integer is relatively prime to n. Table 3.1c shows that the integers 1, 3, 5, and 7 have a multiplicative inverse, but 2, 4, and 6 do not.

4.2 Euclid's algorithm

One of the basic techniques of number theory is Euclid's algorithm, which is a simple procedure for determining the greatest common divisor of two positive numbers.

4.2.1 Greatest common divisor

We will use notation gcd(a,b) to mean the greatest common divisor of a and b. The positive integer c is said tob the greatest common divisor of a and b if

1. c is a divisor of a and of b
2. any divisor of a and b is a divisor of c

An equivalent definition is the following:

gcd(a,b)=max[k, such that k|a and k|b]

Because we require that the greatest common divisor be positive, gcd(a,b)=gcd(a,-b)=gcd(-a,b)=gcd(-a,-b). In general, gcd(a,b)=gcd(|a|,|b|).

gcd(60,24)=gcd(60,-24)=12

Also, because all nonzero integers divide 0, we have gcd(a,0)=|a|.

We stated that two integers are relatively prime if their only common positive integer factor is 1. This is equivalent to saying that a and b are relatively prime if gcd(a,b)=1.

8 and 15 are relatively prime because the positive divisors of 8 are 1,2,4, and 8, and the positive divisors of 15 are 1,3,5, and 15, so 1 is the only integer on both lists.

4.2.2 Finding the greatest common divisor

Euclid's algorithm is based on the following theorem: For any nonnegative integer a and any positive integer b,gcd(a, b)=gcd(b, a mod b)

gcd(55,22)=gcd(22, 55 mod 22) = gcd(22,11) = 11

To see, that works, let d=gcd(a,b). Then, by the definition of gcd, d|a and d|b. For any positive integer b, a can be expressed in the form

a = kb+r ≡r mod b

a mod b = r

with k, r integers.

Therefore, (a mod b) = a-kb for some integer k.

But because d|b, it also divides kb.

We also have d|a. Therefore, d|(a mod b).

This shows, that d is a common divisor of b and (a mod b).

Conversely, if d is a common divisor of b and (a mod b), then d|kb and thus d|[kb+(a mod b)], which is equivalent to d|a.

Thus, the set of common divisors of a and b is equal to the set of common divisors of b and (a mod b). Therefore, the gcd of one pair is the same as the gcd of the other pair, proving the theorem.

Equation (4.3) can be used repetitively to determine the greatest common divisor:

gcd(18,12)=gcd(12,6)=gcd(6,0)=6

gcd(11,10)=gcd(10,1)=gcd(1,0)=1

Euclid's algorithm makes repeated use of (4.3) to determine the greatest common divisor, as follows. The algorithm assumes a>b>0. It is acceptable to restrict the algorithm to positive integers because gcd(a,b) = gcd(|a|,|b|)

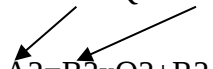
4.2.3 Euclid's algorithm

EUCLID(a,b)

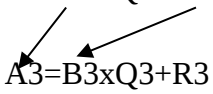
1. A:=a; B:=b
2. if B=0 return A=gcd(a,b)
3. R=A mod B
4. A:=B
5. B:=R
6. goto 2

The algorithm has the following progression:

A1=B1xQ1+R1



A2=B2xQ2+R2



A3=B3xQ3+R3

To find gcd(1970,1066)

1970=1x1066+904 *gcd(1066,904)*
1066=1x904+162 *gcd(904,162)*
904=5x162+94 *gcd(162,94)*
162=1x94+68 *gcd(94,68)*
94=1x68+26 *gcd(68,26)*
68=2x26+16 *gcd(26,16)*
26=1x16+10 *gcd(16,10)*
16=1x10+6 *gcd(10,6)*
10=1x6+4 *gcd(6,4)*
6=1x4+2 *gcd(4,2)*
4=2x2+0 *gcd(2,0)*

Therefore, gcd(1970,1066)=2

This process should terminate, otherwise we would get an endless sequence of positive integers, each one is strictly smaller than the one before, and this is clearly impossible.

Lecture 5

5.1 FINITE FIELD

Z_n^* is usually used to denote (Coprime-n, *), i.e. the multiplicative group of integers modulo n.

Now we are ready for finite fields. A field is a non-empty set F with **two** binary operators which are usually denoted by + and *, that satisfy the usual arithmetic properties:

- $(F, +)$ is an Abelian group with (additive) identity denoted by 0.
- $(F\backslash\{0\}, \cdot)$ is an Abelian group with (multiplicative) identity denoted by 1.
- The distributive law holds: (a+b)*c = a*c+b*c for all a, b, c ∈ F.

If the set F is finite, then the field is said to be a **finite field**. The **order** of a finite field is the number of elements in the finite field. By definition, $(Z, +, *)$ does not form a field because $(Z\backslash\{0\}, *)$ is not a multiplicative group. $(Z_n, +, *)$ in general is not a finite field. For example, $Z_8/\{0\} = \{1, 2, 3, 4, 5, 6, 7\}$ along with modulo 8 multiplication does not form a group. However, when n is a prime number, things become different. For example $Z_5/\{0\} = \{1, 2, 3, 4\}$ along with modulo 5 multiplication forms the Abelian group Z_5^*. Therefore, $(Z_5, +, *)$ is a finite field. There is a (not so funny) limerick may help you memorise this fact:

In arctic and tropical climes,

The integers, addition, and times,

Taken (mod p) will yield

A full finite field,

As p ranges over the primes.

5.2 Polynomial Arithmetic

For the very simple reason that we can represent a bit pattern by a polynomial in, say, the variable x. Each power of x in the polynomial can stand for a bit position in a bit pattern. For example, we can represent the bit pattern 111 by the polynomial $x^2 + x + 1$. On the other hand, the bit pattern 101 would be represented by the polynomial $x^2 + 1$ and the pattern 011 by the polynomial $x + 1$. Since all the bits in 111 are set, we have all powers of x up to 2 in the corresponding polynomial. On the other hand, in the bit pattern 101, the second bit is not set, so the coefficient of x in the polynomial is zero, which results in the polynomial $x^2 + 1$. Finally, in the bit pattern 011, since only the two least significant bits are set, the polynomial becomes $x + 1$. As you will see in this lecture, this ploy of representing a bit pattern with a polynomial will allow us to create a finite field with bit patterns. • In general, a polynomial is an expression of the form

$$a_n x^n + a_{n-1} x^{n-1} + \ldots\ldots + a_1 x + a_0$$

for some non-negative integer n and where the coefficients a0, a1, ..., an are drawn from some designated set S. S is called the coefficient set. • When an 6= 0, we have a polynomial of degree n.A zeroth-degree polynomial is called a constant polynomial. Polynomial arithmetic deals with the addition, subtraction, multiplication, and division of polynomials. Note that we have no interest in evaluating the value of a polynomial for any value of the variable x.

5.3 Prime Numbers

- prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
  - eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- prime numbers are central to number theory
- list of prime number less than 200 is:
  2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

5.4 Prime Factorisation

- to **factor** a number n is to write it as a product of other numbers: n=a x b x c
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number n is when its written as a product of primes
  eg. $91=7x13$; $3600=2^4x3^2x5^2$

5.5 Fermat's Theorem

Pierre de Fermat (1601-1665) was a lawyer by profession and an amateur mathematician. Fermat rarely published his mathematical discoveries. It was mostly through his correspondence with other mathematicians that his work is known at all. Fermat was one the inventors of analytic geometry and came up with some of the fundamental ideas of calculus. He is probably most famous for a problem that went unsolved until 1994; that the equation $x^n + y^n = z^n$ has no non-trivial solution when n>2. One of Fermat's books contained a handwritten note in the margin declaring that he had a proof for this equation, but it would not fit in the margin. He never published his proof, nor was it found after his death. In 1994 Andrew Wiles worked out a proof of this equation using advanced modern techniques.

5.5.1 Theorem:

If p is prime and a is an integer not divisible by p, then . . .
$a^{p-1} \equiv 1 \pmod{p}$.
And for every integer a
$a^p \equiv a \pmod{p}$.
This theorem is useful in public key (RSA) and primality testing.

5.2 Euler Totient Function:(n)

$\phi(n)$ = how many numbers there are between 1 and n-1 that are relatively prime to n.
$\phi(4) = 2$ (1, 3 are relatively prime to 4)
$\phi(5) = 4$ (1, 2, 3, 4 are relatively prime to 5)

(6) = 2 (1, 5 are relatively prime to 6)
(7) = 6 (1, 2, 3, 4, 5, 6 are relatively prime to 7)

As you can see from (5) and (7), (*n*) will be *n*-1 whenever *n* is a prime number. This implies that (*n*) will be easy to calculate when *n* has exactly two different prime factors: (*P* * *Q*) = (*P*-1)* (*Q*-1), if *P* and *Q* are prime.

5.2.1 Euler's Totient Theorem:

This theorem generalizes Fermat's theorem and is an important key to the RSA algorithm.

If GCD(*a*, *p*) = 1, and *a* < *p*, then $a^{(p)} \equiv 1 \pmod{p}$.

In other words, If *a* and *p* are relatively prime, with *a* being the smaller integer, then when we multiply *a* with itself (*p*) times and divide the result by *p*, the remainder will be 1.

Example:

Let's test the theorem:

If *a* = 5 and *p* = 6

Then (6) = (2-1) * (3-1) = 2

So, $5^{(6)} = 25$ and 25 = 24+1 = 6*4+1

=> 25 = 1(mod 6) OR 25 % 6 = 1

It also follows that $a^{(p)+1} \equiv a \pmod{p}$ so that *p* does not necessarily need to be relatively prime to *a*.

Euler's theorem uses modulus arithmetic which helps to lay the foundation for RSA encryption. To construct a personal cipher key we need an appropriate value we will call variable *R*. So, we select two very large prime numbers *U* and *V* and multiply them.

=> (*R*) = (*U*-1)*(*V*-1). This makes *R* difficult to factor, since the fewer factors a number has, the longer it takes to find them.

Euler's Theorem

- a generalisation of Fermat's Theorem
- $a^{ø(n)} = 1 \pmod{n}$
  - for any a,n where gcd(a,n)=1
- eg.

  a=3;*n*=10; ø(10)=4;
  hence $3^4 = 81 = 1 \bmod 10$
  a=2;*n*=11; ø(11)=10;
  hence $2^{10} = 1024 = 1 \bmod 11$

Lecture 6

6.1 Chinese Remainder Theorem

Let $n_1, n_2, \ldots, n_r$ be positive integers such that $\gcd(n_i, n_j) = 1$ for $i \neq j$. Then the system of linear congruences

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_r \pmod{n_r} \end{aligned}$$

has a simultaneous solution, which is unique modulo the integer $n_1 n_2 \cdots n_r$.

Proof:

We start by forming the product $n = n_1 n_2 \cdots n_r$. For each $k = 1, 2, \ldots r,$ let

$$N_k = \frac{n}{n_k} = n_1 \cdots n_{k-1} n_{k+1} \cdots n_r .$$

That is, N_k is the product of all the integers n_i with the factor n_k omitted. By hypothesis, the n_i are relatively prime in pairs, so that $\gcd(N_k, n_k) = 1$. According to the theory of a single linear congruence, it is therefore possible to solve the congruence $N_k x \equiv 1 \pmod{n_k}$; call the unique solution x_k.

Claim: The integer $\bar{x} = a_1 N_1 x_1 + a_2 N_2 x_2 + \cdots + a_r N_r x_r$ … (1)
is a simultaneous solution of the given system.

First, observe that $N_i \equiv 0 \pmod{n_k}$ for $i \neq k,$ because $n_k \mid N_i$ in this case. The result is

$$\bar{x} = a_1 N_1 x_1 + \cdots + a_r N_r x_r \equiv a_k N_k x_k \pmod{n_k}$$

But the integer x_k was chosen to satisfy the congruence $N_k x \equiv 1 \pmod{n_k},$ which forces

$$\bar{x} \equiv a_k \cdot 1 \equiv a_k \pmod{n_k} .$$

That is, the integer $\bar{x} = a_1 N_1 x_1 + a_2 N_2 x_2 + \cdots + a_r N_r x_r$ is a simultaneous solution of the given system. This shows that a solution to the given system of congruences exists.

As for the uniqueness assertion, suppose that x' is any other integer that satisfies these congruences. Then $\bar{x} \equiv a_k \equiv x' \pmod{n_k}$ $k = 1, 2, \ldots, r$ and so $n_k \mid \bar{x} - x'$ for each value of *k*.

That is, $n_1 \mid \bar{x} - x', \quad n_2 \mid \bar{x} - x', \ \ldots \ , \ n_r \mid \bar{x} - x',$ with $\gcd(n_i, n_j) = 1$ for $i, j = 1, 2, \ldots, r$ with $i \neq j$.

This, with the application of mathematical induction on the result "If $a \mid c$ and $b \mid c$ with $\gcd(a, b) = 1,$ then $ab \mid c$", gives $n_1 n_2 \cdots n_r \mid \bar{x} - x'.$

Hence, $\bar{x} \equiv x' \pmod{n}$. This completes the proof of the Chinese Remainder Theorem.

Examples:

***Example* 1** Find the least positive integer x such that , $x \equiv 7 \pmod{11}$, and $x \equiv 3 \pmod{13}$.

Solution

In the notation of Chinese Remainder Theorem, we have $n = 7 \cdot 11 \cdot 13 = 1001$ and

$$N_1 = \frac{n}{7} = 143 \qquad N_2 = \frac{n}{11} = 91 \qquad N_3 = \frac{n}{13} = 77.$$

Now the linear congruences

$$143x \equiv 1 (\mathrm{mod}\, 7)$$
$$91x \equiv 1 (\mathrm{mod}\, 11)$$
$$77x \equiv 1 (\mathrm{mod}\, 13)$$

are satisfied by $x_1 = 5,\ x_2 = 4$, $x_3 = 12$, respectively. Thus, a solution of the system is given by

$$x = 5 \cdot 143 \cdot 5 + 7 \cdot 91 \cdot 4 + 3 \cdot 77 \cdot 12 = 8895.$$

Thus 887 is the least positive solution.

***Example* 2** The problem posed by Sun-Tsu is the following: Find a number that leaves the remainders 2, 3, and 2 when divided by 3,5, and 7, respectively.

Solution

The given problem corresponds to the system of three congruences

$$x \equiv 2 (\mathrm{mod}\, 3)$$
$$x \equiv 3 (\mathrm{mod}\, 5)$$
$$x \equiv 2 (\mathrm{mod}\, 7)$$

In the notation of Chinese Remainder Theorem, we have $n = 3 \cdot 5 \cdot 7 = 105$ and

$$N_1 = \frac{n}{3} = 35 \qquad N_2 = \frac{n}{5} = 21 \qquad N_3 = \frac{n}{7} = 15.$$

Now the linear congruences $35x \equiv 1 (\mathrm{mod}\, 3)$ $21x \equiv 1 (\mathrm{mod}\, 5)$ $15x \equiv 1 (\mathrm{mod}\, 7)$

are satisfied by $x_1 = 2,\ x_2 = 1$, $x_3 = 1$, respectively. Thus, a solution of the system is given by

$$x = 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 = 233.$$

Modulo 105, we get the unique solution $x = 233 \equiv 23 (\mathrm{mod}\, 105)$.

***Example* 3** Show that there is no *x* for which both $x \equiv 29 (\mathrm{mod}\, 52)$ and $x \equiv 19 (\mathrm{mod}\, 72)$.

Solution

Since $52 = 4 \cdot 13$, the first congruence is equivalent to the simultaneous congruences

$$x \equiv 29 (\mathrm{mod}\, 4)$$

and $$x \equiv 29 (\mathrm{mod}\, 13),$$

which reduces to

$$x \equiv 1 (\mathrm{mod}\, 4)$$

and $$x \equiv 3 (\mathrm{mod}\, 13).$$

Similarly, $72 = 8 \cdot 9$ and the second congruence given in equivalent to the simultaneous congruences

$$x \equiv 19 (\mathrm{mod}\, 8)$$

and $$x \equiv 19 (\mathrm{mod}\, 9).$$

These reduce to

$$x \equiv 3 (\mathrm{mod}\, 8)$$

and $$x \equiv 1 (\mathrm{mod}\, 9).$$

The given congruences are inconsistent because there is no x for which both $x \equiv 1 (\mathrm{mod}\, 4)$ and $x \equiv 3 (\mathrm{mod}\, 8)$. Hence there is no x for which both $x \equiv 29 (\mathrm{mod}\, 52)$ and $x \equiv 19 (\mathrm{mod}\, 72)$.

***Example* 4** Determine whether the system $x \equiv 3 (\mathrm{mod}\, 10)$, $x \equiv 8 (\mathrm{mod}\, 15)$, $x \equiv 5 (\mathrm{mod}\, 84)$ has a solution, and find them all, if any exist.

Solution

We factor each modulus into prime powers. Since $10 = 2 \cdot 5$, the first congruence of the system is equivalent to the two simultaneous congruences

$$x \equiv 3 (\mathrm{mod}\, 2),$$

and

$$x \equiv 3 (\mathrm{mod}\, 5).$$

Similarly, the second congruence of the system is equivalent to the two conditions

$$x \equiv 8 (\mathrm{mod}\, 3),$$

and

$$x \equiv 8 (\mathrm{mod}\, 5),$$

while the third congruence is equivalent to the three congruences

$$x \equiv 5 (\mathrm{mod}\, 4),$$

$$x \equiv 5 (\mathrm{mod}\, 3),$$

and

$$x \equiv 5 (\mathrm{mod}\, 7).$$

The new system of seven simultaneous congruences is equivalent to the ones given, but now all moduli are prime powers. We consider the powers of 2 first. The two conditions are

$$x \equiv 3 (\mathrm{mod}\, 2)$$

and

$$x \equiv 1 (\mathrm{mod}\, 4).$$

These two are consistent, but the second one implies the first, so that the first one may be dropped. The conditions modulo 3 are

$$x \equiv 8 (\mathrm{mod}\, 3)$$

and

$$x \equiv 5 (\mathrm{mod}\, 3).$$

These are equivalent, and may be expressed as

$$x \equiv 2 (\mathrm{mod}\, 3).$$

Third, the conditions modulo 5 are

$$x \equiv 3 (\mathrm{mod}\, 5)$$

and

$$x \equiv 8 (\mathrm{mod}\, 5).$$

These are equivalent, so we drop the second of them. Finally, we have the condition

$$x \equiv 5 (\mathrm{mod}\, 7).$$

Hence our system of seven congruences is equivalent to the four conditions

$$x \equiv 1 (\mathrm{mod}\, 4),$$

$$x \equiv 2 (\mathrm{mod}\, 3),$$

$$x \equiv 3 (\mathrm{mod}\, 5),$$

and

$$x \equiv 5 (\mathrm{mod}\, 7).$$

Hence the moduli are relatively prime in pairs, so we may apply the formula used in the proof of the Chinese Remainder Theorem. Proceeding as in the solution of Example 3, we find that x satisfies the given congruences if and only if $x \equiv 173 (\mathrm{mod}\, 420)$.

***Example* 5** Solve the linear congruence

$$17x \equiv 9 (\mathrm{mod}\ 276).$$

Solution

Because $276 = 3 \cdot 4 \cdot 23,$ this is equivalent to finding a solution for the system of congruences

$$17x \equiv 9 (\mathrm{mod}\ 3)$$
$$17x \equiv 9 (\mathrm{mod}\ 4)$$
$$17x \equiv 9 (\mathrm{mod}\ 23)$$

or

$$x \equiv 0 (\mathrm{mod}\ 3)$$
$$x \equiv 1 (\mathrm{mod}\ 4)$$
$$17x \equiv 9 (\mathrm{mod}\ 23)$$

Note that if $x \equiv 0 (\mathrm{mod}\ 3),$ then $x = 3k$ for any integer *k*. We substitute into the second congruence of the system and obtain

$$3k \equiv 1 (\mathrm{mod}\ 4).$$

Multiplication of both sides of this congruence by 3 gives us

$$k \equiv 9k \equiv 3 (\mathrm{mod}\ 4).$$

Hence $k = 3 + 4j,$ where *j* is an integer. Then

$$x = 3(3 + 4j) = 9 + 12j$$

For x to satisfy the last congruence, we must have

$$17(9 + 12j) \equiv 9 (\mathrm{mod}\ 23)$$

or $204j \equiv -144 (\mathrm{mod}\ 23),$ which reduces to $3j \equiv 6 (\mathrm{mod}\ 23).$ Hence, $j = 2 (\mathrm{mod}\ 23).$ This yields $j \equiv 2 + 23t,$ with *t* an integer, and hence

$$x = 9 + 12(2 + 23t) = 33 + 276t.$$

Module -2: [8L]

BLOCK CIPHERS AND PUBLIC KEY CRYPTOGRAPHY

Lecture 7

7.1 BLOCK CIPHER PRINCIPLES

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Fiestel block cipher. For that reason, it is important to examine the design principles of the Fiestel cipher. We begin with a **comparison of stream cipher with block cipher.**

- **A stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. E.g, vigenere cipher. **A block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically a block size of 64 or 128 bits is used.

7.1.1 Block cipher principles

- Most symmetric block ciphers are based on a **Feistel Cipher Structure** needed since must be able to **decrypt** ciphertext to recover messages efficiently. block ciphers look like an extremely large substitution
- would need table of 264 entries for a 64-bit block
- Instead create from smaller building blocks
- using idea of a product cipher in 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks called modern substitution-transposition product cipher these form the basis of modern block ciphers
- S-P networks are based on the two primitive cryptographic operations we have seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
- provide *confusion* and *diffusion* of message
- **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
- **confusion** – makes relationship between ciphertext and key as complex as possible

7.2 Modes of operation

7.2.1 Block Modes

Splits messages in blocks (ECB, CBC)

7.2.1.1Electronic Codebook Book (ECB)

- Where the message is broken into independent 64-bit blocks which are encrypted
  C_(i) = DES_(K1) (P_(i))

7.2.1.2Cipher Block Chaining (CBC)

Again the message is broken into 64-bit blocks, but they are linked together in the encryption operation with an Initialization Vector (IV)
C_(i) = DES_(K1) (P_(i)(+)C_(i-1)) C_(-1)=IV

7.2.2Stream Modes

On bit stream messages (CFB, OFB)

7.2.2.1Cipher Feedback (CFB)

- Where the message is treated as a stream of bits, added to the output of the DES, with the result being feedback for the next stage
  C_(i) = P_(i)(+) DES_(K1) (C_(i-1)) C_(-1)=IV

7.2.2.2 Output Feedback (OFB)

- Where the message is treated as a stream of bits, added to the message, but with the feedback being independent of the message
  C_(i) = P_(i)(+) O_(i) O_(i) = DES_(K1)(O_(i-1)) O_(-1)=IV

7.2.3 Limitations of Various Modes

1. ECB

Repetitions in message can be reflected in ciphertext

- if aligned with message block
- particularly with data such graphics
- or with messages that change very little, which become a code-book analysis problem

Weakness is because enciphered message blocks are independent of each other

	Message (plaintext)	Ciphertext
1	T H E T C A N	60 99 46 42 52 82 22 49
2	H A V E S E	FF BF BC 77 8B BB F2 06
3	V E R A L A C	0D 4D 86 DE B6 CD 92 5D
4	T I V E P E R	99 63 A8 OF 32 D3 E7 E9
5	M A N E N T V	10 49 1F 3B DE 67 21 B7
6	I R T U A L C	BD 2D 6D 61 42 06 C7 B8
7	D A T A S	19 F1 01 A4 89 6A AE 4C
8	A G D / C A Y	84 DB CC EE 32 1B 5B 9C
9	I R T U A L C	BD 2D 6D 61 42 0B C7 B8
10	A L L S A T	D4 3C D4 5A 9E 0B A5 ED
11	T H E S A M E	84 52 01 AC 2D FE 98 3A
12	T I M E .	89 F1 89 E9 DB CC CB BB
	Key	01 23 45 67 89 AB CD EF

Fig. 4.1 A weakness in ECB encipherment

2. CBC

- Use result of one encryption to modify input of next. Hence, each ciphertext block is dependent on **all** message blocks before it. Thus, a change in the message affects the ciphertext block after the change as well as the original block

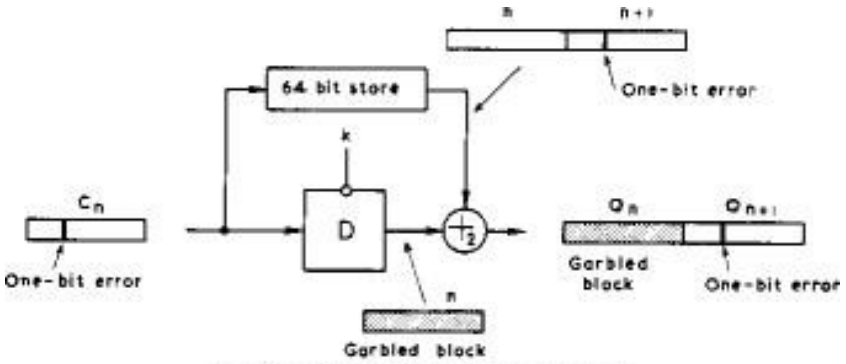


Fig. 4.5 One-bit error in cipher block chaining

- Moreover, to start an **Initial Value** (IV) is needed which must be known by both sender and receiver. However, if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate. Hence, either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message
- Also, at the end of the message, have to handle a possible last short block, either pad last block (possible with count of pad size), or use some fiddling to double up last two blocks

3. CFB

- When data is bit or byte oriented, want to operate on it at that level, so use a stream mode
- The block cipher is use in **encryption** mode at **both** ends, with input being a feed-back copy of the ciphertext
- Can vary the number of bits feedback, trading off efficiency for ease of use
- Again errors propogate for several blocks after the error

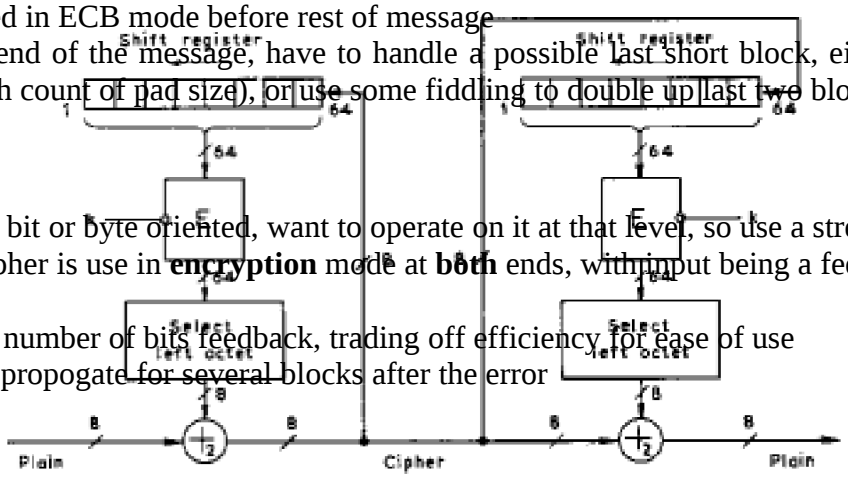


Fig. 4.7 8-bit cipher feedback

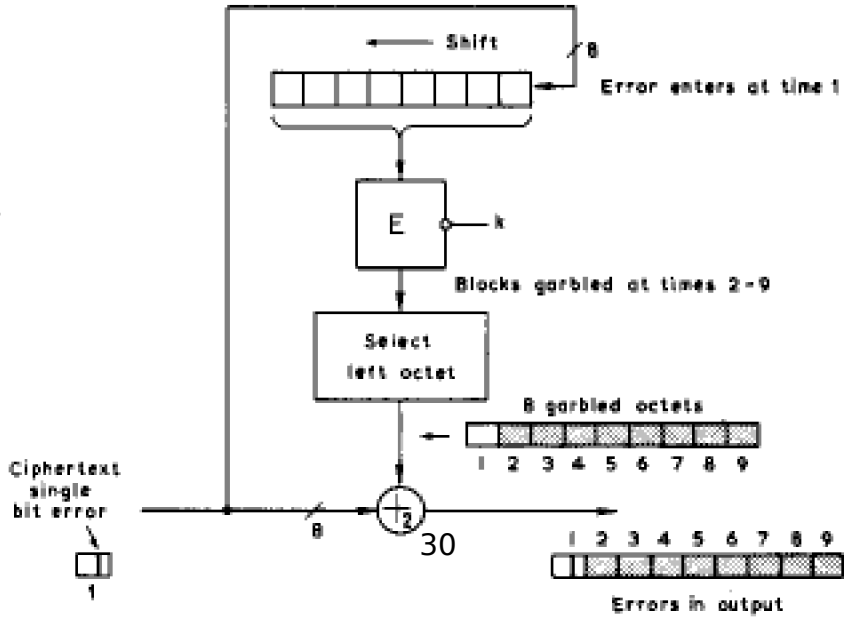


Fig. 4.8 One-bit error in 8-bit cipher feedback

4. OFB

- Also a stream mode, but intended for use where the error feedback is a problem, or where the encryptions want to be done before the message is available
- Is superficially similar to CFB, but the feedback is from the output of the block cipher and is independent of the message, a variation of a Vernam cipher
- Again an IV is needed
- Sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
- Although originally specified with varying m-bit feedback in the standards, subsequent research has shown that only **64-bit OFB** should ever be used (and this is the most efficient use anyway)

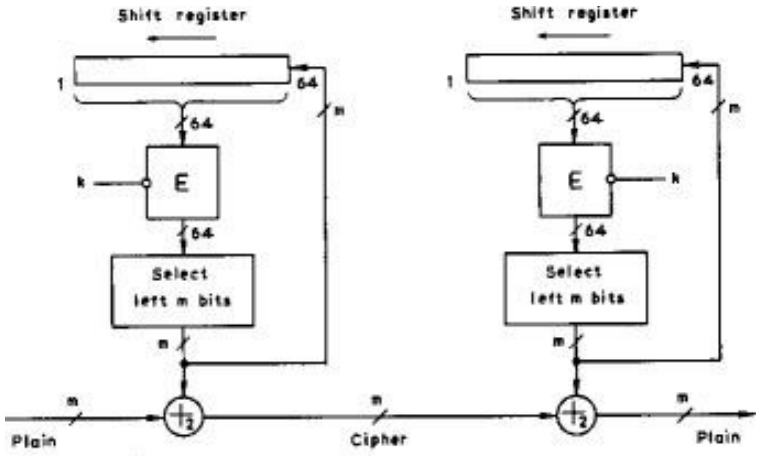


Fig. 4.12 m-bit output feedback

Lecture 8

8.1 DATA ENCRYPTION STANDARD (DES)

In May 1973, and again in Aug 1974 the NBS (now NIST) called for possible encryption algorithms for use in unclassified government applications response was mostly disappointing, however IBM submitted their Lucifer design following a period of redesign and comment it became the Data Encryption Standard (DES).

It was adopted as a (US) federal standard in Nov 76, published by NBS as a hardware only scheme in Jan 77 and by ANSI for both hardware and software standards in ANSI X3.92-1981 (also X3.106-1983 modes of use) subsequently it has been widely adopted and is now published in many standards around the world of Australian Standard AS2805.5-1985.

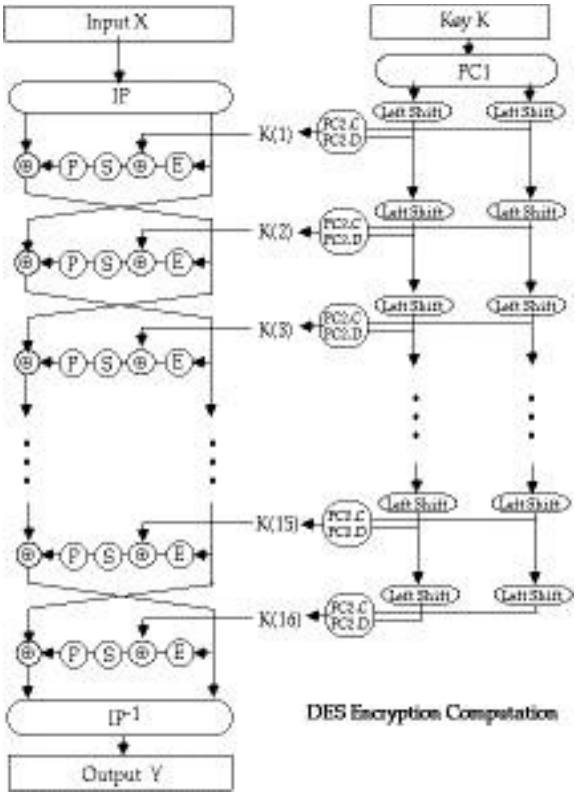
One of the largest users of the DES is the banking industry, particularly with EFT, and EFTPOS.

It is for this use that the DES has primarily been standardized, with ANSI having twice reconfirmed its recommended use for 5 year periods - a further extension is not expected however although the standard is public, the design criteria used are classified and have yet to be released there has been considerable controversy over the design, particularly in the choice of a 56-bit key

- recent analysis has shown despite this that the choice was appropriate, and that DES is well designed
- rapid advances in computing speed though have rendered the 56 bit key susceptible to exhaustive key search, as predicted by Diffie & Hellman

- The DES has also been theoretically broken using a method called Differential Cryptanalysis, however in practice this is unlikely to be a problem (yet).

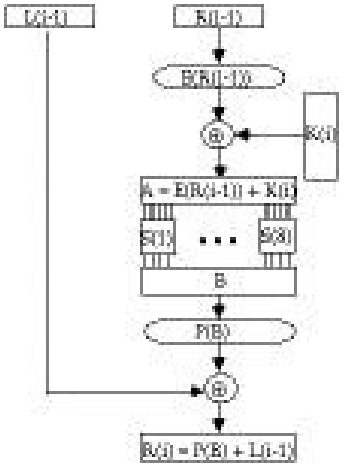
8.1.1 Overview of the DES Encryption Algorithm



The basic process in enciphering a 64-bit data block using the DES consists of:

- an initial permutation (IP)
- 16 rounds of a complex key dependent calculation f
- a final permutation, being the inverse of IP

in more detail the 16 rounds of f consist of:



This can be described functionally as
L(i) = R(i-1)

R(i) = L(i-1) (+) P(S(E(R(i-1))(+) K(i)))

and forms one round in an S-P network.

The subkeys used by the 16 rounds are formed by the **key schedule** which consists of:

- an initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
- 16 stages consisting of
- selecting 24-bits from each half and permuting them by PC2 for use in function f,
- rotating each half either 1 or 2 places depending on the **key rotation schedule** KS

This can be described functionally as:

K(i) = PC2(KS(PC1(K),i))

The **key rotation schedule** KS is specified as:

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
KS	1	1	2	2	2	2	2	2	1	2	2	2	2	2	1	272
Total Rot	1	2	4	6	8	10	12	14	15	17	19	21	23	25		

8.1.2 DES Modes of Use

- DES encrypts 64-bit blocks of data, using a 56-bit key
- we need some way of specifying how to use it in practise, given that we usually have an arbitrary amount of information to encrypt
- the way we use a block cipher is called its **Mode of Use** and four have been defined for the DES by ANSI in the standard: ANSI X3.106-1983 Modes of Use)
- Modes are either:
  - Block Modes
  - Stream Modes

8.1.3 DES Weak Keys

- With many block ciphers there are some keys that should be avoided, because of reduced cipher complexity
- These keys are such that the same sub-key is generated in more than one round, and they include:
  - **Weak Keys**: Same sub-key is generated for every round. DES has 4 weak keys
  - **Semi-Weak Keys**: Only two sub-keys are generated on alternate rounds. DES has 12 of these (in 6 pairs)

- **Demi-Semi Weak Keys**: Have four sub-keys generated

- None of these cause a problem since they are a tiny fraction of all available keys
- However they MUST be avoided by any key generation program

8.1.4 DES S-Box Design Criteria

Each S-box may be considered as four substitution functions
- These 1-1 functions map inputs 2,3,4,5 onto output bits
- A particular function is selected by bits 1,6
- This provides an autoclave feature

8.1.4 DES Design Criteria

There were 12 criterion used, resulting in about 1000 possible S-Boxes, of which the implementers chose 8. These criteria are CLASSIFIED SECRET. However, some of them have become known. The following are design criterion:

R1 : Each row of an S-box is a permutation of 0 to 15

R2 : No S-Box is a linear of affine function of the input

R3 : Changing one input bit to an S-box results in changing at least two output bits

R4 : S(x) and S(x+001100) must differ in at least 2 bits

The following are said to be caused by design criteria

R5 : S(x) [[pi]] S(x+11*ef* 00) for any choice of *e* and *f*

R6 : The S-boxes were chosen to minimize the difference between the number of 1's and 0's in any S-box output when any single input is held constant

R7 : The S-boxes chosen require significantly more minterms than a random choice would require Meyer Tables 3-17, 3-18

8.1.5 DES Permutation Tables

There are 5 Permutations used in DES:

IP and **IP^(-1) , P, E, PC1, PC2**

Their design criteria are CLASSIFIED SECRET. It has been noted that **IP** and **IP^(-1)** and **PC1** serve no cryptological function when DES is used in ECB or CBC modes, since searches may be done in the space generated after they have been applied. **E, P,** and **PC2** combined with the S-Boxes must supply the required dependence of the output bits on the input bits and key bits (**avalanche** and **completeness** effects).

8.1.6 Ciphertext Dependence on Input and Key

The role of **P, E,** and **PC2** is distribute the outputs of the S-boxes so that each output bit becomes a function of all the input bits in as few rounds as possible. Carl Meyer (in Meyer 1978, or Meyer & Matyas 1982) performed this analysis on the current DES design.

8.1.7 Ciphertext dependence on Plaintext

Define **G_(i,j)**, a 64*64 array which shows the dependence of output bits X(j) on input bits X(i). Examine **G_(0,j)** to determine how fast complete dependence is achieved to build **G_(0,1)** use the following:

L(i) = R(i-1)

R(i) = L(i-1) (+) f(K(i), R(i-1))
DES P reaches complete dependence after 5 rounds.

8.1.8 Ciphertext dependence on Key

Carl Meyer also performed this analysis.
Define F_(i,j) a 64*56 array which shows the dependence of output bits X(j) on key bits U(i) (after PC1 is used). Examine F_(0,j) to determine how fast complete dependence is achieved. DES PC2 reaches complete dependence after 5 round.

8.1.9 Key Scheduling and PC2

Key Schedule is a critical component in the design must provide different keys for each round otherwise security may be compromised. Current scheme can result in weak keys which give the same, 2 or 4 keys over the 16 rounds

8.1.9.1 Design

- Is performed in two 28-bit independent halves
- C-side provides keys to S-boxes 1 to 4
- D-side provides keys to S-boxes 5 to 8
- The rotations are used to present different bits of the key for selection on successive rounds
- PC-2 selects key-bits and distributes them over the S-box inputs

8.1.10 Possible Techniques for Improving DES

- Multiple enciphering with DES
- Extending DES to 128-bit data paths and 112-bit keys
- Extending the Key Expansion calculation

8.1.11 Triple DES

DES variant standardised in ANSI X9.17 & ISO 8732 and in PEM for key management proposed for general EFT standard by ANSI X9. It is backwards compatible with many DES schemes. Uses 2 or 3 keys.
C = DES_(K1) Bbc{(DES^(-1)_(K2)Bbc{(DES_(K1)(P)))

There are no known practical attacks. Brute force search is impossible. Man-in-the-middle attacks need 2^(56) PC pairs per key.

8.1.12 Substitution-Permutation Ciphers

- An S-P network is the modern form of a substitution-transposition product cipher
- S-P networks are based on the two primitive cryptographic operations we have seen before

8.1.12.1 Substitution Operation

- A binary word is replaced by some other binary word
- The whole substitution function forms the key
- If use n bit words, the key is 2^(n)!bits, grows rapidly

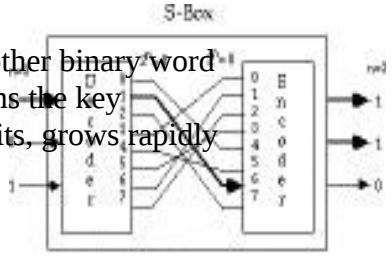


Fig 2.1 Substitution Operation

- Can also think of this as a large lookup table, with n address lines (hence 2^(n) addresses), each n bits wide being the output value
- **Known as S-boxes**

8.1.12.2 Permutation Operation

- A binary word has its bits reordered (permuted)
- The re-ordering forms the key
- If use n bit words, the key is n!bits, which grows more slowly, and hence is less secure than substitution

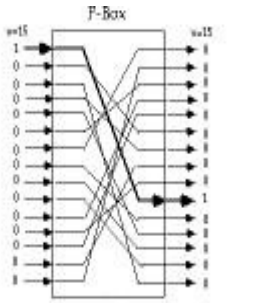


Fig 2.2 - Permutation or Transposition Function

- This is equivalent to a wire-crossing in practise (though is much harder to do in software)
- Known as **P-boxes**

8.1.12.3 Substitution-Permutation Network

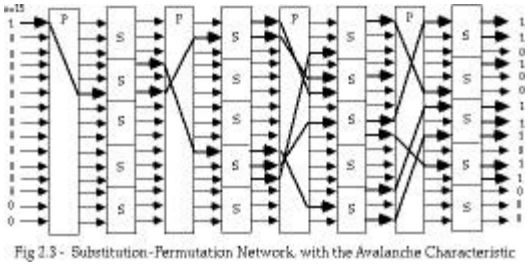


Fig 2.3 - Substitution-Permutation Network, with the Avalanche Characteristic

Shannons mixing transformations are a special form of product ciphers where

S-Boxes provide **confusion** of input bits

P-Boxes provide **diffusion** across S-box inputs

Avalanche effect: where changing **one** input bit results in changes of approx **half** the output bits

More formally, a function f has a good **avalanche** effect if for each bit i,0<=i<m, if the 2^(m) plaintext vectors are divided into 2^(m-1) pairs X and X_(i) with each pair differing only in bit i; and if the 2^(m-1) exclusive-or sums, termed avalanche vectors

V_(i) = f(X) (+) f(X_(i))

Are compared, then about half of these sums should be found to be 1.

Completeness effect: where each output bit is a complex function of **all** the input bits

More formally, a function f has a good **completeness** effect if for each bit j,0<=j<m, in the ciphertext output vector, there is at least one pair of plaintext vectors X and X_(i) which differ only in bit i, and for which f(X) and f(X_(i)) differ in bit j

8.1.12.3 Practical Substitution-Permutation Networks

In practice, we need to be able to decrypt messages, as well as to encrypt them, hence either:

- have to define inverses for each of our S & P-boxes, but this doubles the code/hardware needed, or
- define a structure that is easy to reverse, so can use basically the same code or hardware for both encryption and decryption

Horst Feistel, working at IBM Thomas J Watson Research Labs devised just such a structure in early 70's, which is called **feistel cipher.**

- the idea is to partition the input block into two halves, L(i-1)and R(i-1), and use only R(i-1)in each round i (part) of the cipher
- the function g incorporates one stage of the S-P network, controlled by part of the key K(i)known as the ith subkey

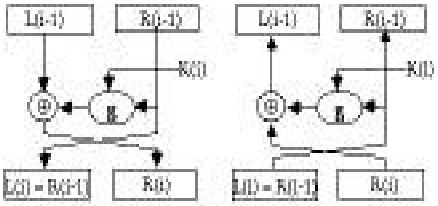


Fig 2.4 - A Round of a Feistel Cipher

this can be described functionally as:

L(i) = R(i-1)

R(i) = L(i-1) (+) g(K(i), R(i-1))

This can easily be reversed as seen in the above diagram, working backwards through the rounds.
In practice, link a number of these stages together (typically 16 rounds) to form the full cipher.

8.1.13 Differential Cryptanalysis of Block Ciphers

Differential Cryptanalysis is a recently (in the public research community) developed method which provides a powerful means of analyzing block ciphers. It has been used to analyze most of the currently proposed block ciphers with varying degrees of success. Usually have a break-even point in number of rounds of the cipher used for which differential cryptanalysis is faster than exhaustive key-space search. If this number is greater than that specified for the cipher, then it is regarded as broken.

8.1.14 Overview of Differential Cryptanalysis

It is a statistical attack against Feistel ciphers. Uses structure in cipher not previously used. Design of S-P networks is such that the output from function *f* is influenced by both input and key:
R(i)=L(i-1) (+) f(K(i)(+)R(i-1))

Hence, cannot trace values back through cipher without knowing the values of the key.

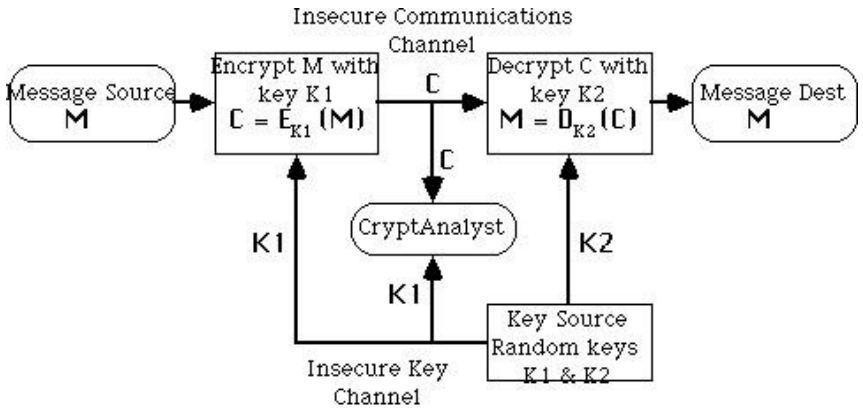
Lecture 9

9.1 Public-Key Ciphers

Traditional **secret key** cryptography uses a single key shared by both sender and receiver. If this key is disclosed communications are compromised. Also does not protect sender from receiver forging a message & claiming is sent by sender, parties are equal

Public-key (or **two-key**) **cryptography** involves the use of two keys:

- a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
- a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**



Asymmetric (Public-Key) Encryption System

The public-key is easily computed from the private key and other information about the cipher (a polynomial time (P-time) problem). However, knowing the public-key and public description of the cipher, it is still computationally infeasible to compute the private key (an NP-time problem). Thus, the public-key may be distributed to anyone wishing to communicate securely with its owner (although secure distribution of the public-key is a non-trivial problem - the **key distribution** problem)

The three important classes of public-key algorithms:

- **Public-Key Distribution Schemes** (PKDS) - where the scheme is used to securely exchange a single piece of information (whose value depends on the two parties, but cannot be set). This value is normally used as a session key for a private-key scheme
- **Signature Schemes** - used to create a digital signature only, where the private-key signs (create) signatures, and the public-key verifies signatures
- **Public Key Schemes (PKS)** - used for encryption, where the public-key encrypts messages, and the private-key decrypts messages.

Any public-key scheme can be used as a PKDS, just by selecting a message which is the required session key. Many public-key schemes are also signature schemes (provided encryption and decryption can be done

Lecture 11

11.1 RSA Public-Key Cryptosystem

It is a public-key scheme which may be used for encrypting messages, exchanging keys, and creating digital signatures. It is based on exponentiation in a finite (Galois) field over integers modulo a prime. Its security relies on the difficulty of calculating factors of large numbers.

RSA is a public key encryption algorithm based on exponentiation using modular arithmetic to use the scheme, first generate keys:

Key-Generation by each user consists of:

- selecting two large primes at random (~100 digit), p, q
- calculating the system modulus R=p.q p, q primes
- selecting at random the encryption key e
- e < R, gcd(e, F(R)) = 1
- solving the congruence to find the decryption key d
- e.d [[equivalence]] 1 mod [[phi]](R) 0 <= d <= R
- publishing the public encryption key: K1={e,R}
- securing the private decryption key: K2={d,p,q}
- Encryption of a message M to obtain ciphertext C is:

  $C = M^e \bmod R \quad 0 <= d <= R$
- Decryption of a ciphertext C to recover the message M is:

  M = Cd = Me.d = M1+n.[[phi]](R) = M mod R

The RSA system is based on the following result:

If R = pq where p, q are distinct large primes then

X [[phi]](R) = 1 mod R

for all x not divisible by p or q and [[Phi]](R) = (p-1)(q-1)

11.2 RSA Example

Usually the encryption key e is a small number, which must be relatively prime to [[phi]](R) (ie GCD(e, [[phi]](R)) = 1). Typically e may be the same for all users (provided certain precautions are taken), 3 is suggested. The decryption key d is found by solving the congruence:

e.d [[equivalence]] 1 mod [[phi]](R), 0 <= d <= R,

An extended Euclid's GCD or Binary GCD calculation is done to do this.

Given e=3, R=11*47=517, [[phi]](R)=10*46=460

then d=Inverse(3,460) by Euclid's alg:

i	y	g	u	v
0	-	460	1	0
1	-	3	0	1

2 153 1 1 -153
3 3 0 -3 460
ie: d = -153, or 307 mod 517

- a sample RSA encryption/decryption calculation is:

M = 26
C = 263 mod 517 = 515
M = 515307 mod 517 = 26

11.3 RSA and the Chinese Remainder Theorem

A significant improvement in decryption speed for RSA can be obtained by using the Chinese Remainder theorem to work modulo p and q, respectively. Since p, q are only half the size of R=p.q and thus the arithmetic is much faster.

CRT is used in RSA by creating two equations from the decryption calculation:

$$M = Cd \bmod R$$

$M1 = M \bmod p = (C \bmod p)d \bmod (p-1)$
$M2 = M \bmod q = (C \bmod q)d \bmod (q-1)$
then the pair of equations

$M = M1 \bmod p$ $M = M2 \bmod q$
has a unique solution by the CRT, given by:

$M = [((M2 + q - M1)u \bmod q] p + M1$
Where $p.u \bmod q = 1$

11.4 Primality Testing and RSA

The first stage of key-generation for RSA involves finding two large primes p, q. Because of the size of numbers used, must find primes by trial and error. Modern primality tests utilize properties of primes eg:

$a^{n-1} = 1 \bmod n$ where GCD(a,n)=1

all primes numbers 'n' will satisfy this equation

some composite numbers will also satisfy the equation, and are called pseudo-primes.

Most modern tests guess at a prime number 'n', then take a large number (eg 100) of numbers 'a', and apply this test to each. If it fails the number is composite, otherwise it is is probably prime.

There are a number of stronger tests which will accept fewer composites as prime than the above test. eg:

$$\text{GCD}(a,n) = 1, \quad \text{and} \quad \left(\frac{a}{n}\right) (\bmod n) = a^{\frac{(n-1)}{2}} (\bmod n)$$

$$\text{where } \left(\frac{a}{n}\right) \text{ is the Jacobi symbol}$$

Lecture 13

13.1 DIFFIE-HELLMAN KEY EXCHANGE

The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. First, we define a primitive root of a prime number p as one whose power generate all the integers from 1 to (p-1) i.e., if "a" is a primitive root of a prime number p, then the numbers a mod p, a^2 mod p, … a^{p-1} mod p are distinct and consists of integers from 1 to (p-1) in some permutation. For any integer "b" and a primitive root "a" of a prime number "p", we can find a unique exponent "i" such that

$b \equiv a^i \bmod p$ where $0 \leq i \leq (p-1)$

The exponent „i" is referred to as discrete logarithm. With this background, we can define Diffie Hellman key exchange as follows:

There are publicly known numbers: a prime number "q" and an integer **α** that is primitive root of q. suppose users A and B wish to exchange a key. User A selects a random integer XA < q and computes YA = $\boldsymbol{\alpha}^{XA} \bmod q$. Similarly, user B independently selects a random integer XB < q and computes YB = $\boldsymbol{\alpha}^{XB} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as

$K = (YB)^{XA} \bmod q$ and

User B computes the key as

$K = (YA)^{XB} \bmod q$

These two calculations produce identical results.

$K = (YB)^{XA} \bmod q$

$= (\boldsymbol{\alpha}^{XB} \bmod q)^{XA} \bmod q$

$= (\boldsymbol{\alpha}^{XB})^{XA} \bmod q$

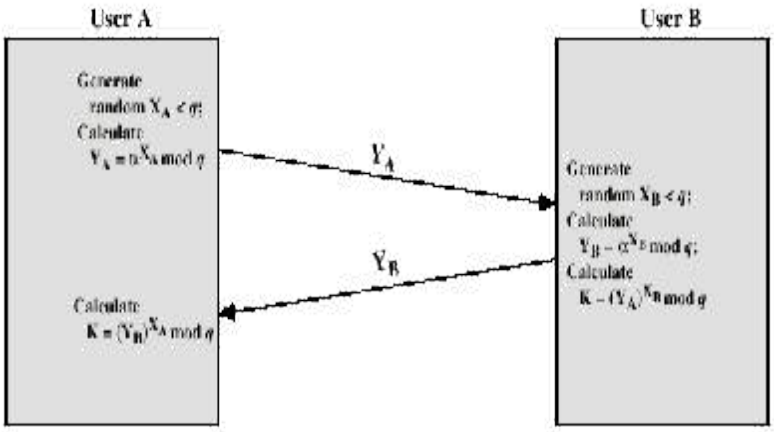
$= (\boldsymbol{\alpha}^{XA})^{XB} \bmod q$

$= (\boldsymbol{\alpha}^{XA} \bmod q)^{XB} \bmod q$

$= (YA)^{XB} \bmod q$

The result is that two sides have exchanged a secret key.

The security of the algorithm lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.


User A
User B
Generate
random $X_A < q$;
Calculate
$Y_A = \alpha^{X_A} \bmod q$
Y_A
Generate
random $X_B < q$;
Calculate
$Y_B = \alpha^{X_B} \bmod q$;
Calculate
$K = (Y_A)^{X_B} \bmod q$
Y_B
Calculate
$K = (Y_B)^{X_A} \bmod q$

Module 3: [6L]
HASH FUNCTIONS AND DIGITAL SIGNATURES

Lecture 15

15.1 AUTHENTICATION REQUIREMENTS

In the context of communication across a network, the following attacks can be identified:

- **Disclosure** – releases of message contents to any person or process not possessing the appropriate cryptographic key.
- **Traffic analysis** – discovery of the pattern of traffic between parties.
- **Masquerade** – insertion of messages into the network fraudulent source.
- **Content modification** – changes to the content of the message, including insertion deletion, transposition and modification.
- **Sequence modification** – any modification to a sequence of messages between parties, including insertion, deletion and reordering.
- **Timing modification** – delay or replay of messages.
- **Source repudiation** – denial of transmission of message by source.
- **Destination repudiation** – denial of transmission of message by destination.

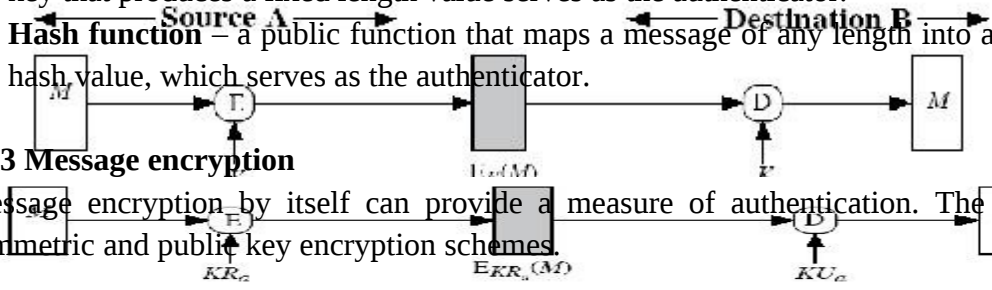
Measures to deal with first two attacks are in the realm of message confidentiality. Measures to deal with 3 through 6 are regarded as message authentication. Item 7 comes under digital signature and dealing with item 8 may require a combination of digital signature and a protocol to counter this attack.

15.2 AUTHENTICATION FUNCTIONS

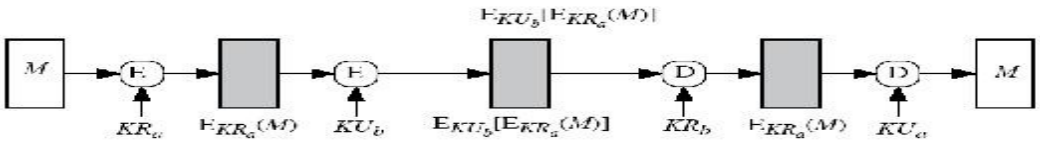
Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels. At the lower level, there may be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower layer function is then used as primitive in a higher-layer authentication protocol that enables a receiver to verify the authenticity of a message.

The different types of functions that may be used to produce an authenticator are as follows:

- **Message encryption** – the cipher text of the entire message serves as its authenticator.
- **Message authentication code (MAC)** – a public function of the message and a secret key that produces a fixed length value serves as the authenticator.
- **Hash function** – a public function that maps a message of any length into a fixed length hash value, which serves as the authenticator.



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

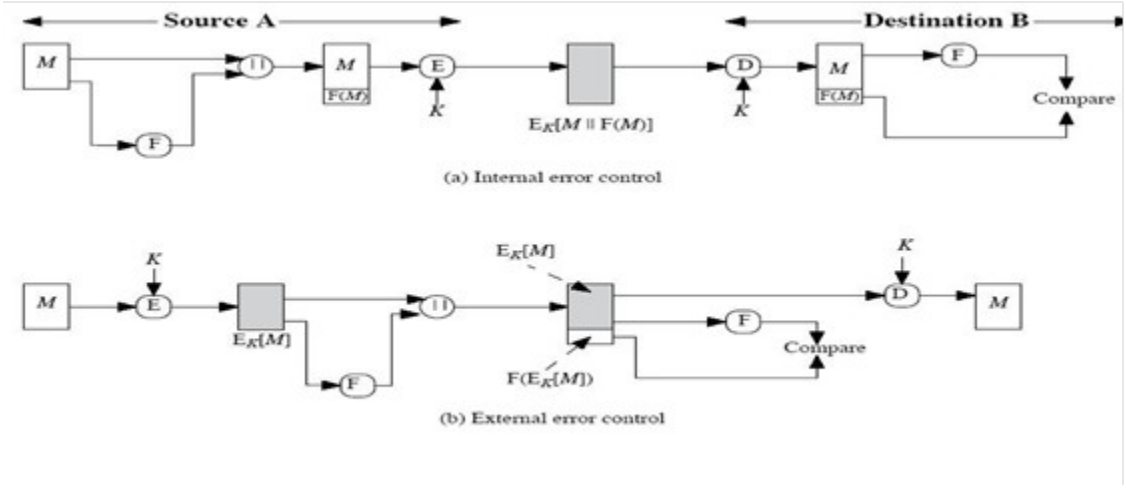
15.3 Message encryption

Message encryption by itself can provide a measure of authentication. The analysis differs from symmetric and public key encryption schemes.

Suppose the message can be any arbitrary bit pattern. In that case, there is no way to determine automatically, at the destination whether an incoming message is the ciphertext of a legitimate message. One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function. We could, for example, append an error detecting code, also known as Frame Check Sequence (FCS) or checksum to each message before encryption

'A' prepares a plaintext message M and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted. At the destination, B decrypts the incoming block and treats the result as a message with an appended FCS. B applies the same function F to attempt to reproduce the FCS. If the calculated FCS is equal to the incoming FCS, then the message is considered authentic.

In the internal error control, the function F is applied to the plaintext, whereas in external error control, F is applied to the ciphertext (encrypted message).



15.4 MESSAGE AUTHENTICATION CODE (MAC)

An alternative authentication technique involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that two communication parties say A and B, share a common secret key 'k'. When A has to send a message to B, it calculates the MAC as a function of the message and the key.

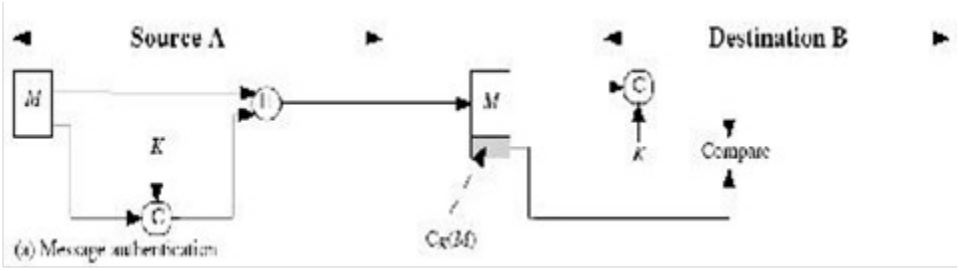
MAC = CK(M)

Where M – input message, C – MAC function, and K – Shared secret key, and MAC - Message Authentication Code

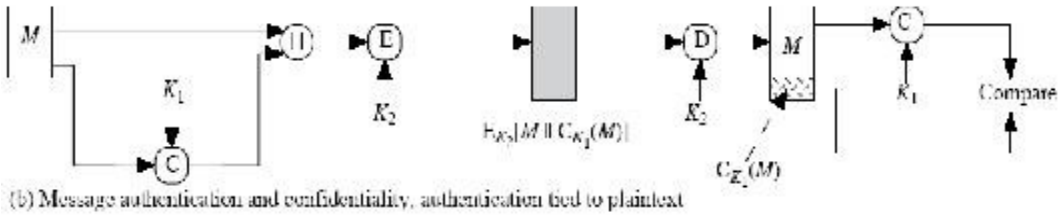
The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the shared secret key, to generate a new MAC. The

received MAC is compared to the calculated MAC. If it is equal, then the message is considered authentic.

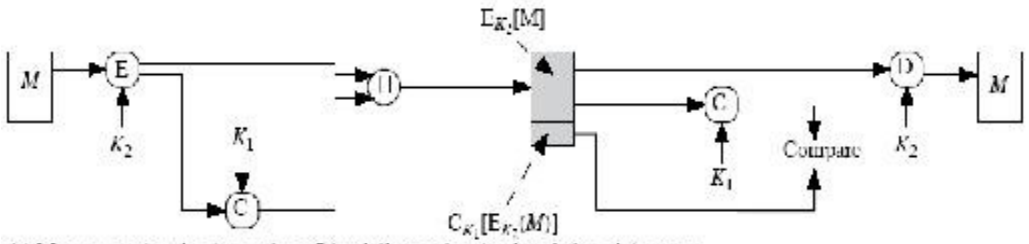
A MAC function is similar to encryption. One difference is that MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many- to-one function.



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

15.4.1 Requirements for MAC:

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute- force attack using all possible keys. On average, such an attack will require $2^{(k-1)}$ attempts for a k-bit key.

In the case of a MAC, the considerations are entirely different. Using brute-force methods, how would an opponent attempt to discover a key?

If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n; that is, suppose that the key size is greater than the MAC size. Then, given a known M1 and MAC1, with $MAC_1 = CK (M_1)$, the cryptanalyst can perform $MAC_i = CK_i (M1)$ for all possible key values K_i.

At least one key is guaranteed to produce a match of $MAC_i = MAC_1$.

Note that a total of 2^k MACs will be produced, but there are only $2^n < 2^k$ different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has no way of knowing which is the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:

Round 1
Given: M_1, $MAC_1 = CK(M_1)$
Compute $MAC_i = CK_i(M_1)$ for all 2^k keys
Number of matches $\approx 2^{(k-n)}$

Round 2
Given: M_2, $MAC_2 = CK(M_2)$
Compute $MAC_i = CK_i(M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1
Number of matches $\approx 2^{(k-2xn)}$

and so on. On average, a rounds will be needed if k = a x n. For example, if an 80-bit key is used and the MAC is 32 bits long, then the first round will produce about 2^{48} possible keys. The second round will narrow the possible keys to about 2^{16} possibilities. The third round should produce only a single key, which must be the one used by the sender.

If the key length is less than or equal to the MAC length, then it is likely that a first round will produce a single match.

Thus, a brute-force attempt to discover the authentication key is no less effort and may be more effort than that required to discover a decryption key of the same length. However, other attacks that do not require the discovery of the key are possible.

Consider the following MAC algorithm. Let $M = (X_1||X_2||...||X_m)$ be a message that is treated as a concatenation of 64-bit blocks X_i. Then define
$(M) = X_1 \oplus X_2 \oplus \ldots \oplus X_m$
$C_k(M) = E_k((M)$

where $\oplus$ is the exclusive-OR (XOR) operation and the encryption algorithm is DES in electronic codebook mode. Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes {M||C(K, M)}, a brute-force attempt to determine K will require at least 2^{56} encryptions. But the opponent can attack the system by replacing X_1 through X_{m-1} with any desired values Y_1 through Y_{m-1} and replacing X_m with Y_m where Y_m is calculated as follows:

$Y_m = Y_1 \oplus Y_2 \quad \oplus \quad \ldots \; Y_{m1} \oplus \quad (M)$

The opponent can now concatenate the new message, which consists of Y_1 through Y_m, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length 64 x (m-1) bits can be fraudulently inserted.

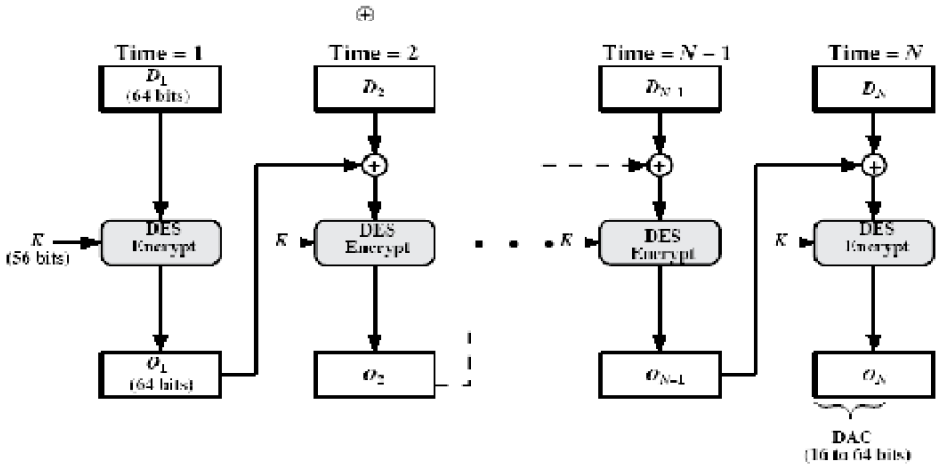
Then the MAC function should satisfy the following requirements: The MAC function should have the following properties:

If an opponent observes M and $C_K(M)$, it should be computationally infeasible for the opponent to construct a message M' such that $C_K(M') = C_K(M)$. $C_K(M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that $C_K(M) = C_K(M')$ is 2^{-n} where n is the number of bits in the MAC.

Let M' be equal to some known transformation on M. i.e., M' = f(M).

15.4.2 MAC based on DES

One of the most widely used MACs, referred to as Data Authentication Algorithm (DAA) is based on DES. The algorithm can be defined as using cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero. The data to be authenticated are grouped into contiguous 64-bit blocks: D_1, D_2, … D_n. if necessary, the final block is padded on the right with zeros to form a full 64-bit block. Using the DES encryption algorithm and a secret key, a data authentication code is calculated as shown in the diagram:



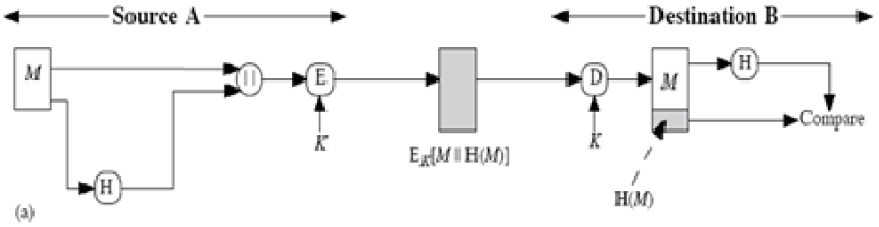
Lecture 16

16.1 HASH FUNCTIONS

A variation on the message authentication code is the one way hash function. As with MAC, a hash function accepts a variable size message M as input and produces a fixed-size output, referred to as hash code H(M). Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value.

There are varieties of ways in which a hash code can be used to provide message authentication, as follows:

a) The message plus the hash code is encrypted using symmetric encryption. This is identical to that of internal error control strategy. Because encryption is applied to the entire message plus the hash code, confidentiality is also provided.



b) Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c) Only the hash code is encrypted, using the public key encryption and using the sender's private key. It provides authentication plus the digital signature.

d) If confidentiality as well as digital signature is desired, then the message plus the public key encrypted hash code can be encrypted using a symmetric secret key.

e) This technique uses a hash function, but no encryption for message authentication. This technique assumes that the two communicating parties share a common secret value 'S'. The source computes the hash value over the concatenation of M and S and appends the resulting hash value to M.

f) Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.

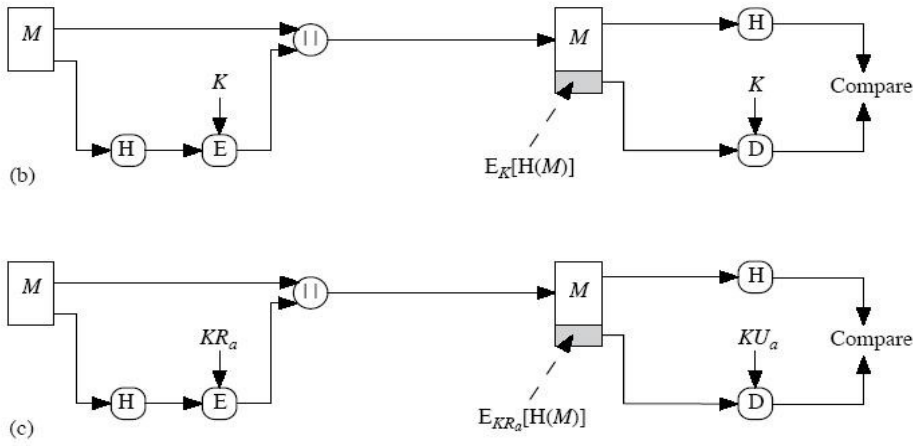


Figure 11.5 Basic Uses of Hash Function (page 1 of 2)

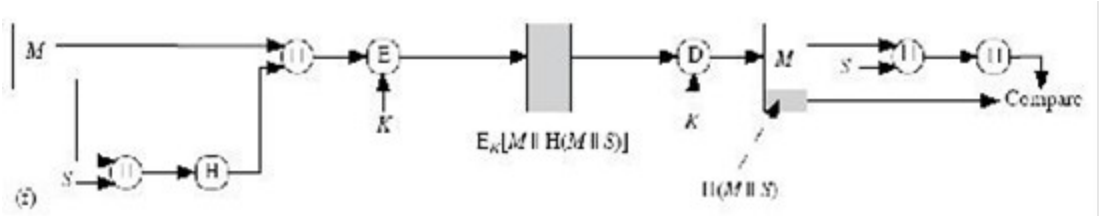


Figure 11.5 Basic Uses of Hash Function (page 2 of 2)

A hash value h is generated by a function H of the form h = H(M), where M is a variable-length message and H(M) is the fixed-length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by re-computing the hash value.

16.2 Requirements for a Hash Function

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.
5. For any given block x, it is computationally infeasible to find y x such that H(y) = H(x). This is sometimes referred to as **weak collision resistance**.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**.

The first three properties are requirements for the practical application of a hash function to message authentication. The fourth property, the one-way property, states that it is easy to generate a code given a message but virtually impossible to generate a message given a code. The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used.The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack, which we examine shortly.

Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

Ci = bi1 ⊕ bi1 ⊕ ⊕ bim
Where
Ci = ith bit of the hash code, 1 ≤i ≤n
m = number of n-bit blocks in the input bij = ith bit in jth block

= XOR operation
Thus, the probability that a data error will result in an unchanged hash value is 2^n. With more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. So if a 128-bit hash value is used, instead of an effectiveness of 2^{128}, the hash function on this type of data has an effectiveness of 2^{112}.

A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed. The procedure can be summarized as follows:

1. Initially set the n-bit hash value to zero.

2. Process each successive n-bit block of data as follows:

a. Rotate the current hash value to the left by one bit. b. XOR the block into the hash value.

Birthday Attacks

Suppose that a 64-bit hash code is used. One might think that this is quite secure. For example, if an encrypted hash code C is transmitted with the corresponding unencrypted

Message M, then an opponent would need to find an M' such that H(M') = H(M) to substitute another message and fool the receiver.
On average, the opponent would have to try about 2^{63} messages to find one that matches the hash code of the intercepted message

However, a different sort of attack is possible, based on **the birthday paradox** The source, A, is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key

1. The opponent generates $2^{m/2 \text{ variations}}$ on the message, all of which convey essentially the same meaning. (Fraudulent message

2. The two sets of messages are compared to find a pair of messages that produces the same hash code. The probability of success, by the birthday paradox, is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.

3. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known.

Thus, if a 64-bit hash code is used, the level of effort required is only on the order of 2^{32} .

Block Chaining Techniques

Divide a message M into fixed-size blocks M1,M2,..., MN and use a symmetric encryption system such as DES to compute the hash code G as follows:

Ho = initial value

Hi = EMi [Hi-1] G = HN

This is similar to the CBC technique, but in this case there is no secret key. As with any hash code, this scheme is subject to the birthday attack, and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable.

Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings.
Here is the scenario; we assume that the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is m bits long:

1. Use the algorithm defined at the beginning of this subsection to calculate the unencrypted hash code G.
2. Construct any desired message in the form Q1, Q2,..., QN2.

3. Compute for Hi = EQi [Hi-1]for $1 \leq i \leq (N-2)$.

4. Generate $2^{m/2}$ random blocks; for each block X, compute EX[HN-2.] Generate an

additional $2^{m/2}$ random blocks; for each block Y, compute DY[G], where D is the decryption function corresponding to E.

5. Based on the birthday paradox, with high probability there will be an X and Y such that EX [HN-2] = DY[G].

6. Form the message Q1, Q2,..., QN-2, X, Y. This message has the hash code G and therefore can be used with the intercepted encrypted signature.

This form of attack is known as a **meet-in-the-middle attack**.

Lecture 17

Security of Hash Functions and Macs

Just as with symmetric and public-key encryption, we can group attacks on hash functions and MACs into two categories: brute-force attacks and cryptanalysis.

Brute-Force Attacks

The nature of brute-force attacks differs somewhat for hash functions and MACs.

Hash Functions

The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm. Recall from our discussion of hash functions that there are three desirable properties:

One-way: For any given code h, it is computationally infeasible to find x such that H(x) = h.

Weak collision resistance: For any given block x, it is computationally infeasible to find y x with H(y) = H(x).

Strong collision resistance: It is computationally infeasible to find any pair (x, y) Such that H(x) = H(y).

For a hash code of length n, the level of effort required, as we have seen is proportional to the following:

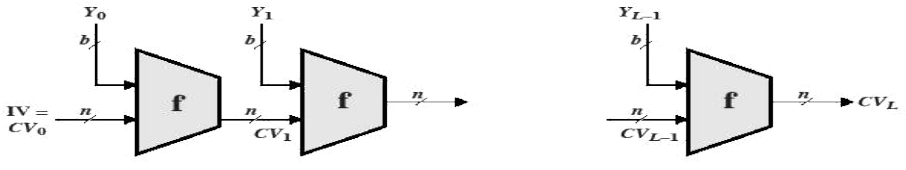


Figure 11.1 Structure of Secure Hash Code

One-way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^{n/2}$

Cryptanalysis

As with encryption algorithms, cryptanalytic attacks on hash functions and MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

Hash Functions

In recent years, there has been considerable effort, and some successes, in developing cryptanalytic attacks on hash functions. To understand these, we need to look at the overall structure of a typical secure hash function, and is the structure of most hash functions in use today, including SHA and Whirlpool.

The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each.

If necessary, the final block is padded to b bits.

The final block also includes the value of the total length of the input to the hash function.The inclusion of the length makes the job of the opponent more difficult.

Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.

The hash algorithm involves repeated use of a **compression function**, f, that takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-bit block) and produces an n-bit output. At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, b > n; hence the term compression. The hash function can be summarized as follows:

CVo = IV = initial n-bit value CVi = f (CVi-1, Yi-1) 1 ≤i ≤L H(M) = CVL

Where the input to the hash function is a message M consisting of the blocks Yo, Y1,..., YL-1.

The structure can be used to produce a secure hash function to operate on a message of any length.

Message Authentication Codes

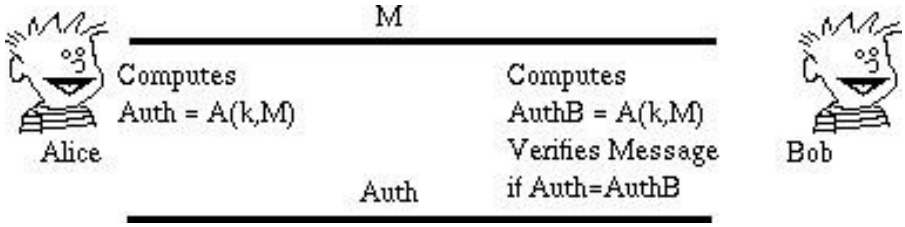
There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs. Further, far less work has been done on developing such attacks.

Message Authentication.

- Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)

It is electronic equivalent of a signature on a message. An **authenticator**, **signature**, or **message authentication code (MAC)** is sent along with the message. The MAC is generated via some

algorithm which depends on both the message and some (public or private) key known only to the sender and receiver. The message may be of any length. The MAC may be of any length, but more often is some fixed size, requiring the use of some **hash function** to condense the message to the required size if this is not achieved by the authentication scheme. Need to consider replay problems with message and MAC. Require a message sequence number, timestamp or negotiated random values



Authentication using Private-key Ciphers

If a message is being encrypted using a session key known only to the sender and receiver, then the message may also be authenticated since only sender or receiver could have created it. Any interference will corrupt the message (provided it includes sufficient redundancy to detect change).

Hashing Functions

Hashing functions are used to condense an arbitrary length message to a fixed size, usually for subsequent signature by a digital signature algorithm. It is a good cryptographic hash function h should have the following properties:

- h should destroy all holomorphic structures in the underlying public key cryptosystem (be unable to compute hash value of 2 messages combined given their individual hash values)
- h should be computed on the entire message
- h should be a one-way function so that messages are not disclosed by their signatures
- it should be computationally infeasible given a message and its hash value to compute another message with the same hash value
- should resist **birthday attacks** (finding any 2 messages with the same hash value, perhaps by iterating through minor permutations of 2 messages)

It is usually assumed that the hash function is public and not keyed. Traditional CRCs do not satisfy the above requirements. Length should be large enough to resist birthday attacks (64-bits is now regarded as too small, 128-512 proposed)

Lecture 18

MD2, MD4 and MD5

Family of one-way hash functions by Ronald Rivest. MD2 is the oldest, produces a 128-bit hash value, and is regarded as slower and less secure than MD4 and MD5. MD4 produces a 128-bit hash of the message, using bit operations on 32-bit operands for fast implementation.

MD4 overview

pad message so its length is 448 mod 512

append a 64-bit message length value to message

initialise the 4-word (128-bit) buffer (A,B,C,D)

process the message in 16-word (512-bit) chunks, using 3 rounds of 16 bit operations

each on the chunk & buffer

output hash value is the final buffer value

Some progress at cryptanalysing MD4 has been made, with a small number of collisions having been found. MD5 was designed as a strengthened version, using four rounds, a little more complex than in MD4. A little progress at cryptanalyzing MD5 has been made with a small number of collisions

having been found. Both MD4 and MD5 are still in use and considered secure in most practical applications. Both are specified as Internet standards (MD4 in RFC1320, MD5 in RFC1321).

SHA (Secure Hash Algorithm)
SHA was designed by NIST & NSA and is the US federal standard for use with the DSA signature scheme (nb the algorithm is SHA, the standard is SHS). It produces 160-bit hash values

SHA overview

pad message so its length is a multiple of 512 bits

initialise the 5-word (160-bit) buffer (A,B,C,D,E) to

(67452301,efcdab89,98badcfe,10325476,c3d2e1f0)

process the message in 16-word (512-bit) chunks, using 4 rounds of 20 bit operations each on the chunk & buffer

output hash value is the final buffer value

SHA is a close relative of MD5, sharing much common design, but each having differences. SHA has very recently been subject to modification following NIST identification of some concerns, the exact nature of which is not public. Current version is regarded as secure.

Lecture 19
Digital Signature Schemes

- public key signature schemes
- the private-key signs (creates) signatures, and the public-key verifies signatures

· only the owner (of the private-key) can create the digital signature, hence it can be used to verify who created a message

· anyone knowing the public key can verify the signature (provided they are confident of the identity of the owner of the public key - the key distribution problem)

· usually don't sign the whole message (doubling the size of information exchanged), but just a **hash** of the message

- digital signatures can provide non-repudiation of message origin, since an asymmetric algorithm is used in their creation, provided suitable timestamps and redundancies are incorporated in the signature

RSA

- RSA encryption and decryption are commutative, hence it may be used directly as a digital signature scheme

o given an RSA scheme {(e,R), (d,p,q)}

- to **sign** a message, compute:

o $S = M^d \pmod R$

- to **verify** a signature, compute:

o $M = S^e \pmod R = M^{e.d} \pmod R = M \pmod R$

- thus know the message was signed by the owner of the public-key

- would seem obvious that a message may be encrypted, then signed using RSA without increasing it size

o but have blocking problem, since it is encrypted using the receivers modulus, but signed using the senders modulus (which may be smaller)

o several approaches possible to overcome this

- more commonly use a hash function to create a separate MDC which is then signed

Lecture 20
El Gamal Signature Scheme

- whilst the ElGamal encryption algorithm is not commutative, a closely related signature scheme exists

- El Gamal Signature scheme

- given prime p, public random number g, private (key) random number x, compute

o $y = g^x \pmod p$

- public key is (y,g,p)

- nb (g,p) may be shared by many users
- p must be large enough so discrete log is hard

· private key is (x)

· to **sign** a message M

- choose a random number k, GCD(k,p-1)=1
- compute a = gk(mod p)
- use extended Euclidean (inverse) algorithm to solve
- M = x.a + k.b (mod p-1)
- the signature is (a,b), k must be kept secret
- (like ElGamal encryption is double the message size)

· to **verify** a signature (a,b) confirm:

- $y^a.a^b(\text{mod } p) = g^M(\text{mod } p)$

Example of ElGamal Signature Scheme

· given p=11, g=2

· choose private key x=8

· compute

- $y = g^x(\text{mod } p) = 2^8(\text{mod } 11) = 3$

· public key is y=3,g=2,p=11)

· to sign a message M=5

- choose random k=9
- confirm gcd(10,9)=1
- compute

  $a = gk(\text{mod } p) = 29(\text{mod } 11) = 6$

- solve
  - $M = x.a+k.b \pmod{p-1}$
  - $5 = 8.6+9.b \pmod{10}$
  - giving b = 3
  - signature is (a=6,b=3)

- to verify the signature, confirm the following are correct:
  - $y^{a.ab} \pmod{p} = g^{M} \pmod{p}$
  - $36.63 \pmod{11} = 25 \pmod{11}$

DSA (Digital Signature Algorithm)

- DSA was designed by NIST & NSA and is the US federal standard signature scheme (used with SHA hash alg)
  - DSA is the algorithm, DSS is the standard
  - There was considerable reaction to its announcement!
    - debate over whether RSA should have been used
    - debate over the provision of a signature only alg
- DSA is a variant on the ElGamal and Schnorr algorithms
- description of DSA
  - $p = 2^{L}$ a prime number, where L= 512 to 1024 bits and is a multiple of 64
  - q a 160 bit prime factor of p-1
  - $g = h^{(p-1)/q}$ where h is any number less than p-1 with $h^{(p-1)/q} \pmod{p} > 1$
  - x a number less than q
  - $y = g^{x} \pmod{p}$
- to **sign** a message M

- **o** generate random k, k<q
- **o** compute
  - r = $(g^k \pmod p) \pmod q$
  - s = $k^{-1}.SHA(M) + x.r \pmod q$
- **o** the signature is (r,s)

· to **verify** a signature:

- **o** $w = s^{-1} \pmod q$
- **o** $u1 = (SHA(M).w) \pmod q$
- **o** $u2 = r.w \pmod q$
- **o** $v = (gu1.yu2 \pmod p) \pmod q$
- **o** if v=r then the signature is verified

· comments on DSA

o was originally a suggestion to use a common modulus, this would make a tempting target, discouraged

o it is possible to do both ElGamal and RSA encryption using DSA routines, this was probably not intended :-)

o DSA is patented with royalty free use, but this patent has been contested, situation unclear

o Gus Simmons has found a subliminal channel in DSA, could be used to leak the private key from a library - make sure you trust your library implementer

Module -4: [9L]
SECURITY PRACTICE AND SYSTEM SECURITY

Lecture 21

21.1: Authentication applications:

a) INTRODUCTION TO KERBEROS:

Kerberos is an authentication protocol that lets clients and servers reliably verify each other's identity before establishing a network connection. Developed at MIT in the late 1980s, Kerberos takes its name from the three headed dog on Greek mythology that guards the entrance of Hades. Instead of guarding the underworld, today's Kerberos brings a measure of security to a distributed computer environment, where one computer can access the resources of any other machine on a network. Like Kerberos the dog, Kerberos the protocol has three heads; two principals and one trusted third party.

The basic Kerberos authentication process proceeds as follows: A client sends a request to the authentication server (AS) requesting 'credentials' for a given server. The AS responds with these credentials, encrypted in the client's key. The credentials consist of 1) a 'ticket' for the server and 2) a temporary encryption key (often called a "session key"). The client transmits the ticket (which contains the client's identity and a copy of the session key, all encrypted in the server's key) to the server. The session key (now shared by the client and server) is used to authenticate the client, and may optionally be used to authenticate the server. It may also be used to encrypt further communication between the two parties or to exchange a separate sub-session key to be used to encrypt further communication. The implementation can consist of one or more authentication servers running on physically secure hosts. The authentication servers maintain a database of principals (i.e., users and servers) and their secret keys. Code libraries provide encryption and implement the Kerberos protocol. In order to add authentication to its transactions, a typical network application adds one or two calls to the Kerberos library, which results in the transmission of the necessary messages to achieve authentication. The Kerberos protocol consists of several exchanges. A client can ask a Kerberos server for credentials by two methods. In the first approach, the client sends a clear text request for a ticket for the desired server to the AS. The reply is sent encrypted in the client's secret key. Usually this request is for a ticket-granting ticket (TGT) that can later be used with the ticket-granting server (TGS). In the second method, the client sends a request to the TGS. The client sends the TGT to the TGS in the same manner as if it were contacting any other application server that requires Kerberos credentials. The reply is encrypted in the session key from the TGT. Once obtained, credentials may be used to verify the identity of the principals in transaction, to ensure the integrity of messages exchanged between them, or to preserve privacy of the messages. The application is free to choose whatever protection may be necessary. To verify the identities of the principals in a transaction, the client transmits the ticket to the server. Since the ticket is sent in the clear and might be intercepted and reused by an attacker, additional information is sent to prove that the

message was originated by the principal to whom the ticket was issued. This information is encrypted in the session key, and includes a timestamp. The timestamp proves that the message was recently generated and is not a replay. Encrypting the authenticator in the session key proves that a party possessing the session key generated it. Since no one except the requesting principal and the server know the session key this guarantees the identity of the client. The integrity of the messages exchanged between principals can also be guaranteed using the session key. This approach provides detection of both replay attacks and message stream modification attacks. It is accomplished by generating and transmitting a hash function of the client's message, keyed with the session key. Privacy and integrity of the messages exchanged between principals can be secured by encrypting the data to be passed using the session key passed in the ticket, and contained in the credentials.

ADVANTAGES OF KERBEROS:

As a distributed security service, Kerberos provides authentication, confidentiality, and integrity security capabilities. Those three elements of security are essential to any organization that wants to operate a secure environment. Kerberos uses encryption to provide each of these three security services and supports both public key cryptography (asymmetric) and secret key (symmetric) encryption for authentication; however, the core functionality of any Kerberos implementation relies on secret key encryption. [Hynes 2001] As a side-effect of the dual-key encryption scheme employed in the Kerberos model, a service-session key is generated which constitutes a shared secret between a particular client system and a particular service. This shared secret may be used as a key for encrypting the conversation between the client and the target service, further enhancing the security of Kerberized transactions. One of the most significant benefits of using Kerberos is its standing as a relatively mature, IETF standards-based protocol (RFC 1510) supported by Windows 2000, Linux, and Windows platforms. Unlike many of its proprietary counterparts, Kerberos has been scrutinized by many of the top programmers, cryptologists and security experts in the industry. This public scrutiny has ensured and continues to ensure that any new weaknesses discovered in the protocol or its underlying security model will be quickly analyzed and corrected. Another advantage is the tickets passed between clients and servers in the Kerberos authentication model include timestamp and lifetime information. This allows Kerberos clients and Kerberized servers to limit the duration of their users' authentication. While the specific length of time for which a user's authentication remains valid after his initial ticket issued is implementation dependent, Kerberos systems typically use small enough ticket lifetimes to prevent brute-force and replay attacks. In general, no authentication ticket should have a lifetime longer than the expected time required to crack the encryption of the ticket.

DISADVANTAGES OF KERBEROS:

Kerberos was designed for use with single-user client systems. In the more general case, where a client system may itself be a multi-user system, the Kerberos authentication scheme can fall prey to a variety of ticket stealing and replay attacks. The overall security of multiuser Kerberos client

systems (file system security, memory protection, etc.) is therefore a limiting factor in the security of Kerberos authentication. No amount of cleverness in the implementation of a Kerberos authentication system can replace good system administration practices on Kerberos client and server machines. Because Kerberos uses a mutual authentication model, it is necessary for both client machines and service providers (servers) to be designed with Kerberos authentication in mind. Many proprietary applications already provide support for Kerberos or will be providing Kerberos support in the near future. Some legacy systems and many locally-written and maintained packages, however, were not designed with any third-party authentication mechanism in mind, and would have to be re-written (possibly extensively) to support Kerberos authentication. The Kerberos authentication model is vulnerable to brute-force attacks against the KDC (the initial ticketing service and the ticket-granting service). The entire authentication system depends on the trust ability of the KDC(s), so anyone who can compromise system security on a KDC system can theoretically compromise the authentication of all users of systems depending on the KDC. Again, no amount of cleverness in the design of the Kerberos system can take the place of solid system administration practices employed in managing the Kerberos KDC(s).

b) X.509:

In cryptography, X.509 is a standard defining the format of public key certificates. X.509 certificates are used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure protocol for browsing the web. They are also used in offline applications, like electronic signatures. An X.509 certificate contains a public key and an identity (a hostname, or an organization, or an individual), and is either signed by a certificate authority or self-signed. When a certificate is signed by a trusted certificate authority, or validated by other means, someone holding that certificate can rely on the public key it contains to establish secure communications with another party, or validate documents digitally signed by the corresponding private key.

X.509 also defines certificate revocation lists, which are a means to distribute information about certificates that have been deemed invalid by a signing authority, as well as a certification path validation algorithm, which allows for certificates to be signed by intermediate CA certificates, which are, in turn, signed by other certificates, eventually reaching a trust anchor.

LECTURE22:

22.1: Firewalls:

A firewall is software used to maintain the security of a private network. Firewalls block unauthorized access to or from private networks and are often employed to prevent unauthorized Web users or illicit software from gaining access to private networks connected to the Internet. A firewall may be implemented using hardware, software, or a combination of both.

A firewall is recognized as the first line of defense in securing sensitive information. For better safety, the data can be encrypted.

Firewalls generally use two or more of the following methods:

- Packet Filtering: Firewalls filter packets that attempt to enter or leave a network and either accept or reject them depending on the predefined set of filter rules.
- Application Gateway: The application gateway technique employs security methods applied to certain applications such as Telnet and File Transfer Protocol servers.
- Circuit-Level Gateway: A circuit-level gateway applies these methods when a connection such as Transmission Control Protocol is established and packets start to move.
- Proxy Servers: Proxy servers can mask real network addresses and intercept every message that enters or leaves a network.
- State ful Inspection or Dynamic Packet Filtering: This method compares not just the header information, but also a packet's most important inbound and outbound data parts. These are then compared to a trusted information database for characteristic matches. This determines whether the information is authorized to cross the firewall into the network.

Roles of Firewalls:
Firewalls are systems which protect networks or network devices, such as industrial PCs, control systems, cameras, etc., from unauthorized access by preventing network traffic to or from this systems.The fundamental technical function of any firewall are to filter packets.

LECTURE23:
23.1:Types of Firewalls:
One of the major challenges that companies face when trying to secure their sensitive data is finding the right tools for the job. Even for a common tool such as a firewall, many businesses might not have a clear idea of how to find the right firewall (or firewalls) for their needs, how to configure those firewalls, or why such firewalls might be necessary.

The first step in finding the right firewalls to protect your company's data is to know what kind of firewalls there are. Right now, there are five different types of firewall architectures, broadly speaking:

- Packet-filtering firewalls
- Stateful inspection firewalls
- Circuit-level gateways
- Application-level gateways (a.k.a. proxy firewalls)
- Next-gen firewalls

How do these firewalls work? And, which ones are the best for your business' cybersecurity needs? Here are a few brief explainers:

- Packet-Filtering Firewalls

As the most “basic” and oldest type of firewall architecture, packet-filtering firewalls basically create a checkpoint at a traffic router or switch. The firewall performs a simple check of the data packets coming through the router—inspecting information such as the destination and origination IP address, packet type, port number, and other surface-level information without opening up the packet to inspect its contents.

If the information packet doesn’t pass the inspection, it is dropped.

The good thing about these firewalls is that they aren’t very resource-intensive. This means they don’t have a huge impact on system performance and are relatively simple. However, they’re also relatively easy to bypass compared to firewalls with more robust inspection capabilities.

- Circuit-Level Gateways

As another simplistic firewall type that is meant to quickly and easily approve or deny traffic without consuming significant computing resources, circuit-level gateways work by verifying the transmission control protocol (TCP) handshake. This TCP handshake check is designed to make sure that the session the packet is from is legitimate.

While extremely resource-efficient, these firewalls do not check the packet itself. So, if a packet held malware, but had the right TCP handshake, it would pass right through. This is why circuit-level gateways are not enough to protect your business by themselves.

- Stateful Inspection Firewalls

These firewalls combine both packet inspection technology and TCP handshake verification to create a level of protection greater than either of the previous two architectures could provide alone.

However, these firewalls do put more of a strain on computing resources as well. This may slow down the transfer of legitimate packets compared to the other solutions.

- Proxy Firewalls (Application-Level Gateways)

Proxy firewalls operate at the application layer to filter incoming traffic between your network and the traffic source—hence, the name “application-level gateway.” Rather than letting traffic connect directly, the proxy firewall first establishes a connection to the source of the traffic and inspects the incoming data packet.

This check is similar to the stateful inspection firewall in that it looks at both the packet and at the TCP handshake protocol. However, proxy firewalls may also perform deep-layer packet

inspections, checking the actual contents of the information packet to verify that it contains no malware.
Once the check is complete, and the packet is approved to connect to the destination, the proxy sends it off. This creates an extra layer of separation between the "client" (the system where the packet originated) and the individual devices on your network—obscuring them to create additional anonymity and protection for your network.

If there's one drawback to proxy firewalls, it's that they can create significant slowdown because of the extra steps in the data packet transferal process.

- Next-Generation Firewalls

Many of the most recently-released firewall products are being touted as "next-generation" architectures. However, there is not as much consensus on what makes a firewall truly next-gen.

Some common features of next-generation firewall architectures include deep-packet inspection (checking the actual contents of the data packet), TCP handshake checks, and surface-level packet inspection. Next-generation firewalls may include other technologies as well, such as intrusion prevention systems (IPSs) that work to automatically stop attacks against your network.

The issue is that there is no one definition of a next-generation firewall, so it's important to verify what specific capabilities such firewalls have before investing in one.

LECTURE24:

24.1:Firewall Design Principles

A firewall is a dedicated hardware, or software or a combination of both, which inspects network traffic passing through it, and denies or permits passage based on a set of rules.

FIREWALL CHARACTERISTICS

The following are the Firewall Capabilities:

- A firewall defines a single choke point that keeps unauthorized users out the protected network……..
- A firewall provides a location for monitoring security-related events. Audits and alarms can be implemented on the firewall system.
- A firewall is a convenient platform for several Internet functions that are not security related.
- A firewall can serve as the platform for IPSec. Using the tunnel mode capability, the firewall can be used to implement virtual private network.

Firewall Limitations

- The firewall can not protect against attacks that bypass the firewall (dial-up…).
- The firewall does not protect against internal threats.
- The firewall can not protect against the transfer of virus-infected programs or files.

DESIGN GOALS

- All traffic from inside to outside, and vice verse, must pass through the firewall.
- Only authorized traffic, as defined by the local security policy, will be allowed to pass.
- The firewall itself is immune to penetration. This implies the use of a trusted system with a secure operating system

METHODS OF CONTROL IN FIREWALL

- User control

Only authorized users are having access to the other side of the firewall

- Access control

The access over the firewall is restricted to certain services. A service is characterized e.g. by IP address and port number.

- Behavior control

For an application, the allowed usage scenarios are known. E.g. filters for e-mail attachments (virus removing)

- Direction control

Different rules for traffic into the Intranet and outgoing traffic to the Internet can be defined.

LECTURE 25:

25.1:Secure Electronic Transaction (SET) Protocol

Secure Electronic Transaction or SET is a system which ensures security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied on those payments. It uses different encryption and hashing techniques to secure payments over internet done through credit cards. SET protocol was supported in development by major organizations like Visa, Mastercard, Microsoft which provided its Secure Transaction Technology (STT) and NetScape which provided technology of Secure Socket Layer (SSL).

SET protocol restricts revealing of credit card details to merchants thus keeping hackers and thieves at bay. SET protocol includes Certification Authorities for making use of standard Digital Certificates like X.509 Certificate.

Before discussing SET further, let’s see a general scenario of electronic transaction, which includes client, payment gateway, client financial institution, merchant and merchant financial institution.

Requirements in SET :

SET protocol has some requirements to meet, some of the important requirements are :

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is intended user or not and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.
- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.
- SET also needs to provide interoperability and make use of best security mechanisms.

LECTURE 26:

26.1:Intruder

A significant issue for networked systems

– hostile or unwanted access

– either via network or local

● Classes of intruders:

– masquerader

– misfeasor

– clandestine user

● Varying levels of competence

26.2: Intrusion Detection System (IDS)

There are a multiple ways detection is performed by an IDS. In signature-based detection, a pattern or signature is compared to previous events to discover current threats. This is useful for finding already known threats, but does not help in finding unknown threats, variants of threats or hidden threats.

Another type of detection is anomaly-based detection, which compares the definition or traits of a normal action against characteristics marking the event as abnormal.

There are three primary components of an IDS:

- Network Intrusion Detection System (NIDS): This does analysis for traffic on a whole subnet and will make a match to the traffic passing by to the attacks already known in a library of known attacks.
- Network Node Intrusion Detection System (NNIDS): This is similar to NIDS, but the traffic is only monitored on a single host, not a whole subnet.
- Host Intrusion Detection System (HIDS): This takes a “picture” of an entire system’s file set and compares it to a previous picture. If there are significant differences, such as missing files, it alerts the administrator.

LECTURE 27:

27.1 VIRUSES AND RELATED THREATS

VIRUS

Computer Virus is a kind of malicious software written intentionally to enter a computer without the user's permission or knowledge, with an ability to replicate itself, thus continuing to spread. Some viruses do little but replicate others can cause severe harm or adversely effect program and performance of the system.

TYPES OF VIRUS

- Resident Viruses

This type of virus is a permanent which dwells in the RAM memory. From there it can overcome and interrupt all of the operations executed by the system: corrupting files and programs that are opened, closed, copied, renamed etc.
Examples include: Randex, CMJ, Meve, and MrKlunky.

- Boot Virus

This type of virus affects the boot sector of a floppy or hard disk. This is a crucial part of a disk, in which information on the disk itself is stored together with a program that makes it possible to boot (start) the computer from the disk.

The best way of avoiding boot viruses is to ensure that floppy disks are write-protected and never start your computer with an unknown floppy disk in the disk drive.
Examples of boot viruses include: Polyboot.B, AntiEXE.

- Macro Virus

Macro viruses infect files that are created using certain applications or programs that contain macros. These mini-programs make it possible to automate series of operations so that they are performed as a single action, thereby saving the user from having to carry them out one by one.
Examples of macro viruses: Relax, Melissa.A, Bablas, O97M/Y2K.

- Polymorphic Virus

Polymorphic viruses encrypt or encode themselves in a different way (using different algorithms and encryption keys) every time they infect a system.This makes it impossible for anti-viruses to find them using string or signature searches (because they are different in each encryption) and also enables them to create a large number of copies of themselves.
Examples include: Elkern, Marburg, Satan Bug, and Tuareg.

- Parasitic Viruses

Parasitic viruses modify the code of the infected file. The infected file remains partially or fully functional.Parasitic viruses are grouped according to the section of the file they write their code to:

- Prepending: the malicious code is written to the beginning of the file
- Appending: the malicious code is written to the end of the file
- Inserting: the malicious code is inserted in the middle of the file

Inserting file viruses use a variety of methods to write code to the middle of a file: they either move parts of the original file to the end or copy their own code to empty sections of the target file.

- WORMS

A computer worm is a self-replicating computer program. It uses a network to send copies of itself to other nodes (computers on the network) and it may do so without any user intervention. Unlike a virus, it does not need to attach itself to an existing program.

- LOGIC BOMB

A logic bomb is a piece of code intentionally inserted into a software system that will set off a malicious function when specified conditions are met. For example, a programmer may hide a piece of code that starts deleting files (such as the salary database trigger), should they ever leave the company.

- TROJAN HORSES

The Trojan horse, also known as trojan, in the context of computing and software, describes a class of computer threats (malware) that appears to perform a desirable function but in fact performs undisclosed malicious functions that allow unauthorized access to the host machine, giving them the ability to save their files on the user's computer or even watch the user's screen and control the computer.

- MALWARE

Malwares software designed to infiltrate or damage a computer system without the owner's informed consent. The expression is a general term used by computer professionals to mean a variety of forms of hostile, intrusive, or annoying software or program code.

- SPYWARE

Spyware is computer software that is installed surreptitiously on a personal computer to collect information about a user, their computer or browsing habits without the user's informed consent.

27.3: Countermeasures

In computer security a countermeasure is an action, device, procedure, or technique that reduces a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken.

LECTURE 28:

28.1: Trusted systems:

In the security engineering subspecialty of computer science, a trusted system is a system that is relied upon to a specified extent to enforce a specified security policy. This is equivalent to saying that a trusted system is one whose failure would break a security policy (if a policy exists that the trusted system is trusted to enforce).

The meaning of the word "trust" is critical, as it does not carry the meaning that might be expected in everyday usage. A system trusted by a user, is one that the user feels safe to use, and trusts to do tasks without secretly executing harmful or unauthorised programs; while trusted computing refers to whether programs can trust the platform to be unmodified from that expected, whether or not those programs are innocent, malicious or execute tasks that are undesired by the user.

Cryptography originated about 4000 years ago, and the world of cryptography has evolved a lot since then. Today ‘Cryptography’ is omnipresent in our lives without most of us realizing it. The fundamental aspect of ‘Cryptography’ has remained the same through time which is to hide information in transit and make it available only for the intended recipients. We will see the basic types of cryptography followed by the application and use of cryptography in real life.

Cryptographic principles:

Cryptography involves the use of terms like plain text, cipher text, algorithm, key, encryption, and decryption. ‘Plain text’ is the text or message that needs to be transmitted to the intended recipients and which needs to be hidden. ‘Cipher text’ on the other hand, is the text that has been transformed by algorithms and which is gibberish.

The process of converting the information from ‘plain text’ to ‘cipher text’ is known as ‘encryption.’ On similar lines, the process of converting ‘cipher text’ to ‘plain text’ is decryption.

The complex mathematical formula that is used to convert ‘plain text’ to ‘cipher text’ is known as ‘algorithm.’ Further, both the sender and the receiver have similar or different “keys” to encrypt and decrypt the message. A “key” is a “value that comprises a large sequence of random bits” (Harris 2008). The larger the key size, the more difficult will it be to crack the algorithm. The “algorithm” and the “key” are the two important components of a cryptosystem.

Kerckhoff’s principle:

Auguste Kerckhoff in 1883 stated that encryption algorithms should be made public and the "keys" be kept secret. While the academic sector is in favor of the idea, the other sectors are not in perfect synchronization with this idea. The encryption algorithms in the academic sector are made public to enable one to find new vulnerabilities and improve their algorithm. The government sector prefers to keep encryption algorithms private as an additional step to security.

There are two types of encryption – symmetric encryption and asymmetric encryption.

1. In symmetric encryption, the sender and receiver use a separate instance of the same "key" to encrypt and decrypt messages. Symmetric encryption heavily relies on the fact that the keys "must" be kept secret. Distributing the key in a secure way is one of the primary challenges of symmetric encryption. This is also known as the "key distribution problem." Further, as per Kerckhoff's principle, since the algorithm in symmetric encryption is public, anyone can have access to it. It is the "key" that is the vital link in symmetric cryptography and which must be kept secret. It cannot be lost or misplaced. If the individual keys are misplaced, the message can be decrypted by scrupulous elements. Further, symmetric cryptography upholds only the 'confidentiality' tenet of the CIA triad. Confidentiality is making sure that the information that is sent is received only by the intended recipients.

One of the main advantages of symmetric cryptography is that it is much faster than asymmetric cryptography. Two important disadvantages of symmetric encryption include key distribution problem and. "key management problem." When the number of keys needed grows in number with growth in users, it becomes the "Key management problem." This happens because each set of users needs a pair of instance keys.

Some examples of symmetric encryption are DES (Data encryption standard), Triple DES (3DES) and Blowfish.

2. Asymmetric encryption is when the sender and the receiver use different "keys" to encrypt and decrypt messages. Here a public key is used to encrypt the message, and a private key is used to decrypt the message. In asymmetric cryptography, the public keys are widely known – whereas the private key is kept protected. The two keys are mathematically related. In spite of being mathematically related, one will not be able to calculate the private key even if the public key is known.

Asymmetric cryptography upholds the security tenets of authenticity and non-repudiation. When one encrypts a message with a private key and sends it, authenticity (proof of identity) is established. Similarly, non-repudiation (cannot deny sending it) is established when a message is encrypted with a private key. It should be noted that any key (public or private) can be used to encrypt and any key (public or private) can be used to be decrypt as well.

One of the main disadvantages of asymmetric encryption is that it is slow when compared with symmetric encryption. The main advantage of asymmetric encryption is the lack of “key distribution problem” and “key management problem.”

Some examples of asymmetric encryption algorithms are, RSA (Rivest, Shamir, Adleman), AES (Advanced Encryption Standard) and El Gamal.

Having seen the two basic types of encryption, let us next see the practical applications of cryptography. We will see cryptography being implemented in mobile messaging tool, “Whatsapp,” “Digital signatures” and HTTP (HTTP over SSL or ‘Secure Sockets Layer’)

Module -5: [6L]
E-MAIL, IP,AND WEB SECURITY

LECTURE 29:

29.1: Encryption in Whatsapp:

‘Whatsapp’ is currently one of the most popular mobile messaging software. It is available for different platforms such as Android, Windows Phone, and iPhone. ‘Whatsapp’ also enables users

to make free calls with other users. In the latest version of 'Whatsapp,' the conversations and calls are "end-to-end" encrypted.

- What does end-to-end encryption mean?

In end-to-end encryption, only the data is encrypted. The headers, trailers, and routing information are not encrypted. End-to-to end encryption in Whatsapp has been developed in collaboration with 'Open Whisper Systems.'

End-to-end encryption makes sure that a message that is sent is received only by the intended recipient and none other. Whatsapp has ensured, that even "it" cannot read the messages bolstering a very strong messaging platform. It also means that outsiders or third party individuals cannot snoop on conversations between intended recipients as well.

- How is end-to-end encryption in Whatsapp implemented?

Whatsapp end-to-end encryption is implemented using asymmetric cryptography or public key systems. Recall, that in asymmetric encryption, when one key is used to encrypt (here, the public key), the other key is used to decrypt (here, the private key) the message.

Once 'Whatsapp' is installed on a user's smartphone, the public keys of 'Whatsapp' clients are registered with the Whatsapp server. It is important to note here that the private key is not stored on Whatsapp servers.

Encrypted session between Whatsapp clients:
Once the client is registered, an encrypted session is created between two clients willing to take part in a conversation. A session needs to be re-created only when the device is changed or when the Whatsapp software is re-installed.

If for example, client1 wants to send a message to client 2, the public keys of the client2 are retrieved from the Whatsapp server, and this used to encrypt the message and send it to the client2. Client2 then decrypts the message with his own private key. "Once a session has been established, clients exchange messages that are protected with a Message Key using AES256 in CBC mode for encryption and HMAC-SHA256 for authentication" (WhatsApp Encryption Overview 2016)

- Digital signatures:

Having seen how encryption is implemented in Whatsapp, let us see the next practical application of cryptography – Digital signatures. Digital signatures are signatures applied digitally. They enforce the concepts of authentication, non-repudiation, and confidentiality. Wikipedia defines digital signatures the following way: "A digital signature is a mathematical

scheme for demonstrating the authenticity of a digital message or documents." (Digital Signature 2016).

With the world being more technically tuned now, business transactions occurring all around the world are fairly common. Manually signing a document and transferring it to different locations is time-consuming. This time lag might not bode well for either the customer or the client. By digitally signing the documents, the business transaction will be completed on time.

Consider another case when two parties are required to sign documents relating to a business transaction. The two parties might never have met each other and might not trust each other. Digital signatures thus ensure timeliness and authenticity of business transactions.

LECTURE 30:

30.1:E-mail Security :

Email security describes various techniques for keeping sensitive information in email communication and accounts secure against unauthorized access, loss, or compromise.

Security Services for E-mail:

There are multiple ways to secure email accounts, and for enterprises, it's a two-pronged approach encompassing employee education and comprehensive security protocols. Best practices for email security include:

- Engage employees in ongoing security education around email security risks and how to avoid falling victim to phishing attacks over email.
- Require employees to use strong passwords and mandate password changes periodically.
- Utilize email encryption to protect both email content and attachments.
- Implement security best practices for BYOD if your company allows employees to access corporate email on personal devices.
- Ensure that webmail applications are able to secure logins and use encryption.
- Implement scanners and other tools to scan messages and block emails containing malware or other malicious files before they reach your end users.
- Implement a data protection solution to identify sensitive data and prevent it from being lost via email

30.2:END USER EMAIL SECURITY BEST PRACTICE

There are also some important best practices that end users should follow to ensure secure email usage. Arming your employees with the know-how to avoid risky behaviors can make a substantial impact on your company's ability to reduce risks associated with email. Email security best practices for end users/employees include:

- Never open attachments or click on links in email messages from unknown senders.
- Change passwords often and use best practices for creating strong passwords.

- Never share passwords with anyone, including co-workers.
- Try to send as little sensitive information as possible via email, and send sensitive information only to recipients who require it.
- Use spam filters and anti-virus software.
- When working remotely or on a personal device, use VPN software to access corporate email.
- Avoid accessing company email from public wi-fi connections.

By educating employees on email security and implementing the proper measures to protect email, enterprises can mitigate many of the risks that come with email usage and prevent sensitive data loss or malware infections via email.

30.3:Attacks possible through E-mail:

- Types of Email Attack

There are different types of attack vectors used by hackers to target email systems. It's important to note that while different attack vectors may employ different methods, they ultimately have a specific purpose when executing the attack.

The most common techniques used to attack emails include identity theft, phishing, virus and spam emails. Let us take a closer look into the common techniques that threaten email security.

- Identity Theft

  Many organizations these days are either using Microsoft Office 365, G Suite, Zoho or similar services to manage their email systems. Other than hosting emails, services like these offer a suite of useful business tools to manage information in one place. Some apps in the suite include added cloud storage space, project management and collaboration tools, Office suite and much more.

  Since they are all part of the same suite as the email service, end users do not need a separate set of login credentials to access them. Regardless of whether a company uses the above-mentioned services or their own proprietary service, they all tend to face the same consequences when a hacker manages to get hold of a user's identity (i.e. login credentials).

  Employees usually use the suite to store confidential data which will, in a short period of time, be exposed if an attacker gains a handle on the employee's email account. Email identity theft can have much bigger consequences than it did a few years ago.

- Phishing Attacks

Phishing is one of the fastest growing attack vectors. For hackers, it is a tried and tested method that has been successfully working for more than a decade. In fact, it has been more than two decades since the first reported phishing attack in 1995.

As the internet grew, so did the number of users having a minimum of two email accounts. Hackers now have far wider reach than ever before. According to a recent report by Tripwire, there were 9,576 phishing incidents recorded in 2015, with 916 of them reporting a breach of data.

Phishing as a tactic employs several different techniques. Each type of attack has its own target audience and purpose.

In an attack called Pharming, the hacker changes IP address associated with the website. This redirects the user to the malicious website despite entering a correct domain name in the URL. Deceptive phishing scams the user by posing as a legitimate website and scares them into paying money. Spear phishing uses the same technique as deceptive phishing, except that this attack makes the user hand over their personal data. According to a report by Symantec, spear-phishing campaigns targeting employees increased 55% in 2015. You can read their complete report.

- Virus

Attacking with a virus through email is another form using email as a vector. Creating a virus and implementing it requires a meticulous amount of planning, an activity more likely to be conceived and executed by a group rather than an individual.
A targeted virus can have one specific or multiple purposes. Regardless of that, email itself is rarely a target, merely the first stage of the attack. If the attack is successful, the virus could quickly spread across the network in a short time and can even have the ability to shut down the complete network.

Even the simplest virus will attempt to lure the end user into downloading an attachment. Masquerading as documents, they are in fact files which if executed could either take control over the host or even lead to the consequence mentioned above.

In a 2015 report , Kaspersky Lab’s web antivirus detected 121,262,075 unique malicious objects: scripts, exploits, executable files, etc.

- Spam

Spam is the most commonly known form of email attack. Perhaps the reason is because we all have a “spam” folder within our email accounts where we receive unwanted emails or emails we didn’t subscribe to.

This is likely why even people from non-IT backgrounds know what a spam email is, although they are usually thought of as merely harmless emails which they can directly delete without even bothering to open it.

It is true to some extent that some of those emails really are harmless from the end user's perspective. Spam emails saw a rise in the last couple of years because of the growth of social media and e-commerce websites. Companies, for example, usually broadcast their "latest news" or announcements over email to large numbers of people who are a part of an opt-in list.

However, with the right kind of planned attack, spamming could prove to be fatal for companies if not the users. If a hacker is somehow able to gain control of an organization's email, they can send unsolicited emails to even larger numbers of people.

Worse, since the emails are going out from legitimate email addresses, hackers could take advantage of the situation and send emails with a phishing attack or by attaching a virus within an email, hence infecting large amount of users simultaneously.

On the other end, a company could also face some serious consequences such as being questioned by the governing authorities for the spam emails. They risk having their internet connection shut down by their internet service providers, which can bring the company's operations to a complete halt.

LECTURE 31:

31.1: Message integrity:

The validity of a transmitted message. Message integritymeans that a message has not been tampered with or altered. The most common approach is to use a hash function that combines all the bytes in the message with a secret key and produces a message digest that is difficult to reverse.

31.2:Non-repudiation:

Nonrepudiation is the assurance that someone cannot deny something. Typically, nonrepudiation refers to the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.

To repudiate means to deny. For many years, authorities have sought to make repudiation impossible in some situations. You might send registered mail, for example, so the recipient cannot deny that a letter was delivered. Similarly, a legal document typically requires witnesses to signing so that the person who signs cannot deny having done so.

On the Internet, a digital signature is used not only to ensure that a message or document has been electronically signed by the person that purported to sign the document, but also, since a digital signature can only be created by one person, to ensure that a person cannot later deny that they furnished the signature.

Since no security technology is absolutely fool-proof, some experts warn that a digital signature alone may not always guarantee nonrepudiation. It is suggested that multiple approaches be used, such as capturing unique biometric information and other data about the sender or signer that collectively would be difficult to repudiate.

Email nonrepudiation involves methods such as email tracking that are designed to ensure that the sender cannot deny having sent a message and/or that the recipient cannot deny having received it.

31.3:Pretty Good Privacy:
PGP is an encryption program that provides cryptographic privacy and authentication for data communication. PGP is used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications.

31.4: S/MIME :
Secure MIME (S/MIME) is an Internet standard for digitally signing MIME-based email data and its public key encryption. It was initially developed by RSA Security, Inc. and is based on the company's public key encryption mechanism. Most email services and software use S/MIME to secure email communication.

LECTURE32:
32.1: IP Security :
The IP security (IPSec) is an Internet Engineering Task Force (IETF) standard suite of protocols between 2 communication points across the IPnetwork that provide data authentication, integrity, and confidentiality. It also defines the encrypted, decrypted and authenticated packets.

Overview of IPSec:
The IP Security Architecture (IPsec) provides cryptographic protection for IP datagrams in IPv4 and IPv6 network packets. The protection can include confidentiality, strong integrity of the data, data authentication, and partial sequence integrity. ... IPsec is performed inside the IP module.

IPv4 and IPv6-Authentication Header :
IPv6 extension headers contain supplementary information used by network devices (such as routers, switches, and endpoint hosts) to decide how to direct or process an IPv6 packet. The

length of each extension header is an integer multiple of 8 octets. This allows subsequent extension headersto use 8-octet structures.

Encapsulation Security Payload :
An Encapsulating Security Payload (ESP) is a protocol within the IPSec for providing authentication, integrity and confidentially of network packets data/payload in IPv4 and IPv6 networks. ESP provides message/payload encryption and the authentication of a payload and its origin within the IPSec protocol suite.

LECTURE 33:
33.1: Internet Key Exchange :
In computing, Internet Key Exchange (IKE, sometimes IKEv1 or IKEv2, depending on version) is the protocol used to set up a security association (SA) in the IPsec protocol suite. IKE builds upon the Oakley protocol and ISAKMP.

Phases of IKE:
In computing, Internet Key Exchange (IKE, sometimes IKEv1 or IKEv2, depending on version) is the protocol used to set up a security association (SA) in the IPsec protocol suite. IKE builds upon the Oakley protocol and ISAKMP IKE uses X.509 certificates for authentication – either pre-shared or distributed using DNS (preferably with DNSSEC) – and a Diffie–Hellman key exchange to set up a shared session secret from which cryptographic keys are derived. In addition, a security policy for every peer which will connect must be manually maintained.

Web Security:
Transport Layer Security (TLS) is the successor protocol to SSL. TLS is an improved version of SSL. It works in much the same way as the SSL, using encryption to protect the transfer of data and information.

33.2: Client authentication:
Client Authentication is the process by which users securely access a server or remote computer by exchanging a Digital Certificate. The Digital Certificate is in part seen as your 'Digital ID' and is used to cryptographically bind a customer, employee, or partner's identity to a unique Digital Certificate (typically including the name, company name and location of the Digital Certificate owner). The Digital Certificate can then be mapped to a user account and used to provide access control to network resources, web services and websites.

Just as organizations need to control which individual users have access to corporate networks and resources, they also need to be able to identify and control which machines and servers have access. Implementing device authentication means only machines with the appropriate credentials can access, communicate, and operate on corporate networks.

Organizations can leverage the registry information stored in Active Directory to automatically issue template-based and optionally configured certificates to all machines and servers residing within a single domain, or multiple domains in a single or multiple forest configuration.

The Digital Certificates used for client and device authentication may look the same as any other Digital Certificate that you may already be using within your organization, such as certificates for securing web services (SSL) or email/document signatures (digital signatures), but Digital Certificates are likely to have a few different properties depending on the use.

Client authentication can be used to prevent unauthorized access, or simply to add a second layer of security to your current username and password combination. Client authentication and access control also enables organizations to meet regulatory and privacy compliancy, as well as fulfil internal security policies using PKI-based two-factor authentication – 'something you have' (a GlobalSign Digital Certificate) and 'something you know' (an internally managed password).

The benefits of client authentication:
Client authentication has multiple benefits as an authentication method especially when compared to the basic username and password method:

- You can decide whether or not a user is required to enter a username and password
- Encrypts transactions over the network, identifies the server and validates any messages sent
- Validates the user identity using a trusted party (the Certificate Authority) and allows for centralized management of certificates which enables easy revocation
- Optional - you can configure the certificate so it cannot be exported to other devices, making it unique to the device it is installed on
- Restrict access by user, group, roles, or device based on Active Directory (using GlobalSign's Auto Enrolment Gateway (AEG) solution)
- Serves more purposes than authentication such as integrity and confidentiality
- Prevents malicious attacks/problems, including but not limited to phishing, keystroke logging and man-in-the-middle (MITM) attacks

Support for client authentication:
Many enterprise applications and networks natively support X.509 Digital Certificates, the standard format for public key certificates. This means with just a few configuration changes, you can enable client authentication for many popular use cases, including Windows logon, Google Apps, Salesforce, SharePoint, SAP and access to remote servers via portals like Citrix or SonicWALL.

This means:

- Minimal configuration is needed to implement strong authentication
- Easily enable two-factor authentication across multiple applications and networks
- Support a mobile/remote workforce

The below images are an example of using X.509 Digital Certificates as a method of two-factor authentication. First the user will login with their own username and password: